ARM® Compiler toolchain

Version 5.03

Errors and Warnings Reference

ARM DUI 0496I (ID012213)

ARM Compiler toolchain
Errors and Warnings Reference

Release Information

Change History

Date	Issue	Confidentiality	Change
May 2010	A	Non-Confidential	ARM Compiler toolchain v4.1 Release
30 September 2010	B	Non-Confidential	Update 1 for ARM Compiler toolchain v4.1
28 January 2011	C	Non-Confidential	Update 2 for ARM Compiler toolchain v4.1 Patch 3
30 April 2011	D	Non-Confidential	ARM Compiler toolchain v5.0 Release
29 July 2011	E	Non-Confidential	Update 1 for ARM Compiler toolchain v5.0
30 September 2011	F	Non-Confidential	ARM Compiler toolchain v5.01 Release
29 February 2012	G	Non-Confidential	Document update 1 for ARM Compiler toolchain v5.01 Release
27 July 2012	H	Non-Confidential	ARM Compiler toolchain v5.02 Release
31 January 2013	I	Non-Confidential	ARM Compiler toolchain v5.03 Release

Proprietary Notice

Confidentiality Status

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
ARM Compiler toolchain Errors and Warnings Reference

Chapter 1
Conventions and feedback

The following describes the typographical conventions and how to give feedback:

Typographical conventions

The following typographical conventions are used:

`monospace` Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.

`mono`space Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.

`monospace italic`

Denotes arguments to commands and functions where the argument is to be replaced by a specific value.

`monospace bold`

Denotes language keywords when used outside example code.

italic Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold Highlights interface elements, such as menu names. Also used for emphasis in descriptive lists, where appropriate, and for ARM® processor signal names.

Feedback on this product

If you have any comments and suggestions about this product, contact your supplier and give:

- your name and company

- the serial number of the product
- details of the release you are using
- details of the platform you are using, such as the hardware platform, operating system type and version
- a small standalone sample of code that reproduces the problem
- a clear explanation of what you expected to happen, and what actually happened
- the commands you used, including any command-line options
- sample output illustrating the problem
- the version string of the tools, including the version number and build numbers.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- the title
- the number, ARM DUI 0496I
- if viewing online, the topic names to which your comments apply
- if viewing a PDF version of a document, the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

ARM periodically provides updates and corrections to its documentation on the ARM Information Center, together with knowledge articles and *Frequently Asked Questions* (FAQs).

Other information

- ARM Information Center, http://infocenter.arm.com/help/index.jsp
- ARM Technical Support Knowledge Articles, http://infocenter.arm.com/help/topic/com.arm.doc.faqs/index.html
- ARM Support and Maintenance, http://www.arm.com/support/services/support-maintenance.php
- ARM Glossary, http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

Chapter 2
C and C++ Compiler Errors and Warnings

The following topics describe the error and warning messages for the C and C++ compiler, `armcc`:

2.1 Internal errors and other unexpected failures

Internal errors in the compiler are typically errors that have occurred but have not yet been documented, or they might point to a potential issue in the compiler itself.

For example:

```
Internal fault: [0x87ecef:5010591]
```

contains:

- the message description (`Internal Fault`)
- a six hex digit fault code for the error that occurred (`0x87ecef`).

  In RVCT 2.2 and earlier this was a 4 digit code.
- the version number (501 is ARM Compiler 5.01)
- the build number (0591 in this example).

If you see an internal fault, contact your supplier.

To facilitate the investigation, try to send only the single source file or function that is causing the error, plus the compiler options used when compiling the code.

It might be necessary to preprocess the file (that is, to take account of files added with `#include`). To do this, pass the file through the preprocessor as follows:

```
armcc <options> -E sourcefile.c > PPsourcefile.c
```

where `<options>` are your normal compile switches, such as `-O2`, `-g`, `-I`, `-D`, but without `-c`.

Check that the error is still reproducible with the preprocessed file by compiling it with:

```
armcc <options> -c PPsourcefile.c
```

and then provide the `PPsourcefile.c` file and your compile `<options>` to your supplier.

2.2 Suppressing armcc error and warning messages

The compiler normally warns of potential portability problems and other hazards.

When porting legacy code (for example, in old-style C) to the ARM, many warnings might be reported. It might be tempting to disable all such warnings with `-W`. ARM recommends however that for portability reasons, you change the code to make it ANSI compatible rather than suppressing the warnings.

Some warnings are suppressed by default. To override this, use the `--strict_warnings` switch to enable all suppressed warnings.

By default optimization messages, that is most of the messages between 1593 and 2159, are not warnings. To treat optimization messages as warnings, use the `--diag_warning=optimizations` option.

2.2.1 See also

Reference

Compiler Reference:

- *--diag_warning=tag[,tag,...] on page 3-75*
- *--strict_warnings on page 3-195*
- *-W on page 3-220*.

2.3 List of the armcc error and warning messages

The error and warning messages for armcc are:

0 unknown error

1 last line of file ends without a new line

2 last line of file ends with a backslash

3 #include file <entity> includes itself

4 out of memory

5 cannot open <entity> input file <filename>: <reason>

For example:

```
#include <file.h>
```

results in the message:

```
Error: #5: cannot open source input file "file.h": No such file or
directory
```

because file.h does not exist in the system include directory.

6 comment unclosed at end of file

Comment started with /* but no matching */ to close the comment.

7 unrecognized token

8 missing closing quote

For example:

```
char foo[] = {"\"};
```

In this example, the backslash causes the following quote " to be treated as a literal character rather than closing the string. To fix this, use:

```
char foo[] = {"\\"};
```

9 nested comment is not allowed

For example:

```
/*nested
/*comment*/
```

10 "#" not expected here

11 unrecognized preprocessing directive

For example:

```
#foo
```

12 parsing restarts here after previous syntax error

13 expected a file name

For example:

```
#include <stdio.h>
```

14 extra text after expected end of preprocessing directive

For example:

```
#if EMBEDDED foo
```

or:

```
#include <stdio.h> foo
```

or:

```
#ifdef SOMETHING
:
#endif SOMETHING
```

The `#endif` does not expect or require any argument. Enclosing the trailing part of the line in a comment fixes the problem:

```
#endif /* SOMETHING */
```

16 `<entity> is not a valid source file name`

17 `expected a "]"`

18 `expected a ")"`

For example:

```
int main(void
{
```

where there is a missing `)`.

19 `extra text after expected end of number`

For example:

```
int a = 37r;
```

20 `identifier <entity> is undefined`

For example, when compiled for C++, the code:

```
void foo( arg ) { }
```

results in the message:

```
Error: #20: identifier <arg> is undefined
```

Another example of code that can cause this error is:

```
int foo(void)
{
  int a = 4;
  a = i;
}
```

which results in the message:

```
Error: #20: identifier "i" is undefined
```

because `i` has not been declared.

21 `type qualifiers are meaningless in this declaration`

22 `invalid hexadecimal number`

23 `integer constant is too large`

24 `invalid octal digit`

For example:

```
int a = 0378;
```

25 `quoted string should contain at least one character`

For example:

```
char a ='';
```

26 `too many characters in character constant`

For example:

```
char a ='abcd';
```

results in an error.

Note

Only one character is permitted in a single-quoted string. For more than one character, double quotes must be used. Strings must be assigned to an appropriate variable such as `a[]`.

27 `character value is out of range`

For example:

```
char foo[] = {"\xBBBB" };
```

results in the message:

```
Warning: #27-D: character value is out of range
```

28 `expression must have a constant value`

29 `expected an expression`

30 `floating constant is out of range`

31 `expression must have integral type`

32 `expression must have arithmetic type`

33 `expected a line number`

34 `invalid line number`

35 `#error directive: <entity>`

36 `the #if for this directive is missing`

37 `the #endif for this directive is missing`

An open `#if` was still active, but was not closed with `#endif` before the End Of File.

38 `directive is not allowed -- an #else has already appeared`

39 `division by zero`

40 `expected an identifier`

This error is raised if preprocessor statements are incorrectly formatted such as for example, if the identifier which must immediately follow a preprocessor command is missing. For example, a missing identifier after `#define` results in:

```
Error: #40: expected an identifier
```

This error can also occur when C code containing C++ keywords is compiled with the C++ compiler, for example:

```
int *new(void *p) { return p; }
```

causes an error because `new` is a keyword in C++.

41 `expression must have arithmetic or pointer type`

42 `operand types are incompatible (<type> and <type>)`

44 `expression must have pointer type`

45 `#undef may not be used on this predefined name`

46 `<entity> is predefined; attempted redefinition ignored`

47 `incompatible redefinition of macro <entity>`

Macro has been defined twice (with different replacement strings).

If it is necessary to do this, undefine the macro (`#undef`) before the second definition.

For example:

```
#define TEST 0
#define TEST 1
```

causes the compiler to produce:

```
Warning: #47-D: incompatible redefinition of macro "TEST" (declared at
line 1)
```

There is no way to control this error directly by a compiler option, but you can use conditional preprocessing. For example:

```
#ifdef TEST_EQUALS_ZERO
#define TEST 0
#else
#define TEST 1
#endif
```

Compiling with `armcc -c foo.c` defines `TEST` to be 1 (the default).

Compiling with `armcc -c -DTEST_EQUALS_ZERO foo.c` defines `TEST` to be 0.

49 `duplicate macro parameter name`

50 `"##" may not be first in a macro definition`

51 `"##" may not be last in a macro definition`

52 `expected a macro parameter name`

53 `expected a ":"`

54 `too few arguments in macro invocation`

55 `too many arguments in macro invocation`

56 `operand of sizeof may not be a function`

57 `this operator is not allowed in a constant expression`

58 `this operator is not allowed in a preprocessing expression`

59 `function call is not allowed in a constant expression`

60 `this operator is not allowed in an integral constant expression`

61 `integer operation result is out of range`

62 `shift count is negative`

63 `shift count is too large`

64 `declaration does not declare anything`

For example:

```
int;
```

65 `expected a ";"`

66 `enumeration value is out of "int" range`

This diagnostic message is generated by the compiler when an `enum` constant is outside the range of a signed int.

For example, the compiler generates this warning when the following code is compiled in C mode:

```
typedef enum
{
  Bit31 = 0x80000000
} Bits;
```

Note

This description applies to RVCT 2.2 and later.

C mode:

- the warning is produced but the compiler promotes the constants to unsigned
- the switch `--strict` always produces this message as an error.

C++ mode:

- by default the out-of-range constants are promoted to unsigned without a warning and also when `--strict` is used

As a work around for cases where the message is an error use the following code example:

```
typedef enum
{
  Bit31 = (int)0x80000000
} Bits;
```

An overflow no longer occurs, and so no error is reported.

Note

The value of `Bit31` is now negative because it is a signed int.

See the following in in the *Compiler Reference*:

- *Structures, unions, enumerations, and bitfields on page 6-9*.

67 `expected a "}"`

68 `integer conversion resulted in a change of sign`

The constant is too large to be represented in a signed long, and therefore has been given unsigned type.

Example:

```
long l = 2147483648;
```

69 `integer conversion resulted in truncation`

70 `incomplete type is not allowed`

Example:

```
typedef struct {
  unsigned char size;
  char string[];
} FOO;
```

By not declaring a size for the array in the structure, the compiler is not able to allocate a size of the structure. Incomplete types are permitted in `--gnu` and `--c99` modes.

71 `operand of sizeof may not be a bit field`

76 `argument to macro is empty`

77 `this declaration has no storage class or type specifier`

78 `a parameter declaration may not have an initializer`

79 `expected a type specifier`

The ellipses to denote variadic functions, such as, for example, `printf()`, must follow at least one parameter. Change:

```
int foo( ... );
```

to:

```
int foo( int bar, ... );
```

80 `a storage class may not be specified here`

81 `more than one storage class may not be specified`

82 `storage class is not first`

83 `type qualifier specified more than once`

84 `invalid combination of type specifiers`

The type name or type qualifier cannot be used in the same declaration as the second type name or type qualifier. For example:

```
typedef int int;
```

85 `invalid storage class for a parameter`

86 `invalid storage class for a function`

87 `a type specifier may not be used here`

88 `array of functions is not allowed`

89 `array of void is not allowed`

90 `function returning function is not allowed`

91 `function returning array is not allowed`

92 `identifier-list parameters may only be used in a function definition`

93 `function type may not come from a typedef`

94 `the size of an array must be greater than zero`

Zero-sized arrays are permitted only when in GNU mode, `--gnu`. For example:

```
char name[0];
```

See the following in the *Compiler Reference*:

- *--gnu on page 3-107*
- *GNU extensions to the C and C++ languages on page 4-48*.

95 `array is too large`

There is a limit of 4GB on the maximum size of arrays or structures.

96 `a translation unit must contain at least one declaration`

97 `a function may not return a value of this type`

98 `an array may not have elements of this type`

99 `a declaration here must declare a parameter`

100 `duplicate parameter name`

101 `<entity> has already been declared in the current scope`

102 `forward declaration of enum type is nonstandard`

103 `class is too large`

104 `struct or union is too large`

There is a limit of 4GB on the maximum size of arrays or structures.

105 `invalid size for bit field`

Bit fields must not be larger than the size of the type.

For example, with `--strict`:

```
struct X{
  int y:5000;
};
```

106 `invalid type for a bit field`

Bit fields must have integral type.

Example:

```
struct X{
  float x:5;
  float y:2;
};
```

107 `zero-length bit field must be unnamed`

108 `signed bit field of length 1`

109 `expression must have (pointer-to-) function type`

110 `expected either a definition or a tag name`

111 `statement is unreachable`

112 `expected "while"`

114 `<entity> was referenced but not defined`

115 `a continue statement may only be used within a loop`

116 `a break statement may only be used within a loop or switch`

Example:

```
void foo(void){
  int a=0;
  continue;
}
```

or:

```
void bar(void){
  int a=0;
  break;
}
```

117 `non-void <entity> should return a value`

118 `a void function may not return a value`

119 `cast to type <type> is not allowed`

120 `return value type does not match the function type`

121 `a case label may only be used within a switch`

122 `a default label may only be used within a switch`

123 `case label value has already appeared in this switch`

124 `default label has already appeared in this switch`

125 `expected a "("`

126 `expression must be an lvalue`

127 `expected a statement`

128 `loop is not reachable from preceding code`

129 `a block-scope function may only have extern storage class`

130 `expected a "{"`

131 `expression must have pointer-to-class type`

132 `expression must have pointer-to-struct-or-union type`

133 `expected a member name`

134 `expected a field name`

135 `<entity> has no member <entity>`

136 `<entity> has no field <entity>`

137 `expression must be a modifiable lvalue`

138 `taking the address of a register variable is not allowed`

139 `taking the address of a bit field is not allowed`

140 `too many arguments in function call`

Function declaration does not match the number of parameters in an earlier function prototype.

Example:

```
extern void foo(int x);
void bar(void)
{
  foo(1,2);
}
```

141 `unnamed prototyped parameters not allowed when body is present`

142 `expression must have pointer-to-object type`

143 `program too large or complicated to compile`

144 `a value of type <type> cannot be used to initialize an entity of type <type>`

The initializing string for a fixed size character array is exactly as long as the array size, leaving no room for a terminating `\0`, for example:

```
char name[5] = "Hello";
```

The name array can hold up to 5 characters. `"Hello"` does not fit because C strings are always null-terminated (for example, `"Hello\0"`). The compiler reports:

```
Error: #144: a value of type "const char [6]" cannot be used to initialize
an entity of type "char [5]"
```

A similar error is also raised if there is an implicit cast of non-zero int to pointer.

For example:

```
void foo_func( void )
{
  char *foo=1;
}
```

results in the message:

```
#144: a value of type "int" cannot be used to initialize an entity of type
"char *"
```

For the cast, this error can be suppressed with the use of the `--loose_implicit_cast` switch.

145 `<entity> may not be initialized`

146 `too many initializer values`

147 `declaration is incompatible with <entity>`

This incorrect C code:

```
typedef enum { e } E;
typedef enum { f } F;
E g(void);
F g(void);
```

is a discretionary error in all modes, and can be downgraded from an Error to a Warning with `--diag_warning 147`, or suppressed completely with `--diag_suppress 147`.

148 `<entity> has already been initialized`

149 `a global-scope declaration may not have this storage class`

150 `a type name may not be redeclared as a parameter`

151 `a typedef name may not be redeclared as a parameter`

152 `conversion of nonzero integer to pointer`

153 `expression must have class type`

154 `expression must have struct or union type`

155 `old-fashioned assignment operator`

156 `old-fashioned initializer`

157 `expression must be an integral constant expression`

158 `expression must be an lvalue or a function designator`

159 `declaration is incompatible with previous <entity>`

160 `external name conflicts with external name of <entity>`

161 `unrecognized #pragma`

163 `could not open temporary file <entity>`

164 `name of directory for temporary files is too long (<entity>)`

165 `too few arguments in function call`

Function prototype is defined with a number of parameters that does not match the number of parameters passed in the function call.

For example:

```
extern void foo(int x);
void bar(void)
{
  foo();
}
```

166 `invalid floating constant`

167 `argument of type <type> is incompatible with parameter of type <type>`

168 `a function type is not allowed here`

169 `expected a declaration`

This can occur when attempting to compile some C++ header files with the C compiler instead of the C++ compiler. The following message is reported:

```
Error: #169: expected a declaration
```

170 `pointer points outside of underlying object`

171 `invalid type conversion`

172 `external/internal linkage conflict with previous declaration`

Errors about linkage disagreements where functions are implicitly declared as extern and then later re-declared as static are suppressed unless compiled with --strict.

Example:

```
extern void foo(void);
static void foo(void){}
```

173 `floating-point value does not fit in required integral type`

174 `expression has no effect`

175 `subscript out of range`

177 `<entity> was declared but never referenced`

By default, unused declaration warnings are given for:

- local (within a function) declarations of variables, typedefs, and functions
- labels (always within a function)
- top-level static functions and static variables.

The `--diag_suppress 177` option suppresses these warnings.

178 `"&" applied to an array has no effect`

179 `right operand of "%" is zero`

180 `argument is incompatible with formal parameter`

181 `argument is incompatible with corresponding format string conversion`

For example when compiling with `--strict`, the code:

```
unsigned long foo = 0x1234;
printf("%08X", foo);
```

results in the warning:

```
Warning: #181-D: argument is incompatible with corresponding format string
conversion
```

To avoid the warning, the code could be rewritten as:

```
unsigned long foo = 0x1234;
printf("%0lX", foo);
```

or perhaps:

```
unsigned int foo = 0x1234;
printf("%0X", foo);
```

`%0X` can be used for char, short or int. Use `lX` for a long integer, even though both ints and longs are 32-bits wide on an ARM.

182 `could not open source file <entity> (no directories in search list)`

183 `type of cast must be integral`

184 `type of cast must be arithmetic or pointer`

185 `dynamic initialization in unreachable code`

186 `pointless comparison of unsigned integer with zero`

For example:

```
unsigned short foo;
if (foo<0) printf("This never happens");
```

gives a warning that the comparison between an unsigned (for example, char or int) value and zero always evaluates to false.

187 `use of "=" where "==" may have been intended`

Example:

```
int main(void)
{
  int a;
  const int b =1;
  if (a=b)
}
```

188 `enumerated type mixed with another type`

189 `error while writing <entity> file`

190 `invalid intermediate language file`

191 `type qualifier is meaningless on cast type`

The C specification states that a cast does not yield an lvalue, so a cast to a qualified type has the same effect as a cast to the unqualified version of the type. This warning is to inform you that the type qualifier has no effect, although the code is still legal. The warning is suppressible with `--diag_suppress 191`.

Example:

```
"val2 = (const float)val1;" is equivalent to "val2 = (float)val1;"
```

192 `unrecognized character escape sequence`

This error is commonly associated with the attempted use of non-ASCII character sets, such as 16-bit Unicode characters. The compiler supports multibyte character sets, such as Unicode. Source files are compiled according to the selected locale of that machine. It is possible to use *Escape processing* (as recommended by Kernighan and Ritchie, section A2.5.2) to encode specific values instead.

For example:

```
char *p = "\x12\x34\x56\x78"; // 12 34 56 78
```

In character and string escapes, if the character following the \ has no special meaning, the value of the escape is the character itself, for example, \s is the same as s and the warning is given.

193 `zero used for undefined preprocessing identifier <entity>`

194 `expected an asm string`

195 `an asm function must be prototyped`

196 `an asm function may not have an ellipsis`

219 `error while deleting file <entity>`

220 `integral value does not fit in required floating-point type`

221 `floating-point value does not fit in required floating-point type`

222 `floating-point operation result is out of range`

223 `function <entity> declared implicitly`

This is a common warning that occurs where there is no prototype for a function.

For example:

```
void foo(void)
{
  printf("foo");
}
```

To fix this, add `#include <stdio.h>` that includes the prototype for `printf`.

For ANSI C, you can suppress this warning with `--diag_suppress 223`. This is useful when compiling old-style C in ANSI C mode.

224 `the format string requires additional arguments`

225 `the format string ends before this argument`

226 `invalid format string conversion`

227 `macro recursion`

228 `trailing comma is nonstandard`

229 `bit field cannot contain all values of the enumerated type`

230 `nonstandard type for a bit field`

In strict ANSI C, the only types permitted for a bit field are `int`, `signed int`, and `unsigned int`.

Example:

```
struct X {
char y:2;
};
```

231 `declaration is not visible outside of function`

232 `old-fashioned typedef of "void" ignored`

233 `left operand is not a struct or union containing this field`

234 `pointer does not point to struct or union containing this field`

235 `variable <entity> was declared with a never-completed type`

236 `controlling expression is constant`

237 `selector expression is constant`

238 `invalid specifier on a parameter`

239 `invalid specifier outside a class declaration`

240 `duplicate specifier in declaration`

241 `a union is not allowed to have a base class`

242 `multiple access control specifiers are not allowed`

243 `class or struct definition is missing`

244 `qualified name is not a member of class <type> or its base classes`

245 `a nonstatic member reference must be relative to a specific object`

246 `a nonstatic data member may not be defined outside its class`

247 `<entity> has already been defined`

A typical example of this is where a variable name has been used more than once.

This can sometimes occur when compiling legacy code that relies on tentative declarations. Tentative declarations permit a variable to be declared and initialized as separate statements such as:

```
int a;
int a = 1;
```

Tentative declarations are permitted by default for C code, but produce an error with C++ code.

248 `pointer to reference is not allowed`

249 `reference to reference is not allowed`

250 `reference to void is not allowed`

251 `array of reference is not allowed`

252 `reference <entity> requires an initializer`

253 `expected a ","`

254 `type name is not allowed`

This occurs when a typedef name is being used directly in an expression:

```
typedef int footype;
int x = footype; // reports Error: #254: type name is not allowed
```

To fix this, first create an instance of that type (for example, a variable of the new type):

```
typedef int footype;
footype bar = 1;
int x = bar;
```

255 `type definition is not allowed`

256 `invalid redeclaration of type name <entity>`

257 `const <entity> requires an initializer`

258 `"this" may only be used inside a nonstatic member function`

259 `constant value is not known`

260 explicit type is missing ("int" assumed)

261 access control not specified (<entity> by default)

262 not a class or struct name

263 duplicate base class name

264 invalid base class

265 <entity> is inaccessible

For C++ only, the `--diag_warning 265` option downgrades access control errors to warnings.

For example:

```
class A { void f() {}; }; // private member
A a;
void g() { a.f(); } // erroneous access
```

results in the message:

```
Error: #265-D: function "A::f" is inaccessible
```

266 <entity> is ambiguous

267 old-style parameter list (anachronism)

268 declaration may not appear after executable statement in block

269 conversion to inaccessible base class <type> is not allowed

274 improperly terminated macro invocation

276 name followed by "::" must be a class or namespace name

277 invalid friend declaration

278 a constructor or destructor may not return a value

279 invalid destructor declaration

280 declaration of a member with the same name as its class

281 global-scope qualifier (leading "::") is not allowed

282 the global scope has no <entity>

283 qualified name is not allowed

284 NULL reference is not allowed

285 initialization with "<...>" is not allowed for object of type <type>

286 base class <type> is ambiguous

287 derived class <type> contains more than one instance of class <type>

288 cannot convert pointer to base class <type> to pointer to derived class <type> -- base class is virtual

289 no instance of constructor <entity> matches the argument list

290 copy constructor for class <type> is ambiguous

291 no default constructor exists for class <type>

292 <entity> is not a nonstatic data member or base class of class <type>

293 indirect nonvirtual base class is not allowed

294 invalid union member -- class <type> has a disallowed member function

296 invalid use of non-lvalue array

297 expected an operator

298 inherited member is not allowed

299 cannot determine which instance of <entity> is intended

300 a pointer to a bound function may only be used to call the function

301 typedef name has already been declared (with same type)

302 <entity> has already been defined

304 no instance of <entity> matches the argument list

305 type definition is not allowed in function return type declaration

306 default argument not at end of parameter list

307 redefinition of default argument

308 more than one instance of <entity> matches the argument list:

309 more than one instance of constructor <entity> matches the argument list:

310 default argument of type <type> is incompatible with parameter of type <type>

311 cannot overload functions distinguished by return type alone

312 no suitable user-defined conversion from <type> to <type> exists

313 type qualifier is not allowed on this function

314 only nonstatic member functions may be virtual

315 the object has cv-qualifiers that are not compatible with the member function

316 program too large to compile (too many virtual functions)

317 return type is not identical to nor covariant with return type <type> of overridden virtual function <entity>

318 override of virtual <entity> is ambiguous

319 pure specifier ("= 0") allowed only on virtual functions

320 badly-formed pure specifier (only "= 0" is allowed)

321 data member initializer is not allowed

322 object of abstract class type <type> is not allowed:

323 function returning abstract class <type> is not allowed:

324 duplicate friend declaration

325 inline specifier allowed on function declarations only

326 "inline" is not allowed

327 invalid storage class for an inline function

328 `invalid storage class for a class member`

329 `local class member <entity> requires a definition`

330 `<entity> is inaccessible`

332 `class <type> has no copy constructor to copy a const object`

333 `defining an implicitly declared member function is not allowed`

334 `class <type> has no suitable copy constructor`

335 `linkage specification is not allowed`

336 `unknown external linkage specification`

337 `linkage specification is incompatible with previous <entity>`

If the linkage for a function is redeclared with an incompatible specification to a previous declaration this error is produced.

For example:

```
int foo(void);
int bar(void)
{
  int x;
  x = foo();
  return x;
}
extern "C" int foo(void)
{
  return 0;
}
```

results in the message:

```
Error: #337: linkage specification is incompatible with previous "foo"
(declared at line 1)
```

338 `more than one instance of overloaded function <entity> has "C" linkage`

339 `class <type> has more than one default constructor`

340 `value copied to temporary, reference to temporary used`

341 `"operator<entity>" must be a member function`

342 `operator may not be a static member function`

343 `no arguments allowed on user-defined conversion`

344 `too many parameters for this operator function`

345 `too few parameters for this operator function`

346 `nonmember operator requires a parameter with class type`

347 `default argument is not allowed`

348 `more than one user-defined conversion from <type> to <type> applies:`

349 `no operator <entity> matches these operands`

350 `more than one operator <entity> matches these operands:`

351 `first parameter of allocation function must be of type "size_t"`

352 `allocation function requires "void *" return type`

353 `deallocation function requires "void" return type`

354 `first parameter of deallocation function must be of type "void *"`

356 `type must be an object type`

357 `base class <type> has already been initialized`

358 `base class name required -- <type> assumed (anachronism)`

359 `<entity> has already been initialized`

360 `name of member or base class is missing`

361 `assignment to "this" (anachronism)`

362 `"overload" keyword used (anachronism)`

363 `invalid anonymous union -- nonpublic member is not allowed`

364 `invalid anonymous union -- member function is not allowed`

365 `anonymous union at global or namespace scope must be declared static`

366 `<entity> provides no initializer for:`

367 `implicitly generated constructor for class <type> cannot initialize:`

368 `<entity> defines no constructor to initialize the following:`

This indicates that you have a `const` structure or a structure containing a `const`. It is issued as a friendly warning to assist with error 369. This can safely be ignored providing that the `const` members of structures are appropriately initialized.

369 `<entity> has an uninitialized const or reference member`

This indicates that you have an instance of a `const` structure or structure containing a `const` that has not been correctly initialized. You must either initialize it correctly for every instance or provide a constructor to initialize it.

370 `<entity> has an uninitialized const field`

371 `class <type> has no assignment operator to copy a const object`

372 `class <type> has no suitable assignment operator`

373 `ambiguous assignment operator for class <type>`

375 `declaration requires a typedef name`

377 `"virtual" is not allowed`

378 `"static" is not allowed`

379 `cast of bound function to normal function pointer (anachronism)`

380 `expression must have pointer-to-member type`

381 `extra ";" ignored`

In C, this can be caused by an unexpected semicolon at the end of a declaration line, for example:

```
int x;;
```

This might occur inadvertently when using macros.

Similarly, in C++, this might be caused by constructions like:

```
class X { ... } ; ;
```

which probably resulted from some macro usage:

```
#define M(c) class c { ... } ;
M(X);
```

The extra semicolon is illegal because empty declarations are illegal.

382 nonstandard member constant declaration (standard form is a static const integral member)

384 no instance of overloaded <entity> matches the argument list

386 no instance of <entity> matches the required type

387 delete array size expression used (anachronism)

389 a cast to abstract class <type> is not allowed:

390 function "main" may not be called or have its address taken

391 a new-initializer may not be specified for an array

392 member function <entity> may not be redeclared outside its class

393 pointer to incomplete class type is not allowed

394 reference to local variable of enclosing function is not allowed

395 single-argument function used for postfix <entity> (anachronism)

398 cast to array type is nonstandard (treated as cast to <type>)

399 <entity> has an operator new<entity>() but no default operator delete<entity>()

400 <entity> has a default operator delete<entity>() but no operator new<entity>()

401 destructor for base class <entity> is not virtual

403 invalid redeclaration of member <entity>

404 function "main" may not be declared inline

405 member function with the same name as its class must be a constructor

406 using nested <entity> (anachronism)

407 a destructor may not have parameters

408 copy constructor for class <type> may not have a parameter of type <type>

409 <entity> returns incomplete type <type>

410 protected <entity> is not accessible through a <type> pointer or object

411 a parameter is not allowed

412 an "asm" declaration is not allowed here

413 no suitable conversion function from <type> to <type> exists

414 delete of pointer to incomplete class

415 no suitable constructor exists to convert from <type> to <type>

416 more than one constructor applies to convert from <type> to <type>:

417 more than one conversion function from <type> to <type> applies:

418 more than one conversion function from <type> to a built-in type applies:

424 a constructor or destructor may not have its address taken

427 qualified name is not allowed in member declaration

428 enumerated type mixed with another type (anachronism)

429 the size of an array in "new" must be non-negative

430 returning reference to local temporary

433 qualifiers dropped in binding reference of type <type> to initializer of type <type>

434 a reference of type <type> (not const-qualified) cannot be initialized with a value of type <type>

435 a pointer to function may not be deleted

436 conversion function must be a nonstatic member function

437 template declaration is not allowed here

438 expected a "<"

439 expected a ">"

440 template parameter declaration is missing

441 argument list for <entity> is missing

442 too few arguments for <entity>

443 too many arguments for <entity>

450 the type "long long" is nonstandard

451 omission of <entity> is nonstandard

452 return type may not be specified on a conversion function

456 excessive recursion at instantiation of <entity>

457 <entity> is not a function or static data member

458 argument of type <type> is incompatible with template parameter of type <type>

459 initialization requiring a temporary or conversion is not allowed

460 declaration of <entity> hides function parameter

461 initial value of reference to non-const must be an lvalue

463 "template" is not allowed

464 <type> is not a class template

467 invalid reference to <entity> (union/nonunion mismatch)

468 a template argument may not reference a local type

469 tag kind of <entity> is incompatible with declaration of <entity>

470 the global scope has no tag named <entity>

471 <entity> has no tag member named <entity>

473 <entity> may be used only in pointer-to-member declaration

476 name followed by "::~" must be a class name or a type name

477 destructor name does not match name of class <type>

478 type used as destructor name does not match type <type>

479 <entity> redeclared "inline" after being called

485 <entity> is not an entity that can be instantiated

486 compiler generated <entity> cannot be explicitly instantiated

487 inline <entity> cannot be explicitly instantiated

490 <entity> cannot be instantiated -- it has been explicitly specialized

494 declaring a void parameter list with a typedef is nonstandard

When the compiler is in ANSI C mode, this error might be produced by a function declaration `f(V)` where `V` is a void type.

In the special syntax `f(<void>)` that indicates that `f` is a function taking no arguments, the keyword <void> is required. The name of a void type cannot be used instead.

496 template parameter <entity> may not be redeclared in this scope

497 declaration of <entity> hides template parameter

498 template argument list must match the parameter list

501 an operator name must be declared as a function

502 operator name is not allowed

503 <entity> cannot be specialized in the current scope

504 nonstandard form for taking the address of a member function

The C++ standard requires that a pointer to a member be named using a qualified name and a & character such as for `&A::f`.

The front end previously accepted nonstandard forms like `&f`, or even `f`, as a concession to existing practice. This usage now produces a discretionary error.

505 too few template parameters -- does not match previous declaration

506 too many template parameters -- does not match previous declaration

507 function template for operator delete(void *) is not allowed

508 class template and template parameter may not have the same name

511 enumerated type is not allowed

512 type qualifier on a reference type is not allowed

513 a value of type <type> cannot be assigned to an entity of type <type>

514 pointless comparison of unsigned integer with a negative constant

515 cannot convert to incomplete class <type>

516 const object requires an initializer

517 object has an uninitialized const or reference member

518 nonstandard preprocessing directive

519 <entity> may not have a template argument list

520 initialization with "<...>" expected for aggregate object

521 pointer-to-member selection class types are incompatible (<type> and <type>)

522 pointless friend declaration

524 non-const function called for const object (anachronism)

525 a dependent statement may not be a declaration

526 a parameter may not have void type

For example:

```
void foo(void a) { }
```

529 this operator is not allowed in a template argument expression

530 try block requires at least one handler

531 handler requires an exception declaration

532 handler is masked by default handler

533 handler is potentially masked by previous handler for type <type>

534 use of a local type to specify an exception

535 redundant type in exception specification

536 exception specification is incompatible with that of previous <entity>

540 support for exception handling is disabled

541 omission of exception specification is incompatible with previous <entity>

542 could not create instantiation request file <entity>

543 non-arithmetic operation not allowed in nontype template argument

544 use of a local type to declare a nonlocal variable

545 use of a local type to declare a function

546 transfer of control bypasses initialization of:

Example:

```
int main(void){
   int choice = 1;
   int z =1;
   switch(choice)
   {
     case 1:
       int y = 1;
       z = y + z;
       break;
     case 2:
     break;
   }
   return 0;
```

In the example, y is an initialized variable that is in scope (but unused) in the other cases.

The C++ Standard says in section 6.7:

"It is possible to transfer into a block, but not in a way that bypasses declarations with initialization. A program that jumps from a point where a local variable with automatic storage duration is not in scope to a point where it is in scope is ill-formed unless the variable has POD type (3.9) and is declared without an initializer (8.5)."

Note

The transfer from the condition of a switch statement to a case label is considered a jump in this respect.

The usual way to fix this is to enclose the case that declares y in braces:

```
case 1:   {
  int y = 1;
  z = y + z;
}
break;
```

Because y is a POD (Plain Old Data) type, so an alternative is to not use initialization:

```
case 1:
 int y;
 y = 1;
 z = y + z;
 break;
```

548 transfer of control into an exception handler

549 <entity> is used before its value is set

550 <entity> was set but never used

551 <entity> cannot be defined in the current scope

552 exception specification is not allowed

553 external/internal linkage conflict for <entity>

554 <entity> will not be called for implicit or explicit conversions

555 tag kind of <entity> is incompatible with template parameter of type <type>

556 function template for operator new(size_t) is not allowed

558 pointer to member of type <type> is not allowed

559 ellipsis is not allowed in operator function parameter list

560 <entity> is reserved for future use as a keyword

561 invalid macro definition:

562 invalid macro undefinition:

563 invalid <entity> output file <filename>

564 cannot open <entity> output file <filename>: <reason>

570 error in debug option argument

571 invalid option:

574 invalid number:

576 invalid instantiation mode:

578 invalid error limit:

585 virtual function tables can only be suppressed when compiling C++

586 anachronism option can be used only when compiling C++

587 instantiation mode option can be used only when compiling C++

588 automatic instantiation mode can be used only when compiling C++

589 implicit template inclusion mode can be used only when compiling C++

590 exception handling option can be used only when compiling C++

593 missing source file name

594 output files may not be specified when compiling several input files

595 too many arguments on command line

596 an output file was specified, but none is needed

598 a template parameter may not have void type

600 strict mode is incompatible with allowing anachronisms

601 a throw expression may not have void type

602 local instantiation mode is incompatible with automatic instantiation

603 parameter of abstract class type <type> is not allowed:

604 array of abstract class <type> is not allowed:

605 floating-point template parameter is nonstandard

606 this pragma must immediately precede a declaration

607 this pragma must immediately precede a statement

608 this pragma must immediately precede a declaration or statement

609 this kind of pragma may not be used here

611 overloaded virtual function <entity> is only partially overridden in <entity>

612 specific definition of inline template function must precede its first use

613 invalid error tag in diagnostic control option:

614 invalid error number in diagnostic control option:

615 parameter type involves pointer to array of unknown bound

616 parameter type involves reference to array of unknown bound

617 pointer-to-member-function cast to pointer to function

618 struct or union declares no named members

619 nonstandard unnamed field

620 nonstandard unnamed member

624 <entity> is not a type name

625 cannot open precompiled header input file <entity>: <reason>

626 precompiled header file <entity> is either invalid or not generated by this version of the compiler

627 precompiled header file <entity> was not generated in this directory

628 header files used to generate precompiled header file <entity> have changed

629 the command line options do not match those used when precompiled header file <entity> was created

630 the initial sequence of preprocessing directives is not compatible with those of precompiled header file <entity>

631 unable to obtain mapped memory for <entity>: <reason>

This can occur if you are trying to use a large *PreCompiled Header* (PCH), and you have a size limitation on the TMP directory that the ARM Compiler toolchain uses. A possible workaround is to remove the TMP environment variable. This forces the tools to create temporary files in the current working directory.

See the following in *Introducing the ARM® Compiler toolchain*:

- *TMP and TMPDIR environment variables for temporary file directories* on page 2-28

632 "<entity>": using precompiled header file "<entity>"

633 "<entity>": creating precompiled header file "<entity>"

634 memory usage conflict with precompiled header file <entity>

This can occur if a PCH file cannot be mapped back into the build because the required parts of the address space of the compiler are not available.

See also error 631.

635 invalid PCH memory size

636 PCH options must appear first in the command line

637 insufficient memory for PCH memory allocation

638 precompiled header files may not be used when compiling several input files

639 insufficient preallocated memory for generation of precompiled header file (<entity> bytes required)

640 very large entity in program prevents generation of precompiled header file

641 <entity> is not a valid directory

642 cannot build temporary file name

643 "restrict" is not allowed

644 a pointer or reference to function type may not be qualified by "restrict"

645 <entity> is an unrecognized __declspec attribute

646 a calling convention modifier may not be specified here

647 conflicting calling convention modifiers

650 calling convention specified here is ignored

651 a calling convention may not be followed by a nested declarator

652 calling convention is ignored for this type

654 declaration modifiers are incompatible with previous declaration

655 the modifier <entity> is not allowed on this declaration

656 transfer of control into a try block

657 inline specification is incompatible with previous <entity>

658 closing brace of template definition not found

659 wchar_t keyword option can be used only when compiling C++

660 invalid packing alignment value

661 expected an integer constant

662 call of pure virtual function

A pure virtual function `pvfn` is being called, for example:

```
struct T { T(); virtual void pvfn() = 0; };
                    // a pure virtual function
T::T() { pvfn(); } // warning given here
```

By default, calling a pure virtual function results in:

1. a call to the library function `__cxa_pure_virtual()`
2. the `__cxa_pure_virtual()` function raising the signal `SIGPVFN`
3. the signal being trapped by the `default_signal_handler`
4. the handler displaying `Pure virtual fn called` on the console using semihosting.

See the following in the *Compiler Reference*:

- *Calling a pure virtual function on page E-3*.

663 invalid source file identifier string

664 a class template cannot be defined in a friend declaration

665 "asm" is not allowed

666 "asm" must be used with a function definition

667 "asm" function is nonstandard

668 ellipsis with no explicit parameters is nonstandard

669 "&..." is nonstandard

670 invalid use of "&..."

672 temporary used for initial value of reference to const volatile (anachronism)

673 a reference of type <type> cannot be initialized with a value of type <type>

674 initial value of reference to const volatile must be an lvalue

676 using out-of-scope declaration of <entity>

678 call of <entity> cannot be inlined

679 <entity> cannot be inlined

680 invalid PCH directory:

688 <entity> not found on pack alignment stack

689 empty pack alignment stack

690 RTTI option can be used only when compiling C++

691 <entity>, required for copy that was eliminated, is inaccessible

692 <entity>, required for copy that was eliminated, is not callable because reference parameter cannot be bound to rvalue

693 <typeinfo> must be included before typeid is used

694 <entity> cannot cast away const or other type qualifiers

695 the type in a dynamic_cast must be a pointer or reference to a complete class type, or void *

696 the operand of a pointer dynamic_cast must be a pointer to a complete class type

697 the operand of a reference dynamic_cast must be an lvalue of a complete class type

698 the operand of a runtime dynamic_cast must have a polymorphic class type

699 bool option can be used only when compiling C++

702 expected an "="

703 expected a declarator in condition declaration

704 <entity>, declared in condition, may not be redeclared in this scope

705 default template arguments are not allowed for function templates

706 expected a "," or ">"

707 expected a template parameter list

708 incrementing a bool value is deprecated

709 bool type is not allowed

710 offset of base class <entity> within class <entity> is too large

711 expression must have bool type (or be convertible to bool)

712 array new and delete option can be used only when compiling C++

713 <entity> is not a variable name

717 the type in a const_cast must be a pointer, reference, or pointer to member to an object type

718 a const_cast can only adjust type qualifiers; it cannot change the underlying type

719 mutable is not allowed

720 redeclaration of <entity> is not allowed to alter its access

722 use of alternative token "<:" appears to be unintended

723 use of alternative token "%:" appears to be unintended

724 namespace definition is not allowed

725 name must be a namespace name

726 namespace alias definition is not allowed

727 namespace-qualified name is required

728 a namespace name is not allowed

730 <entity> is not a class template

731 array with incomplete element type is nonstandard

732 allocation operator may not be declared in a namespace

733 deallocation operator may not be declared in a namespace

734 <entity> conflicts with using-declaration of <entity>

735 using-declaration of <entity> conflicts with <entity>

736 namespaces option can be used only when compiling C++

737 using-declaration ignored -- it refers to the current namespace

738 a class-qualified name is required

744 incompatible memory attributes specified

745 memory attribute ignored

746 memory attribute may not be followed by a nested declarator

747 memory attribute specified more than once

748 calling convention specified more than once

749 a type qualifier is not allowed

750 <entity> was used before its template was declared

751 static and nonstatic member functions with same parameter types cannot be overloaded

752 no prior declaration of <entity>

753 a template-id is not allowed

754 a class-qualified name is not allowed

755 <entity> may not be redeclared in the current scope

756 qualified name is not allowed in namespace member declaration

757 <entity> is not a type name

758 explicit instantiation is not allowed in the current scope

759 <entity> cannot be explicitly instantiated in the current scope

760 <entity> explicitly instantiated more than once

761 typename may only be used within a template

763 typename option can be used only when compiling C++

764 implicit typename option can be used only when compiling C++

765 nonstandard character at start of object-like macro definition

766 exception specification for virtual <entity> is incompatible with that of overridden <entity>

767 conversion from pointer to smaller integer

768 exception specification for implicitly declared virtual <entity> is incompatible with that of overridden <entity>

769 <entity>, implicitly called from <entity>, is ambiguous

770 option "explicit" can be used only when compiling C++

771 "explicit" is not allowed

772 declaration conflicts with <entity> (reserved class name)

773 only "()" is allowed as initializer for array <entity>

774 "virtual" is not allowed in a function template declaration

775 invalid anonymous union -- class member template is not allowed

776 template nesting depth does not match the previous declaration of <entity>

777 this declaration cannot have multiple "template <...>" clauses

778 option to control the for-init scope can be used only when compiling C++

779 <entity>, declared in for-loop initialization, may not be redeclared in this scope

780 reference is to <entity> -- under old for-init scoping rules it would have been <entity>

781 option to control warnings on for-init differences can be used only when compiling C++

782 definition of virtual <entity> is required here

783 empty comment interpreted as token-pasting operator "##"

784 a storage class is not allowed in a friend declaration

785 template parameter list for <entity> is not allowed in this declaration

786 <entity> is not a valid member class or function template

787 not a valid member class or function template declaration

788 a template declaration containing a template parameter list may not be followed by an explicit specialization declaration

789 explicit specialization of <entity> must precede the first use of <entity>

790 explicit specialization is not allowed in the current scope

791 partial specialization of <entity> is not allowed

792 `<entity> is not an entity that can be explicitly specialized`

793 `explicit specialization of <entity> must precede its first use`

794 `template parameter <entity> may not be used in an elaborated type specifier`

795 `specializing <entity> requires "template<>" syntax`

798 `option old_specializations can be used only when compiling C++`

799 `specializing <entity> without "template<>" syntax is nonstandard`

800 `this declaration may not have extern "C" linkage`

801 `<entity> is not a class or function template name in the current scope`

802 `specifying a default argument when redeclaring an unreferenced function template is nonstandard`

803 `specifying a default argument when redeclaring an already referenced function template is not allowed`

804 `cannot convert pointer to member of base class <type> to pointer to member of derived class <type> -- base class is virtual`

805 `exception specification is incompatible with that of <entity><entity>`

806 `omission of exception specification is incompatible with <entity>`

807 `unexpected end of default argument expression`

808 `default-initialization of reference is not allowed`

809 `uninitialized <entity> has a const member`

810 `uninitialized base class <type> has a const member`

811 `const <entity> requires an initializer -- class <type> has no explicitly declared default constructor`

812 `const object requires an initializer -- class <type> has no explicitly declared default constructor`

814 `strict mode is incompatible with long preserving rules`

815 `type qualifier on return type is meaningless`

For example:

```
__packed void foo( void ) { }
```

The `__packed` qualifier is ignored because the return type cannot be `__packed`.

816 `in a function definition a type qualifier on a "void" return type is not allowed`

817 `static data member declaration is not allowed in this class`

818 `template instantiation resulted in an invalid function declaration`

819 `"..." is not allowed`

821 `extern inline <entity> was referenced but not defined`

822 `invalid destructor name for type <type>`

824 destructor reference is ambiguous -- both <entity> and <entity> could be used

825 <entity> could be used

826 <entity> was never referenced

827 only one member of a union may be specified in a constructor initializer list

828 support for "new[]" and "delete[]" is disabled

829 "double" used for "long double" in generated C code

830 <entity> has no corresponding operator delete<entity> (to be called if an exception is thrown during initialization of an allocated object)

831 support for placement delete is disabled

832 no appropriate operator delete is visible

833 pointer or reference to incomplete type is not allowed

834 invalid partial specialization -- <entity> is already fully specialized

835 incompatible exception specifications

836 returning reference to local variable

837 omission of explicit type is nonstandard ("int" assumed)

A function has been declared or defined with no return type.

Example, with the code:

```
foo(void){
    int a;
}
```

an `int` result is assumed.

If you want it to return no result, use `void` as the return type. This is widespread in old-style C.

The `--diag_suppress 837` option suppresses this warning.

See also message number `938`, that is a special case of this message for `main()`.

838 more than one partial specialization matches the template argument list of <entity>

840 a template argument list is not allowed in a declaration of a primary template

841 partial specializations may not have default template arguments

842 <entity> is not used in template argument list of <entity>

844 the template argument list of the partial specialization includes a nontype argument whose type depends on a template parameter

845 this partial specialization would have been used to instantiate <entity>

846 this partial specialization would have been made the instantiation of <entity> ambiguous

847 expression must have integral or enum type

848 expression must have arithmetic or enum type

849 expression must have arithmetic, enum, or pointer type

850 type of cast must be integral or enum

851 type of cast must be arithmetic, enum, or pointer

852 expression must be a pointer to a complete object type

854 a partial specialization nontype argument must be the name of a nontype parameter or a constant

855 return type is not identical to return type <type> of overridden virtual function <entity>

856 option "guiding_decls" can be used only when compiling C++

857 a partial specialization of a class template must be declared in the namespace of which it is a member

858 <entity> is a pure virtual function

859 pure virtual <entity> has no overrider

860 __declspec attributes ignored

861 invalid character in input line

862 function returns incomplete type <type>

863 effect of this "#pragma pack" directive is local to <entity>

864 <entity> is not a template

865 a friend declaration may not declare a partial specialization

866 exception specification ignored

867 declaration of "size_t" does not match the expected type <type>

868 space required between adjacent ">" delimiters of nested template argument lists (">>" is the right shift operator)

869 could not set locale <entity> to allow processing of multibyte characters

870 invalid multibyte character sequence

871 template instantiation resulted in unexpected function type of <type> (the meaning of a name may have changed since the template declaration -- the type of the template is <type>)

872 ambiguous guiding declaration -- more than one function template <entity> matches type <type>

873 non-integral operation not allowed in nontype template argument

884 pointer-to-member representation <entity> has already been set for <entity>

885 <type> cannot be used to designate constructor for <type>

886 invalid suffix on integral constant

890 variable length array with unspecified bound is not allowed

891 an explicit template argument list is not allowed on this declaration

892 an entity with linkage cannot have a type involving a variable length array

893 a variable length array cannot have static storage duration

894 <entity> is not a template

895 variable length array dimension (declared <entity>)

896 expected a template argument

902 type qualifier ignored

912 ambiguous class member reference -- <entity> used in preference to <entity>

915 a segment name has already been specified

916 cannot convert pointer to member of derived class <type> to pointer to member of base class <type> -- base class is virtual

917 invalid directory for instantiation files:

921 an instantiation information file name may not be specified when compiling several input files

923 more than one command line option matches the abbreviation "--<entity>":

925 type qualifiers on function types are ignored

926 cannot open definition list file: <entity>

928 incorrect use of va_start

929 incorrect use of va_arg

930 incorrect use of va_end

931 pending instantiations option can be used only when compiling C++

932 invalid directory for #import files:

934 a member with reference type is not allowed in a union

935 "typedef" may not be specified here

936 redeclaration of <entity> alters its access

937 a class or namespace qualified name is required

938 return type "int" omitted in declaration of function "main"

`main()` has been declared or defined with no return type.

For example:

```
main(void){
  int a;
}
```

is reported as an error by the compiler if compiled with `--strict`.

If you want it to return no result, use `void` as the return type. This is widespread in old-style C.

For ANSI C, the `--diag_suppress 938` option suppresses this warning.

For C++, this always results in an error.

See also message number `837` for more general cases.

939 `pointer-to-member representation <entity> is too restrictive for <entity>`

940 `missing return statement at end of non-void <entity>`

A return type has been defined for a function, but no value is returned.

Example:

```
int foo(int a)
{
  printf("Hello %d", a);
}
```

941 `duplicate using-declaration of <entity> ignored`

942 `enum bit-fields are always unsigned, but enum <type> includes negative enumerator`

943 `option "class_name_injection" can be used only when compiling C++`

944 `option "arg_dep_lookup" can be used only when compiling C++`

945 `option "friend_injection" can be used only when compiling C++`

946 `name following "template" must be a template`

949 `specifying a default argument on this declaration is nonstandard`

951 `return type of function "main" must be "int"`

952 `a nontype template parameter may not have class type`

953 `a default template argument cannot be specified on the declaration of a member of a class template outside of its class`

954 `a return statement is not allowed in a handler of a function try block of a constructor`

955 `ordinary and extended designators cannot be combined in an initializer designation`

956 `the second subscript must not be smaller than the first`

959 `declared size for bit field is larger than the size of the bit field type; truncated to <entity> bits`

960 `type used as constructor name does not match type <type>`

961 `use of a type with no linkage to declare a variable with linkage`

962 `use of a type with no linkage to declare a function`

963 `return type may not be specified on a constructor`

964 `return type may not be specified on a destructor`

965 `incorrectly formed universal character name`

966 `universal character name specifies an invalid character`

967 `a universal character name cannot designate a character in the basic character set`

968 `this universal character is not allowed in an identifier`

969 `the identifier __VA_ARGS__ can only appear in the replacement lists of variadic macros`

970 the qualifier on this friend declaration is ignored

971 array range designators cannot be applied to dynamic initializers

972 property name cannot appear here

975 a variable-length array type is not allowed

976 a compound literal is not allowed in an integral constant expression

977 a compound literal of type <type> is not allowed

978 a template friend declaration cannot be declared in a local class

979 ambiguous "?" operation: second operand of type <type> can be converted to third operand type <type>, and vice versa

980 call of an object of a class type without appropriate operator() or conversion functions to pointer-to-function type

982 there is more than one way an object of type <type> can be called for the argument list:

983 typedef name has already been declared (with similar type)

984 operator new and operator delete cannot be given internal linkage

985 storage class "mutable" is not allowed for anonymous unions

986 invalid precompiled header file

987 abstract class type <type> is not allowed as catch type:

988 a qualified function type cannot be used to declare a nonmember function or a static member function

989 a qualified function type cannot be used to declare a parameter

990 cannot create a pointer or reference to qualified function type

991 extra braces are nonstandard

992 invalid macro definition:

Incorrect use of -D on the compile line, for example, "-D##"

993 subtraction of pointer types <type> and <type> is nonstandard

994 an empty template parameter list is not allowed in a template template parameter declaration

995 expected "class"

996 the "class" keyword must be used when declaring a template template parameter

997 <entity> is hidden by <entity> -- virtual function override intended?

998 a qualified name is not allowed for a friend declaration that is a function definition

999 <entity> is not compatible with <entity>

1000 a storage class may not be specified here

1001 class member designated by a using-declaration must be visible in a direct base class

1006 a template template parameter cannot have the same name as one of its template parameters

1007 recursive instantiation of default argument

1009 <entity> is not an entity that can be defined

1010 destructor name must be qualified

1011 friend class name may not be introduced with "typename"

1012 a using-declaration may not name a constructor or destructor

1013 a qualified friend template declaration must refer to a specific previously declared template

1014 invalid specifier in class template declaration

1015 argument is incompatible with formal parameter

1016 prefix form of ARM function qualifier not permitted in this position

1017 Duplicate ARM function qualifiers not permitted

1018 ARM function qualifiers not permitted on this declaration/definition

ARM function qualifiers include qualifiers such as `__svc`, `__pure` and `__irq` amongst others.

See the following in the *Compiler Reference*:

- *Keywords and operators on page 5-5*.

1019 function qualifier <entity> not permitted on a non-static member function

1020 `__irq` functions must take no arguments

1021 `__irq` functions must return no result

1022 cannot have pointer nor reference to <entity> function

1023 `__global_reg` not allowed on this declaration

1024 invalid global register number; 1 to 8 allowed

An invalid register is being used in `__global_reg`.

Example:

```
__global_reg(786) int x;
```

1025 `__svc` parameter <entity> is not within permitted range (`0` to `0xffffff`) for ARM SVC instruction

SVC numbers are limited to the range 0 to `0xffffff` for the ARM compilers, and 0 to `0xFF` for the Thumb compilers.

For standard semihosting SVCs, `0x123456` is used for ARM, `0xAB` is used for Thumb.

1026 taking the address of a global register variable is not allowed

1027 `__svc_indirect` function must have arguments

1028 conflicting global register declaration with <entity>

1029 `__packed` ignored for non-pointer parameter

1030 <entity> <type> previously declared without `__packed`

1031 `Definition of <type> in packed <type> must be __packed`

The *Compiler Reference* states:

"All substructures of a packed structure must be declared using `__packed`."

The compiler faults a non-packed child structure contained in a packed parent structure. This includes the case where the substructure is an array.

For example:

```
typedef struct ChildStruct {
 int a;
 } ChildStruct;
typedef __packed struct ParentStruct {
   ChildStruct child[1];
 } ParentStruct;
```

correctly results in the message:

```
Error: #1031: Definition of "ChildStruct" in packed "ParentStruct" must be
__packed
```

See the following in the *Compiler Reference*:

- *__packed on page 5-17*.

1032 `Definition of nested anonymous <entity> in packed <type> must be __packed`

1033 `<entity> incompatible with function definition`

1034 `__irq functions must not be the target of a function call`

1038 `invalid alignment specified; only integer powers of 2 allowed`

1039 `conflicting alignment declaration with <entity>`

1040 `under-alignment not allowed`

1041 `alignment for an auto object may not be larger than 8`

For example:

```
int main(void){
  __align(16) int foo = 10;
}
```

`__align` is not permitted for a local variable `foo`, so the error is given.

See the following in the *Compiler Reference*:

- *__align on page 5-6*.

1042 `<entity> cannot be dynamically initialized when compiled position independent`

1043 `<entity> cannot be const because it contains a mutable member`

For example:

```
struct foo { int a; mutable int b; };
extern const struct foo bar;
```

When the compiler is in ROPI or RWPI mode, it disallows **`const`** objects from containing **`mutable`** members.

The reason for this restriction is that in these modes, the compiler addresses read-only data differently from read-write data. It therefore must know whether an object is in the RO or RW data section. In the following example, this restriction means that `bar` cannot contain any **`mutable`** members and is therefore in the RO data section:

```
struct foo;
extern const struct foo bar;
const struct foo *get_foo()  { return &bar; }
```

See the following in the *Compiler Reference*:

- *--apcs=qualifier...qualifier on page 3-11*.

1044 option "dep_name" can be used only when compiling C++

1045 loop in sequence of "operator->" functions starting at class <type>

1046 <entity> has no member class <entity>

1047 the global scope has no class named <entity>

1048 recursive instantiation of template default argument

1049 access declarations and using-declarations cannot appear in unions

1050 <entity> is not a class member

1051 nonstandard member constant declaration is not allowed

1053 option "parse_templates" can be used only when compiling C++

1054 option "dep_name" cannot be used with "no_parse_templates"

1055 language modes specified are incompatible

1056 invalid redeclaration of nested class

1057 type containing an unknown-size array is not allowed

1058 a variable with static storage duration cannot be defined within an inline function

1059 an entity with internal linkage cannot be referenced within an inline function with external linkage

1060 argument type <type> does not match this type-generic function macro

1062 friend declaration cannot add default arguments to previous declaration

1063 <entity> cannot be declared in this scope

1064 the reserved identifier <entity> may only be used inside a function

1065 this universal character cannot begin an identifier

1066 expected a string literal

1070 incorrect use of va_copy

1071 <entity> can only be used with floating-point types

1072 complex type is not allowed

1073 invalid designator kind

1074 floating-point value cannot be represented exactly

1075 complex floating-point operation result is out of range

1077 an initializer cannot be specified for a flexible array member

1079 standard requires that <entity> be given a type by a subsequent declaration ("int" assumed)

1080 a definition is required for inline <entity>

1081 conversion from integer to smaller pointer

1082 a floating-point type must be included in the type specifier for a _Complex or _Imaginary type

1083 Inline assembler syntax error

In ARM Compiler 4.1p2 and earlier, this error could be generated when inline assembler code contained instructions that were not supported by the inline assembler (for example, the WFI instruction). In 4.1p3 and later, including all ARM Compiler 5 versions, the following error is raised instead:

1084: This instruction not permitted in inline assembler

1084 This instruction not permitted in inline assembler

1085 Missing operand

1086 Operand is wrong type

1087 Operand should be constant

1088 Wrong number of operands

1089 Invalid PSR operand

1090 Expected PSR operand

1091 Invalid shift specified

1092 Should be acc0

1093 Must be a modifiable lvalue

1094 Expected a register expression

1095 Expected a label or function name

1096 Instruction cannot be conditional

1097 Expected a [or]

1098 Expected a shift operation

1099 Unexpected]

1100 Register specified shift not allowed

1101 Pre-Indexed addressing not allowed

1102 Post-Indexed addressing not allowed

1103 Writeback not allowed in the addressing mode

1104 Expected {

1105 Expected }

1106 Too many registers in register list

1107 Only ^ valid here

1108 Cannot mix virtual register and C/C++ expressions in register list

1109 Only virtual registers can be specified in a register range

1110 `User mode register selection/CPSR update not supported in inline assembler. Use embedded assembler or out-of-line assembler`

1111 `Expected a coprocessor name`

1112 `Expected a coprocessor register name`

These errors are given by the inline assembler if either of the following occurs:

- the coprocessor number is accidentally omitted from an `MCR` or `MRC` instruction
- an invalid coprocessor number or coprocessor register number has been given.

This is an example of correct use:

```
void foo()
{
  int reg0;
  __asm
  {
    MRC p15, 0, reg0, c1, c0, 0
  }
}
```

1114 `this feature not supported on target architecture/processor`

Example when compiled with `armcc --cpu 4T`:

```
int main(void) {
  int a,b,c;
  __asm {
  QADD a,b,c
  }
  return(a);
}
```

results in an error message because the saturated add instruction is only supported in ARMv5TE and later.

1115 `Cannot assign to const operand`

1116 `Register list cannot be empty`

1117 `Unqualified virtual function not allowed`

1118 `Expected a newline`

1119 `Reference to static variable not allowed in __asm function`

1120 `Reference to static function not allowed in __asm function`

1121 `Pointer to data member not allowed in __asm function`

1122 `__asm function cannot have static qualifier`

1123 `base class <type> is a virtual base class of <type>`

1124 `base class <type> is not virtual base class of <type>`

1125 `<entity> has no member function <entity>`

1126 `"__asm" is not allowed in this declaration`

1127 `Member initializer list not permitted for __asm constructors`

1128 `try block not permitted for __asm constructors`

1129 `Order of operands not compatible with previous compiler versions`

1130 `__align not permitted in typedef`

1131 `Non portable instruction (LDM with writeback and base in reg. list, final value of base unpredictable)`

1132 `Non portable instruction (STM with writeback and base not first in reg. list, stored value of base unpredictable)`

1133 `Expression operands not permitted with virtual base register`

1134 `literal treated as "long long"`

The constant is too large to be represented in a signed `long`, and therefore has been treated as a (signed) `long long`.

For example:

```
int foo(unsigned int bar)
{   return (bar == 2147483648);
}
```

gives a warning because 2147483648 is one greater than the maximum value permitted for a signed `long`. The `ll` suffix means that the constant is treated as a (64-bit) `long long` type rather than a signed `long`.

To eliminate the warning, explicitly add the `ll` or `LL` suffix to your constants. For example:

```
int foo(unsigned int bar)
{
  return (bar == 2147483648LL);
}
```

See the following in the *Compiler Reference*:

- *long long on page 4-14*.

1135 `literal treated as "unsigned long long"`

The constant is too large to be represented in a signed `long long`, and therefore has been given type unsigned `long long`. See error number 1134.

1137 `Expected a comma`

1138 `Unexpected comma after this expression`

1139 `MRRC operation opcode must lie in range 0-15`

1140 `MCRR operation opcode must lie in range 0-15`

1141 `CDP operation opcode must lie in range 0-15`

1142 `MRC operation opcode must lie in range 0-7`

1143 `MCR operation opcode must lie in range 0-7`

1144 `opcode_2 must lie in range 0-7`

1145 `LDC/STC extra opcode must lie in range 0-255`

1146 `LDC/STC offset must lie in range -1020 to 1020 and be word aligned`

1147 `Constant operand out of range`

1148 `floating-point operator is not permitted with --fpu=none`

1149 `floating-point return type in function definition is not permitted with -fpu=none`

1150 floating-point parameter type in function definition is not permitted with -fpu=none

1151 floating-point variable definition with initialiser is not permitted with -fpu=none

1152 polymorphic base classes need to be exported as well

1153 Cannot assign physical registers in this register list

1154 Can only specify an even-numbered physical register here

1155 Can only specify an assignment to a physical register here

1156 Can only specify an assignment from a physical register here

1157 Can only specify physical registers in a corrupted register list

1158 PSR operand not valid here

1159 Expected an unambiguous label or function name

1160 Calls to destructors for temporaries will overwrite the condition flags updated by this instruction

1161 Cannot directly modify the stack pointer SP (r13)

1162 Cannot directly modify the link register LR (r14)

1163 Cannot directly modify the program counter PC (r15)

1164 Offset must be word-aligned

1165 types cannot be declared in anonymous unions

1166 returning pointer to local variable

1167 returning pointer to local temporary

1168 option "export" can be used only when compiling C++

1169 option "export" cannot be used with "no_dep_name"

1170 option "export" cannot be used with "implicit_include"

1171 declaration of <entity> is incompatible with a declaration in another translation unit

1172 the other declaration is <entity>

1175 a field declaration cannot have a type involving a variable length array

1176 declaration of <entity> had a different meaning during compilation of <entity>

1177 expected "template"

1178 "export" cannot be used on an explicit instantiation

1179 "export" cannot be used on this declaration

1180 a member of an unnamed namespace cannot be declared "export"

1181 a template cannot be declared "export" after it has been defined

1182 a declaration cannot have a label

1183 support for exported templates is disabled

1184 cannot open exported template file: <entity>

1185 <entity> already defined during compilation of <entity>

1186 <entity> already defined in another translation unit

1188 the option to list makefile dependencies may not be specified when compiling more than one translation unit

1190 the option to generate preprocessed output may not be specified when compiling more than one translation unit

1191 a field with the same name as its class cannot be declared in a class with a user-declared constructor

1192 "implicit_include" cannot be used when compiling more than one translation unit

1193 exported template file <entity> is corrupted

1194 <entity> cannot be instantiated -- it has been explicitly specialized in the translation unit containing the exported definition

1196 the object has cv-qualifiers that are not compatible with the member <entity>

1197 no instance of <entity> matches the argument list and object (the object has cv-qualifiers that prevent a match)

1198 an attribute specifies a mode incompatible with <type>

1199 there is no type with the width specified

1200 invalid alignment value specified by attribute

1201 invalid attribute for <type>

1202 invalid attribute for <entity>

1203 invalid attribute for parameter

1204 attribute <entity> does not take arguments

1207 attribute <entity> ignored

1208 attributes may not appear here

1209 invalid argument to attribute <entity>

1210 the "packed" attribute is ignored in a typedef

1211 in "goto *expr" expr must have type "void *"

1212 "goto *expr" is nonstandard

1213 taking the address of a label is nonstandard

1214 file name specified more than once:

1215 #warning directive: <entity>

1216 attribute <entity> is only allowed in a function definition

1217 the "transparent_union" attribute only applies to unions, and <type> is not a union

1218 the "transparent_union" attribute is ignored on incomplete types

1219 <type> cannot be transparent because <entity> does not have the same size as the union

1220 <type> cannot be transparent because it has a field of type <type> which is not the same size as the union

1221 only parameters can be transparent

1222 the <entity> attribute does not apply to local variables

1224 attributes are not permitted in a function definition

1225 declarations of local labels should only appear at the start of statement expressions

1226 the second constant in a case range must be larger than the first

1227 an asm name is not permitted in a function definition

1228 an asm name is ignored in a typedef

1229 unknown register name "<entity>"

1230 modifier letter '<entity>' ignored in asm operand

1231 unknown asm constraint modifier '<entity>'

1232 unknown asm constraint letter '<entity>'

1233 asm operand has no constraint letter

1234 an asm output operand must have one of the '=' or '+' modifiers

1235 an asm input operand may not have the '=' or '+' modifiers

1236 too many operands to asm statement (maximum is 30; '+' modifier adds an implicit operand)

1237 too many colons in asm statement

1238 register "<entity>" used more than once

1239 register "<entity>" is both used and clobbered

1240 register "<entity>" clobbered more than once

1241 register "<entity>" has a fixed purpose and may not be used in an asm statement

1242 register "<entity>" has a fixed purpose and may not be clobbered in an asm statement

1243 an empty clobbers list must be omitted entirely

1244 expected an asm operand

1245 expected a register to clobber

1246 "format" attribute applied to <entity> which does not have variable arguments

1247 first substitution argument is not the first variable argument

1248 format argument index is greater than number of parameters

1249 format argument does not have string type

1250 the "template" keyword used for syntactic disambiguation may only be used within a template

1253 attribute does not apply to non-function type <type>

1254 arithmetic on pointer to void or function type

1255 storage class must be auto or register

1256 <type> would have been promoted to <type> when passed through the ellipsis parameter; use the latter type instead

1257 <entity> is not a base class member

1262 mangled name is too long

1263 Offset must be half-word aligned

1264 Offset must be double-word aligned

1265 converting to and from floating-point type is not permitted with --fpu=none

1266 Operand should be a constant expression

1267 Implicit physical register <entity> should be defined as a variable

1268 declaration aliased to unknown entity <entity>

1269 declaration does not match its alias <entity>

1270 entity declared as alias cannot have definition

1271 variable-length array field type will be treated as zero-length array field type

1272 nonstandard cast on lvalue not supported

1273 unrecognized flag name

1274 void return type cannot be qualified

1275 the auto specifier is ignored here (invalid in standard C/C++)

1276 a reduction in alignment without the "packed" attribute is ignored

1277 a member template corresponding to <entity> is declared as a template of a different kind in another translation unit

1278 excess initializers are ignored

1279 va_start should only appear in a function with an ellipsis parameter

1282 variable <entity> cannot be used in a register range

1283 A physical register name is required here

1284 A register range cannot be specified here

1285 Implicit physical register <entity> has not been defined

1286 LDRD/STRD instruction will be expanded

When LDRD and STRD instructions are used in inline assembler the compiler expands these into two LDR or STR instructions before being passed through the compiler optimization stage.

The optimization stage normally combines the two LDR or STR instruction back into a single LDRD or STRD instruction, however it is possible in some cases that a LDRD or STRD is not used.

1287 `LDM/STM instruction may be expanded`

When LDM and STM instructions are used in inline assembler the compiler expands these into a number of LDR or STR instructions before being passed through the compiler optimization stage.

The optimization stage normally combines the two LDR or STR instruction back into LDM or STM instructions, however it is possible that in some cases that a single LDM or STM instruction is not used.

1288 `Implicit ARM register <entity> was not defined due to name clash`

1289 `statement expressions are only allowed in block scope`

1291 `an asm name is ignored on a non-register automatic variable`

1292 `inline function also declared as an alias; definition ignored`

1293 `assignment in condition`

In a context where a boolean value is required (the controlling expression for if, while, for or the first operand of a conditional expression, an expression contains one of:

- A bitwise not operator (~). It is likely that a logical not operator (!) was intended.
- An assignment operator (=). This could be a mis-typed equality operator (==).

In either case if the operator is intended adding an explicit comparison against 0 might suppress the warning.

This warning can be suppressed with the `--diag_suppress 1293` option.

Example:

```
int main(void)
{
  int a,b;
  if (a=b)
}
```

1294 `Old-style function <entity>`

The compilers accept both old-style and new-style function declarations.

The difference between an old-style and a new-style function declaration is as follows.

```
// new style
int add2(int a, int b)
{
  return a+b;
}
// old style
int oldadd2(a,b)
int a;
int b;
```

```
{
  return a+b;
}
```

When compiling old style functions in C mode the compiler reports:

```
Warning: #1294-D: Old-style function oldadd2
```

1295 `Deprecated declaration <entity> - give arg types`

This warning is normally given when a declaration without argument types is encountered in ANSI C mode. In ANSI C, declarations like this are deprecated. However, it is sometimes useful to suppress this warning with the `--diag_suppress 1295` option when porting old code.

In C++:

```
void foo();
```

means:

```
void foo(void);
```

and no warning is generated.

1296 `extended constant initialiser used`

The expression used as a constant initializer might not be portable.

This warns that there is a constant initializer that does not follow the strict rules of ANSI C.

The solution is to rewrite your code to be ANSI compliant.

The following examples show code that generates this warning, and suggest potential alternatives for achieving the same goal with ANSI C compliant code.

Compiling with `--diag_suppress 1296` suppresses the warning.

Example 1:

This code generates warning 1296 when `x` and `y` are static objects, that is, global variables or static local variables, because the C standard does not permit a cast of a pointer to an integer in a constant expression:

```
int x;
int y = (int) &x;
```

ANSI C requires the initializer for a static object to be a constant expression. `(int) &x` is not considered to be a constant expression.

Be aware that addresses are not arithmetic types, so this example C code is disallowed for ANSI C. Unfortunately, this is a common ANSI non-compliancy amongst other compilers, and can result in problems when porting legacy code to ARM. This is why the ARM compiler issues a warning rather than an error.

An ANSI C compliant alternative method would be to rewrite the code so that `y` is a pointer to `x`:

```
int x;
int* y = &x;
```

Example 2:

This code, compiled with the `--c90` switch, generates warning 1296:

```
const int foo_table[] = { (int)"foo", 0, 1, 2};
```

An ANSI C compliant alternative method would be to rewrite the code as follows:

```
const char* foo_table[] = { "foo", 0, (char*)1, (char*)2};
```

Example 3:

This code generates warning 1296 because the C standard does not permit a cast of a pointer to a long integer in a constant expression:

```
char value;
long array[] = {
    (long)&value,
    (long)"string"
};
```

An ANSI C compliant alternative method would be to rewrite the code to use pointers:

```
char value;
char *array[] = {
    (char*)&value,
    (char*)"string"
};
```

This solution works because pointer-to-pointer casts are allowed in C.

1297 `Header file not guarded against multiple inclusion`

This warning is given when an unguarded header file is #included.

An unguarded header file is a header file not wrapped in a declaration such as:

```
#ifdef foo_h
#define foo_h
/* body of include file */
#endif
```

This warning is off by default. It can be enabled with:

```
--diag_warning 1297
```

1298 `Header file is guarded by '<entity>', but does not #define it`

Example:

```
#ifndef MYHEADER_H
//#define MYHEADER_H
#endif
```

To correct the code, remove the comment slashes (//). This warning is off by default. It can be enabled with:

```
--diag_warning 1298
```

1299 `members and base-classes will be initialized in declaration order, not in member initialisation list order`

1300 `<entity> inherits implicit virtual`

This warning is issued when a non-virtual member function of a derived class hides a virtual member of a parent class. For example:

```
struct Base { virtual void f(); };
struct Derived : Base { void f(); };
```

results in the message:

```
Warning: #1300-D: f inherits implicit virtual
struct Derived : Base { void f(); };
                                ^
```

Adding the `virtual` keyword in the derived class prevents the warning. For C++, the `--diag_suppress 1300` option suppresses the implicit virtual warning.

1301 `padding inserted in struct <entity>`

For the members of the structure to be correctly aligned, some padding has been inserted between members. This warning is off by default and can be enabled with `--diag_warning 1301` or `--remarks`.

For example:

```
struct X {
```

```
  char x;
  int y;
}
```

results in the message:

```
Warning: #1301-D: padding inserted in struct X
```

The compiler can also warn of padding added at the end of a `struct` or between `structs`, see 2530.

1302 `type too large to be returned in registers - __value_in_regs ignored`

1303 `using --force_new_nothrow: added "throw()"`

1304 `operator new missing exception specification`

1305 `using --force_new_nothrow: added "(::std::nothrow)"`

1307 `floating point argument not permitted with -fpu=none`

1308 `Base class <type> of __packed class <type> must be __packed`

1310 `shared block size does not match one previously specified`

1311 `bracketed expression is assumed to be a block size specification rather than an array dimension`

1312 `the block size of a shared array must be greater than zero`

1313 `multiple block sizes not allowed`

1314 `strict or relaxed requires shared`

1316 `block size specified exceeds the maximum value of <entity>`

1317 `function returning shared is not allowed`

1320 `shared type inside a struct or union is not allowed`

1321 `parameters may not have shared types`

1323 `shared variables must be static or extern`

1327 `affinity expression must have a shared type or point to a shared type`

1328 `affinity has shared type (not pointer to shared)`

1329 `shared void* types can only be compared for equality`

1331 `null (zero) character in input line ignored`

1332 `null (zero) character in string or character constant`

1333 `null (zero) character in header name`

1334 `declaration in for-initializer hides a declaration in the surrounding scope`

1335 `the hidden declaration is <entity>`

1336 `the prototype declaration of <entity> is ignored after this unprototyped redeclaration`

1338 `<entity> must have external C linkage`

1339 `variable declaration hides declaration in for-initializer`

1340 typedef <entity> may not be used in an elaborated type specifier

1341 call of zero constant ignored

1342 parameter <entity> may not be redeclared in a catch clause of function try block

1343 the initial explicit specialization of <entity> must be declared in the namespace containing the template

1345 "template" must be followed by an identifier

1347 layout qualifier cannot qualify pointer to shared

1348 layout qualifier cannot qualify an incomplete array

1349 declaration of <entity> hides handler parameter

1350 nonstandard cast to array type ignored

1351 this pragma cannot be used in a _Pragma operator (a #pragma directive must be used)

1352 field uses tail padding of a base class

1353 GNU C++ compilers may use bit field padding

1354 memory mapping conflict with precompiled header file <entity>

1355 abstract class <type> has a non-virtual destructor, calling delete on a pointer to this class is undefined behaviour

1356 an asm name is not allowed on a nonstatic member declaration

1357 static initialisation of <entity> using address of <entity> may cause link failure <option>

See error number 1359.

1358 static initialisation of extern const <entity> using address of <entity> cannot be lowered for ROPI

1359 static initialisation of <entity> using address of <entity> may cause link failure <option>

Warnings 1357 and 1359 warn against the use of non-PI code constructs and that a subsequent link step might fail.

For example, when compiled with `--apcs /ropi`:

```
char *str = "test"; /* global pointer */
```

results in the message:

```
Warning: #1357-D: static initialisation of variable "str" using address of string literal may cause link failure --ropi
```

because the global pointer `str` must be initialized to the address of the char string `test` in the `.constdata` section, but absolute addresses cannot be used in a `PI` system.

For example, when compiled with `--apcs /rwpi`:

```
int bar;
int *foo = &bar; /* global pointer */
```

results in the message:

```
Warning: #1359-D: static initialisation of variable "foo" using address of bar may cause link failure --rwpi
```

because the global pointer foo must be initialized to the address of bar in the .data section, but absolute addresses cannot be used in a PI system.

The following workarounds are possible:

- Change your code to avoid use of a global pointer. You can, for example, use a global array or local pointer instead.
- Do the initialization at run-time, for example:

  ```
  int bar;
  int *foo;
  ```

  Then write code inside a function that sets foo = &bar;. This is because when generating code as opposed to statically initializing data, the compiler has scope to work around the ROPI/RWPI constraints.

See the linker error L6248E.

1360 static initialisation of extern const <entity> using address of <entity> cannot be lowered for RWPI

For example, when compiled with --apcs /rwpi:

```
extern int y;
int* const x = &y;
int* foo()
{
  return(x);
}
```

produces a warning because prefixing y by extern prevents the compiler defining a direct address offset between the variables x and y.

1361 <entity> was declared "deprecated"

1362 unrecognized format function type <entity> ignored

1363 base class <entity> uses tail padding of base class <entity>

1366 this anonymous union/struct field is hidden by <entity>

1367 invalid error number

1368 invalid error tag

1369 expected an error number or error tag

1370 size of class is affected by tail padding

1371 labels can be referenced only in function definitions

1372 transfer of control into a statement expression is not allowed

1374 transfer of control out of a statement expression is not allowed

1375 a non-POD class definition is not allowed inside of a statement expression

1376 destructible entities are not allowed inside of a statement expression

1377 a dynamically-initialized local static variable is not allowed inside of a statement expression

1378 a variable-length array is not allowed inside of a statement expression

1379 a statement expression is not allowed inside of a default argument

1382 nonstandard conversion between pointer to function and pointer to data

1383 interface types cannot have virtual base classes

1384 interface types cannot specify "private" or "protected"

1385 interface types can only derive from other interface types

1386 <type> is an interface type

1387 interface types cannot have typedef members

1388 interface types cannot have user-declared constructors or destructors

1389 interface types cannot have user-declared member operators

1390 interface types cannot be declared in functions

1391 cannot declare interface templates

1392 interface types cannot have data members

1393 interface types cannot contain friend declarations

1394 interface types cannot have nested classes

1395 interface types cannot be nested class types

1396 interface types cannot have member templates

1397 interface types cannot have static member functions

1398 this pragma cannot be used in a __pragma operator (a #pragma directive must be used)

1399 qualifier must be base class of <type>

1400 declaration must correspond to a pure virtual member function in the indicated base class

1401 integer overflow in internal computation due to size or complexity of <type>

1402 integer overflow in internal computation

1404 potentially narrowing conversion when compiled in an environment where int, long, or pointer types are 64 bits wide

1405 current value of pragma pack is <entity>

1406 arguments for pragma pack(show) are ignored

1407 invalid alignment specifier value

1408 expected an integer literal

1409 earlier __declspec(align(...)) ignored

1410 expected an argument value for the <entity> attribute parameter

1411 invalid argument value for the <entity> attribute parameter

1412 expected a boolean value for the <entity> attribute parameter

1413 a positional argument cannot follow a named argument in an attribute

1414 attribute <filename> has no parameter named <filename>

1415 expected an argument list for the <entity> attribute

1416 expected a "," or "]"

1417 attribute argument <entity> has already been given a value

1418 a value cannot be assigned to the <entity> attribute

1419 a throw expression may not have pointer-to-incomplete type

1420 alignment-of operator applied to incomplete type

1421 <entity> may only be used as a standalone attribute

1422 <entity> attribute cannot be used here

1423 unrecognized attribute <entity>

1424 attributes are not allowed here

1425 invalid argument value for the <entity> attribute parameter

1426 too many attribute arguments

1427 conversion from inaccessible base class <type> is not allowed

1428 option "export" requires distinct template signatures

1429 string literals with different character kinds cannot be concatenated

1430 GNU layout bug not emulated because it places virtual base <entity> outside <entity> object boundaries

1431 virtual base <entity> placed outside <entity> object boundaries

1432 nonstandard qualified name in namespace member declaration

1433 reduction in alignment ignored

1434 const qualifier ignored

1436 __breakpoint argument must be an integral compile-time constant

1437 __breakpoint argument must be within 0-65535 when compiling for ARM

1438 __breakpoint argument must be within 0-255 when compiling for Thumb

1439 BKPT instruction is not supported on target architecture/processor

1440 oversize bitfield layout will change -- consider preceeding with "<entity>:0;"

1441 nonstandard cast on lvalue

The C specification states "An assignment operator shall have a modifiable lvalue as its left operand" and "a cast does not yield an lvalue".

1442 polymorphic base classes need to be exported if they are to be used for exported derivation

1443 polymorphic base classes inherited via virtual derivation need to be exported

1444 polymorphic base classes inherited via virtual derivation need all virtual functions to be exported

1446 non-POD class type passed through ellipsis

1447 a non-POD class type cannot be fetched by va_arg

The C++ ISO Specification defines that the non-required arguments of a variadic function must be of type POD (plain-old-data), such as an `int` or a `char`, but not structs or classes.

To avoid the error or warning the address of a class or struct could be given instead.

1448 `the 'u' or 'U' suffix must appear before the 'l' or 'L' suffix in a fixed-point literal`

1450 `integer operand may cause fixed-point overflow`

1451 `fixed-point constant is out of range`

1452 `fixed-point value cannot be represented exactly`

1453 `constant is too large for long long; given unsigned long long type (nonstandard)`

1454 `layout qualifier cannot qualify pointer to shared void`

1456 `a strong using-directive may only appear in a namespace scope`

1457 `<entity> declares a non-template function -- add <> to refer to a template instance`

1458 `operation may cause fixed-point overflow`

1459 `expression must have integral, enum, or fixed-point type`

1460 `expression must have integral or fixed-point type`

1461 `function declared with "noreturn" does return`

1462 `asm name ignored because it conflicts with a previous declaration`

1463 `class member typedef may not be redeclared`

1464 `taking the address of a temporary`

1465 `attributes are ignored on a class declaration that is not also a definition`

1466 `fixed-point value implicitly converted to floating-point type`

1467 `fixed-point types have no classification`

1468 `a template parameter may not have fixed-point type`

1469 `hexadecimal floating-point constants are not allowed`

1471 `floating-point value does not fit in required fixed-point type`

1472 `value cannot be converted to fixed-point value exactly`

1473 `fixed-point conversion resulted in a change of sign`

1474 `integer value does not fit in required fixed-point type`

1475 `fixed-point operation result is out of range`

1481 `fixed-point value does not fit in required floating-point type`

1482 `fixed-point value does not fit in required integer type`

1483 `value does not fit in required fixed-point type`

1485 a named-register storage class is not allowed here

1486 <entity> redeclared with incompatible named-register storage class

1487 named-register storage class cannot be specified for aliased variable

1488 named-register storage specifier is already in use

1492 invalid predefined macro entry at line <entity>: <reason>

1493 invalid macro mode name <entity>

1494 incompatible redefinition of predefined macro <entity>

1495 redeclaration of <entity> is missing a named-register storage class

1496 named register is too small for the type of the variable

1497 arrays cannot be declared with named-register storage class

1498 const_cast to enum type is nonstandard

1500 __svc parameter <entity> is not within permitted range (0 to 0xff) for Thumb SVC instruction

1501 too many arguments for __svc or __svc_indirect function

1502 arguments for __svc or __svc_indirect function must have integral type

1503 __svc_indirect function must have arguments

1504 first argument for __svc_indirect function must have integral type

1505 result of __svc or __svc_indirect function must be returned in integer registers

1506 source file <entity> has bad format

1507 error while writing <entity> file: <reason>

1508 cannot overload functions distinguished by function qualifier alone

1509 function qualifier <entity> not permitted on a virtual member function

1510 function "__attribute__((__<entity>__))" present on overridden virtual function <entity> must be present on overridding function

1511 function qualifier <entity> is not identical on overridden virtual function <entity>

1512 function qualifier <entity> present on overridden virtual function <entity> must be present on overridding function

1514 an empty initializer is invalid for an array with unspecified bound

1515 function returns incomplete class type <type>

1516 <entity> has already been initialized; the out-of-class initializer will be ignored

1517 declaration hides <entity>

1519 invalid suffix on fixed-point or floating-point constant

1522 <entity> has no corresponding member operator delete<entity> (to be called if an exception is thrown during initialization of an allocated object)

1523 a thread-local variable cannot be declared with "dllimport" or "dllexport"

1525 an initializer cannot be specified for a flexible array member whose elements have a nontrivial destructor

1526 an initializer cannot be specified for an indirect flexible array member

1528 variable attributes appearing after a parenthesized initializer are ignored

1529 the result of this cast cannot be used as an lvalue

1530 negation of an unsigned fixed-point value

1531 this operator is not allowed at this point; use parentheses

1532 flexible array member initializer must be constant

1533 register names can only be used for register variables

1534 named-register variables cannot have void type

1535 __declspec modifiers not valid for this declaration

1536 parameters cannot have link scope specifiers

1537 multiple link scope specifiers

1538 link scope specifiers can only appear on functions and variables with external linkage

1539 a redeclaration cannot weaken a link scope

1540 link scope specifier not allowed on this declaration

1541 nonstandard qualified name in global scope declaration

1542 implicit conversion of a 64-bit integral type to a smaller integral type (potential portability problem)

1543 explicit conversion of a 64-bit integral type to a smaller integral type (potential portability problem)

1544 conversion from pointer to same-sized integral type (potential portability problem)

1547 only static and extern variables can use thread-local storage

1548 multiple thread-local storage specifiers

1549 virtual <entity> was not defined (and cannot be defined elsewhere because it is a member of an unnamed namespace)

1550 carriage return character in source line outside of comment or character/string literal

1551 expression must have fixed-point type

1552 invalid use of access specifier is ignored

1553 pointer converted to bool

1554 pointer-to-member converted to bool

1555 storage specifier ignored

1556 dllexport and dllimport are ignored on class templates

1557 base class dllexport/dllimport specification differs from that of the derived class

1558 redeclaration cannot add dllexport/dllimport to <entity>

If this message is suppressed, the behavior is as though the `dllexport` or `dllimport` had been omitted. For example:

```
void f(void);
__declspec(dllimport) void f(void) { } /* suppress treats as
                                          void f(void) { } */
```

1559 dllexport/dllimport conflict with <entity>; dllexport assumed

Indicates that the compiler has seen an entity that is marked as both `dllimport` and `dllexport`. In this case, the compiler assumes that the entity is `dllexport`.

In the following example, the function definition `foo()` conflicts with the declaration `__declspec(dllimport) void foo()`. In this situation, the compiler assumes `dllexport`.

```
---test.cpp---
__declspec(dllimport) void foo();

void foo{}
{
}
------------
armcc -c test.cpp
"test.cpp", line 3: Warning: #1559-D: dllexport/dllimport conflict with
"foo" (declared at line 1); dllexport assumed

fromelf -s test.o
...
# Symbol Name Value Bind Sec Type Vis Size
=================================================================
6 _Z3foov 0x00000000 Gb 1 Code Pr 0x4
...
```

The warning message and the symbol visibility indicate that the function `foo()` is `dllexport` assumed.

1560 cannot define dllimport entity

1561 dllexport/dllimport requires external linkage

1562 a member of a class declared with dllexport/dllimport cannot itself be declared with such a specifier

1563 field of class type without a DLL interface used in a class with a DLL interface

1564 parenthesized member declaration is nonstandard

1565 white space between backslash and newline in line splice ignored

1566 dllexport/dllimport conflict with <entity>; dllimport/dllexport dropped

1567 invalid member for anonymous member class -- class <type> has a disallowed member function

1568 nonstandard reinterpret_cast

1569 positional format specifier cannot be zero

1570 a local class cannot reference a variable-length array type from an enclosing function

1571 member <entity> already has an explicit dllexport/dllimport specifier

1572 a variable-length array is not allowed in a function return type

1573 variable-length array type is not allowed in pointer to member of type <type>

1574 the result of a statement expression cannot have a type involving a variable-length array

1575 Load/Store with translation not supported in inline assembler. Use embedded assembler or out-of-line assembler

1576 Flag-setting multiply instructions not supported in inline assembler. Use embedded assembler or out-of-line assembler

1577 Flag-setting MOV/MVN instructions with constant operand not supported in inline assembler. Use embedded assembler or out-of-line assembler

1578 an asm name is ignored on an automatic variable

1593 Could not optimize: Use of unsigned index prevents optimization

1594 Could not optimize: Loop parameters must be integer for full optimization

1604 Could not optimize: Reference to this function inhibits optimization

1613 Could not optimize: Multiple store conflict

1617 Could not optimize: Loop too complex

1621 Optimization: Dead code eliminated

1624 Could not optimize: Too many overlapping conditions for efficient translation

1629 Could not optimize: Iteration count too short for array optimization

1636 Could not optimize: Complicated use of variable

1637 Unknown pragma - ignored

1638 Unable to determine last value of scalar temporary

1639 Use nolstval directive if possible

1641 Could not optimize: Too many data dependency problems

1656 Problem in pragma syntax

1661 Could not optimize: Backward transfers cannot be optimized

1662 Could not optimize: Last value of promoted scalar required

1663 Could not optimize: Branches out of the loop prevent translation

1670 Optimization: If loop converted to for loop

1676 Could not optimize: This statement prevents loop optimization

1679 Optimization: Loop vectorized

1687 Could not optimize: Reduction function suppressed - needs associative transformation

1690 Could not optimize: Unsupported data type for explicit vector operations

1691 Optimization: Loop fused with previous loop

1714 Could not optimize: Outer loop conditionally executes inner loop

1730 No indexing done along this loop

1742 Could not optimize: Feedback of array elements (equivalenced arrays)

1750 Optimization: Loop re-rolled

1759 Could not optimize: Non-unit stride interferes with vector optimization

1771 Could not optimize: Volatile items prevent analysis

1801 Optimization: Function expanded

1824 Could not optimize: Not enough vector operations to justify translation

1885 Could not optimize: Loop bounds exceed array dimensions

1861 Could not optimize: This store into array prevents optimization of outer loop

1866 Could not optimize: Non-integer subscript

1894 Optimization: Iterations peeled from loop in order to avoid dependence

1896 Optimization: Logical clause simplified

1947 Could not optimize: Cannot transform this combination of data types and operations

1978 Could not optimize: Unable to optimize user-selected loop

1979 Could not optimize: This operation inhibits loop transformation

1987 Optimization: Loop switched

1988 Optimization: Alternate code generated

1997 Optimization: Constant-length loop unrolled

2091 Optimization: Loop unrolled

2168 Optimization: Outer loop moved inside inner loop(s)

2170 Optimization: Invariant expression moved outside of outer loop

2189 Optimization: Loop unrolled and rotated

2190 Optimization: Loop unrolled and optimized

2191 Optimization: Some loads lifted to top of loop

2218 Idiom detected and optimized

2300 Might not be able to optimize: Feedback of scalar value from one loop pass to another. Conflict on line <entity>. Loop index is <entity> (<filename>,<entity>)"

2301 Might not be able to optimize: Feedback of scalar value from one loop pass to another. Conflict on line <entity>. Loop index is <entity> (<filename>)

2302 Might not be able to optimizee: Feedback of scalar value from one loop pass to another. Conflict on line <entity>. (<entity>,<filename>)

2303 Might not be able to optimize: Feedback of scalar value from one loop pass to another. Conflict on line <entity>. (<entity>)

2304 Might not be able to optimize: Potential multiple store conflict between loop iterations. Conflict on line <entity>. Loop index is <entity> (<filename>,<entity>)

2305 Might not be able to optimize: Potential multiple store conflict between loop iterations. Conflict on line <entity>. Loop index is <entity> (<filename>)

2306 Might not be able to optimize: Potential multiple store conflict between loop iterations. Conflict on line <entity>. (<entity>,<filename>)

2307 Might not be able to optimize: Potential multiple store conflict between loop iterations. Conflict on line <entity>. (<entity>)

2308 Might not be able to optimize: Potential feedback between loop iterations. Conflict on line <entity>. Loop index is <entity> (<filename>,<entity>)

2309 Might not be able to optimize: Potential feedback between loop iterations. Conflict on line <entity>. Loop index is <entity> (<filename>)

2310 Might not be able to optimize: Potential feedback between loop iterations. Conflict on line <entity>. (<entity>,<filename>)

2311 Might not be able to optimize: Potential feedback between loop iterations. Conflict on line <entity>. (<entity>)

2312 Could not optimize: Potential pointer aliasing - use restrict qualifier if ok. Conflict on line <entity>. Loop index is <entity> (<filename>,<entity>)

2313 Could not optimize: Potential pointer aliasing - use restrict qualifier if ok. Conflict on line <entity>. Loop index is <entity> (<filename>)

2314 Could not optimize: Potential pointer aliasing - use restrict qualifier if ok. Conflict on line <entity>. (<entity>,<filename>)

2315 Could not optimize: Potential pointer aliasing - use restrict qualifier if ok. Conflict on line <entity>. (<entity>)

2351 Loop nest fused with following nest(s)

2438 Could not inline: Void function used in expression

2439 Could not inline: Identifier declaration

2442 Could not inline: Cannot remove function from expression

2516 High Level Optimization halted: assembly code in routine

2519 Unable to determine constant iteration count for this loop

2524 #pragma pop with no matching #pragma push

`#pragma push` and `#pragma pop` save and restore the current pragma state.

Each pop must be paired with a push, so an error is raised for the following code:

```
#pragma push
;
#pragma pop
;
#pragma pop
```

2525 #pragma push with no matching #pragma pop

`#pragma push` and `#pragma pop` save and restore the current pragma state.

Each pop must be paired with a push.

2529 `expression must be an integral constant in range <entity> to <entity>`

2530 `padding added to end of struct <entity>`

The compiler can warn of padding added at the end of a struct or between structs. This warning is off by default and can be enabled with `--diag_warning 2530` or `--remarks`.

For example:

```
typedef struct {
  int x;
  char y;
} A;
typedef struct {
  int p;
  int q;
} B;
```

results in the message:

```
Warning: #2530-D: padding added to end of struct 'anonymous'
```

The compiler can also warn of padding inserted within a structs, see 1301.

2531 `dllimport/dllexport applied to a member of an unnamed namespace`

2533 `the <entity> attribute can only appear on functions and variables with external linkage`

2534 `strict mode is incompatible with treating namespace std as an alias for the global namespace`

2535 `in expansion of macro "<entity>" <entity>,`

2537 `in expansion of macro "<entity>" <entity><entity>`

2540 `invalid symbolic operand name <entity>`

2541 `a symbolic match constraint must refer to one of the first ten operands`

2544 `thread-local variable cannot be dynamically initialized`

2546 `some enumerator values cannot be represented by the integral type underlying the enum type`

2547 `default argument is not allowed on a friend class template declaration`

2548 `multicharacter character literal (potential portability problem)`

2549 `expected a class, struct, or union type`

2550 `second operand of offsetof must be a field`

2551 `second operand of offsetof may not be a bit field`

2552 `cannot apply offsetof to a member of a virtual base`

2553 `offsetof applied to non-POD types is nonstandard`

2554 `default arguments are not allowed on a friend declaration of a member function`

2555 `default arguments are not allowed on friend declarations that are not definitions`

2556 redeclaration of <entity> previously declared as a friend with default arguments is not allowed

2557 invalid qualifier for <type> (a derived class is not allowed here)

2558 invalid qualifier for definition of class <type>

2560 wide string literal not allowed

2565 template argument list of <entity> must match the parameter list

2566 an incomplete class type is not allowed

2567 complex integral types are not supported

2570 <entity> was declared "deprecated (<entity>)"

2571 invalid redefinition of <entity>

2574 explicit specialization of <entity> must precede its first use (<entity>)

2575 a sealed class type cannot be used as a base class

2576 duplicate class modifier

2577 a member function cannot have both the "abstract" and "sealed" modifiers

2578 a sealed member cannot be pure virtual

2579 nonvirtual function cannot be declared with "abstract" or "sealed" modifier

2580 member function declared with "override" modifier does not override a base class member

2581 cannot override sealed <entity>

2582 <entity> was declared with the class modifier "abstract"

2662 unrecognized calling convention <entity>, must be one of:

2665 attribute <entity> not allowed on parameter declarations

2666 underlying type of enum type must be an integral type other than bool

2667 some enumerator constants cannot be represented by <type>

2668 <entity> not allowed in current mode

2676 no #pragma start_map_region is currently active: pragma ignored

2677 <entity> cannot be used to name a destructor (a type name is required)

2678 nonstandard empty wide character literal treated as L'\\0'

2679 "typename" may not be specified here

2680 a non-placement operator delete must be visible in a class with a virtual destructor

2681 name linkage conflicts with previous declaration of <entity>

2682 alias creates cycle of aliased entities

2683 subscript must be constant

2684 a variable with static storage duration allocated in a specific register cannot be declared with an initializer

2685 a variable allocated in a specific register must have POD type

2686 predefined meaning of <entity> discarded

2687 declaration hides built-in <entity>

2688 declaration overloads built-in <entity>

2689 static member function not permitted here

2690 the <entity> attribute can only appear on functions and variables with internal linkage

2813 empty dependent statement in if-statement

This remark indicates that an `if` statement has no dependent statement, and is not followed by an `else` statement. For example:

```
if (x <= 0); // remark 2813 is generated here
{
    foo(x);
}
```

You can enable this remark by using `--diag_warning 2813` or `--remarks`. When using the `--remarks` option, you can suppress this remark by using `--diag_suppress 2813`.

2815 empty dependent statement in while-statement

This remark indicates that a `while` statement has no dependent statement. For example:

```
while (x != 0);
```

You can enable this remark by using `--diag_warning 2815` or `--remarks`. When using the `--remarks` option, you can suppress this remark by using `--diag_suppress 2815`.

2902 unrecognized Unicode source kind (must be one of UTF-8, UTF-16, UTF-16LE, UTF-16BE)

2903 Unicode character with hex value <entity> not representable in preprocessing output

2917 cannot open <entity> file <entity>

2918 cannot open <entity> file <entity>: <entity>

2934 conversion drops "__restrict" qualifier

2935 unable to obtain mapped memory for <entity>: <entity>

2936 array of elements containing a flexible array member is nonstandard

2938 the initialization of <entity> will be done before that of <entity>

In the C++ standard, member variables are initialized in the order they are declared in the class, not in the order they are written in the initializer list. The compiler produces this warning when the order of the initializations in the initializer list does not match the order of declarations in the class. You can enable this warning with `--diag_warning 2938` or `--remarks`.

For example:

```
class Foo {
    int x;
    char y;
public:
    Foo() : y(42), x(32) {}
};
```

results in the message:

```
Warning: #2938-D: the initialization of member "Foo::x" will be done
before that of member "Foo::y"
```

2939 `inheritance kind is not allowed in C`

2940 `inheritance kind is ignored on an enum specifier`

2941 `modifier is not allowed on an enum specifier`

2942 `modifier is ignored on an enum specifier`

2943 `identifier character cannot be represented in Unicode`

2944 `header name contains characters that cannot be represented in Unicode`

2945 `<entity> is not a valid locale name`

2946 `declaring a void parameter list with a template parameter is nonstandard`

2949 `<entity> is not a variable`

2960 `invalid template directory:`

2969 `"\" followed by white space is not a line splice`

2970 `this dynamic_cast cannot be done without runtime type information, which is disabled`

The compiler produces this error when a `dynamic_cast` must perform runtime type checking but you have disabled support for RTTI by compiling using the `--no_rtti` option. For example the following code results in this error:

```
class Foo {
    virtual int foo() = 0;
};
class Bar {};

Bar *convert(Foo *x) {
    return dynamic_cast<Bar *>(x);
}
```

2971 `conversion to <entity> is ambiguous; direct base selected`

2972 `an internal buffer would be too large`

2973 `C++ exception handler used, but exception handling semantics have not been specified`

2974 `type qualifier ignored on constructor`

2981 `explicit template arguments ignored`

2983 `<entity> is not a class type`

2984 `"delete" applied to a pointer-to-array type treated as delete[]`

The compiler produces this warning when the programmer has used the wrong kind of delete operator and the compiler has automatically corrected it. For example the following code results in this warning:

```
void f(char (*data)[10]) {
    delete data;
}
```

2985 "delete" applied to a pointer-to-array type is nonstandard; treated as delete[]

This warning has the same meaning as 2984. The compiler generates this instead of 2984 when you compile with `--strict` on the command-line.

2989 type qualifiers are ignored (underlying type is a reference)

2990 <entity>, declared using a local type, must be defined in this translation unit

2991 <entity>, declared using a type with no linkage, must be defined in this translation unit

2997 __builtin_va_arg_pack/__builtin_va_arg_pack_len can appear only in an inline function with an ellipsis parameter

2999 expected a C++ keyword

3001 offset is not constant

3002 unrecognized #pragma comment type <entity>

3006 invalid string in #pragma comment

3011 default arguments of <entity> is incompatible with a declaration in another translation unit

3012 default arguments of <entity> were different during compilation of <entity>

3014 initializer for <entity> is different in another translation unit

3015 initializer for <entity> was different during compilation of <entity>

3016 a designator into a template-dependent type is not allowed

3017 unrecognized conformance kind

3018 expected "on" or "off"

3019 #pragma conform(forScope) stack is empty

3020 no previous #pragma conform(forScope) entry matches <entity>

3021 forScope behavior is nonstandard

3022 forScope behavior is standard

3024 type qualifiers are meaningless here

3028 function call requires one argument

3029 function call requires a real floating-point argument

3033 nonstandard first parameter <entity> of "main", expected "int"

3034 nonstandard number of parameters for "main", expected zero or two parameters

3035 nonstandard second parameter <entity> of "main", expected "char *[]" or "char **"

3039 "packed" attribute ignored on class with non-POD <entity>

3040 error while deleting file <entity>: <entity>

3049 SWP instructions are deprecated in architecture ARMv6 and above

3050 FLDMX/FSTMX instructions are deprecated

3051 instruction is unpredictable in the current instruction set

3052 instruction is unpredictable with MSB < LSB

3053 instruction is unpredictable with the specified immediate value

3054 instruction is unpredictable with the specified condition

3055 instruction is unpredictable in IT block

3056 instruction is unpredictable with the specified special register

3057 instruction is unpredictable with the specified PSR mask

3058 immediate not in range <entity>

3059 immediate not a multiple of <entity>

3060 selected target does not have VFP

3061 unrecognized instruction opcode

3062 expected "<entity>"

3063 expected flag characters from "<entity>"

3064 expected special register for MSR/MRS

3065 deprecated special register name

3066 deprecated special register field specifier (use "<entity>" instead)

3067 MRS cannot select fields, use APSR, CPSR or SPSR directly

3068 expected a condition code

3069 VCVT conversion between these data types not available

3070 destination operand type or register sort incorrect

3071 source operand type or register sort incorrect

3072 data type specifiers do not match a valid encoding for this instruction

3073 missing data type specifier

3074 expected scalar operand

3075 expected data type specifier "<entity>" for destination operand

3076 expected data type specifier "<entity>" for source operand(s)

3077 writeback with no effect

3078 data type specifiers are not allowed on this instruction

3079 invalid instruction width qualifiers or data type specifiers

3080 unsupported special register

3081 expected end of line or a ";"

2.4 List of the old-style armcc error and warning messages

The following old-style error and warning messages might still be given:

C4002W `illegal unaligned load or store access - use __packed instead`

C4008W `splitting LDM/STM has no benefit`

Inappropriate use of the switch `"--split_ldm"`. This option has no significant benefit for cached systems, or for processors with a write buffer.

C4009E `unsupported CPU <entity>`

C4016W `unknown option '-<entity><entity>': ignored`

C4017W `<entity> may be used before being set`

The data flow analysis feature in the compiler is on by default.

Note

Be aware that data flow analysis is always disabled at `-O0` .

The compiler performs data flow analysis as part of its optimization process, and you can use this information to identify potential problems in the code such as variables being used before being set. However, this is really a by-product of optimization rather than a feature in its own right. The data flow analysis that detects *used before being set* only analyses hardware register use, that is, variables that are held in processor registers. It does not analyze variables or structures that are allocated on the stack, that is, stored in memory rather than in processor registers.

As code (and also register memory usage) generated by the compiler varies with the level of optimization, the warning might appear for code compiled at one level of optimization but not others. You might see it, for example, at `-O2`, but not `-O1`.

Note

The data flow analysis is not intended to be a fully complete feature. You must only treat the warnings of the form `C`*`nnnn`*`W` given by the compiler as a guide, and not rely on these warnings to identify faulty code reliably. The compiler never provides as much information as a special purpose tool such as Lint.

C4018W `division by zero: <entity>`

Constant propagation shows that a divide or remainder operator has a second operand with value 0. It is an error if execution reaches this expression. The compiler returns a result of 0 for a divide by constant 0.

C4038E `Function too large or complicated to compile (0x<num>)`

C4041U `I/O error writing '<entity>': <entity>`

C4047U `Too many errors`

C4048U `out of store while compiling with -g. Allocation size was <entity>, system size is <entity>`

C4049U `out of store. Allocation size was <entity>, system size is <entity>`

A storage allocation request by the compiler failed. Compilation of the debugging tables requested with the `-g` option might require a large amount of memory. Recompiling without `-g`, or with the program split into smaller pieces, might help.

C4050U `Compilation aborted.`

C4051E `couldn't write file '<entity>': <entity>`

C4052E `couldn't read file '<entity>': <entity>`

C4056E `bad option '<s>'`

C4057E `bad option '<s1> <s2>'`

C4065E `type of input file '<entity>' unknown`

C4066E `The code space needed for this object is too large for this version of the compiler`

Split the source file into smaller pieces.

C4075E `Can't open <entity> for output`

C4078E `stdin ('-') combined with other files`

C4079E `<entity> command with no effect`

C4301W `configuration file appears to be from a newer version of the compiler`

The configuration file is one of the XML files supplied to the compiler with the `--arm_linux_config_file` switches when using `--arm_linux_paths` or GCC command-line translation. For example:

```
armcc --arm_linux_paths --arm_linux_config_file=arm_linux_config.xml
```

This warning indicates the file is from a newer compiler so might contain unsupported features. To avoid incompatibilities, either use the newer version of the compiler that generated the configuration file, or re-generate the configuration file using your current compiler version.

See the following in the *Compiler Reference*:

- *--arm_linux_config_file=path on page 3-18*
- *--arm_linux_paths on page 3-21*.

C4302E `configuration file has an invalid version string`

This represents an error reading from or writing to an ARM Linux configuration file.

Do the following:

1. Check that the file can be read from and written to and has valid permissions.
2. Try re-generating the configuration file using `--arm_linux_configure`.

See the following in the *Compiler Reference*:

- *--arm_linux_configure on page 3-19*.

C4303E `configuration file was not specified`

See the description for error `C3302E`.

C4304E `I/O error reading configuration file <file>`

See the description for error `C3302E`.

C4305E `I/O error writing configuration file <file>`

See the description for error `C3302E`.

C4306E `could not parse configuration file <file>`

See the description for error `C3302E`.

C4307E `unable to read configuration file`

See the description for error C3302E.

C4308W `cannot find system include directory`

When using an ARM Linux mode, `--arm_linux`, `--arm_linux_paths`, or GCC command-line translation, set the `ARMCC`*`n`*`INC` environment variable to *`install_directory`*`\include`. This ensures that the compiler can find the `arm_linux` header subdirectory.

See the following in the *Compiler Reference*:

- *--arm_linux on page 3-16*
- *--arm_linux_paths on page 3-21*.

See the following in *Introducing the ARM® Compiler toolchain*:

- *Toolchain environment variables* on page 2-15.

C4309E `automatic configuration failed - cannot find GCC`

This error is produced when you try to automatically configure the tools with `--arm_linux_configure`, but GCC cannot be found. Use the `--configure_gcc=`*`path_to_gcc`* command -line option to specify the path to the GCC executable, such as `arm-none-linux-gnueabi-gcc`.

See the following in the *Compiler Reference*:

- *--arm_linux_configure on page 3-19*
- *--configure_gcc=path on page 3-43*.

C4310W `automatic configuration is incomplete - cannot determine sysroot path from GCC`

The GCC that was used for the ARM Linux configuration process did not provide a valid `sysroot` path. Use `--configure_sysroot=sysroot_path` to set the path.

See the following in the *Compiler Reference*:

- *--configure_sysroot=path on page 3-46*.

C4311E `automatic configuration failed - cannot find GLD`

This error is produced when you try to automatically configure the tools with `--arm_linux_configure`, but the GNU linker (ld) could not be found. Use the `--configure_gkd=`*`path_to_gcc`* command-line option to specify the path to the GNU ld executable, such as `arm-none-linux-gnueabi-ld`.

See the following in the *Compiler Reference*:

- *--arm_linux_configure on page 3-19*
- *--configure_gcc=path on page 3-43*.

C4312E `automatic configuration failed - could not execute GCC`

This error indicates that, when using automatic configuration for ARM Linux with `--arm_linux_configure`, the respective tools (GCC or GNU ld) could not be executed or failed when invoked. Check that they have execute permissions, and your GNU toolchain installation is working correctly.

See the following in the *Compiler Reference*:

- *--arm_linux_configure on page 3-19*.

C4313E `automatic configuration failed - could not execute GLD`

See the description of error C3312E.

C4314W `gcc command line translation - ignoring option with no translation: <option>`

C4315W gcc command line translation - translation for this command is not fully supported: <option>

C4316W option is not supported under arm linux: <option>

C4317W translated cpu or architecture option <option> is not valid

C4318W unable to read file <file>

C4319W cannot recognise type of file <file> - file will be ignored

C4320W cannot find file <file> - file will be ignored

C4321E automatic configuration failed - could not determine configuration from GCC

When configuring automatically for ARM Linux with `--arm_linux_configure`, the compiler could not determine sufficient information from GCC to produce the configuration. Try a manual configuration by specifying a `sysroot` path with `--configure_sysroot` and a path to the GNU C++ header files with `--configure_cpp_headers`.

See the following in the *Compiler Reference*:

- *--arm_linux_configure on page 3-19*
- *--configure_cpp_headers=path on page 3-39*
- *--configure_sysroot=path on page 3-46*.

C4322W could not accurately determine library configuration from GCC - configuration might be incomplete

C4323E automatic configuration failed - GCC internal specs configuration report error: <text>

C4324W could not determine libstdc++ header file path - specify this manually to ensure that C++ code will compile correctly

The path to the libstdc++ header files could not be determined from GCC. Specify this path with `--configure_cpp_headers=`*`path`*

See the following in the *Compiler Reference*:

- *--configure_cpp_headers=path on page 3-39*.

C4327W cannot determine application entry point function - using <value> as default

C4328W cannot determine library paths from GNU linker - trying to use defaults

C4329W option is missing an argument : <option>

C4330E GCC configuration is invalid

C4331W script file <file> will be treated as a scatter file

C4332E I/O error reading via file <file>

C4333E I/O error closing via file <file>

C4334W invalid GCC version in configuration file - using default

C4335E cannot retry because configuration file does not provide path to GNU executable

Displayed when the ARM Linux configuration file specified for GCC fallback does not include the correct path to `gcc`.

See the following in *Using the Compiler*:

- *Using GCC fallback when building applications on page 3-24*.

See the following in the *Compiler Reference*:

- *-Warmcc,--gcc_fallback on page 3-222*.

C4336W `compilation failed - retrying with GNU tools`

Displayed if an `armcc` compilation fails and GCC fallback is specified. `armcc` then attempts to run `gcc` to complete the compilation.

See the following in *Using the Compiler*:

- *Using GCC fallback when building applications on page 3-24*.

See the following in the *Compiler Reference*:

- *-Warmcc,--gcc_fallback on page 3-222*.

C4337E `compilation with GNU tools also failed`

Displayed if `gcc` fails during GCC fallback.

See the following in *Using the Compiler*:

- *Using GCC fallback when building applications on page 3-24*.

See the following in the *Compiler Reference*:

- *-Warmcc,--gcc_fallback on page 3-222*.

C4338W `compilation with GNU tools succeeded`

Displayed when GCC fallback succeeds.

See the following in *Using the Compiler*:

- *Using GCC fallback when building applications on page 3-24*.

See the following in the *Compiler Reference*:

- *-Warmcc,--gcc_fallback on page 3-222*.

C4339W `ambiguous translation mode options specified - using <option>`

Multiple translation mode options `--translate_gcc`, `--translate_g++`, and `--translate_gld` were specified. You must specify only one of these options to select a particular translation mode.

See the following in the *Compiler Reference*:

- *--translate_g++ on page 3-198*
- *--translate_gcc on page 3-200*
- *--translate_gld on page 3-202*.

C4340W `could not obtain license for vectorization (implied by -O3) - defaulting to -fno-tree-vectorize`

With GCC command-line translation, `-O3` implies vectorization. However, this requires a license to use the NEON vectorization feature of the compiler. Where a NEON vectorization license is not available, the compiler emits warning `C3340W` and disables vectorization.

See the following in *Introducing the ARM Compiler toolchain*:

- *Licensed features of the toolchain* on page 2-10.

See the following in the *Compiler Reference*:

- *-Onum on page 3-156*.

C4403E `__alloca_state not defined`

C4419W `dynamic stack alignment veneer inserted in <entity>`

This warning is given when compiling `__irq` functions for `--cpu=Cortex-M3-rev0` to force the stack to be 8-byte aligned on entry into the interrupt.

C4421W `write to string literal`

There is a write through a pointer that has been assigned to point at a literal string. The behavior is undefined by the ANSI standard. A subsequent read from the location written might not reflect the write.

C4435E `reference to <entity> not allowed`

C4447E `option '-E' and input file '<filename>' type conflict`

C4484E `Minimum toplevel array alignment must be 1, 2, 4 or 8`

C4486W `option '-<optionchar>' causes input file '<filename>' to be ignored`

C4487E `read from variable '<var>' with offset out of bounds`

For example :

```
void foo(void) {
  unsigned int pntr;
  pntr = (unsigned int)&pntr;
  pntr -= 4;
  pntr = *(unsigned int*)pntr;
}
```

C4488E `write to variable '<var>' with offset out of bounds`

C4489E `__vfp_status() intrinsic not supported for targets without VFP`

C4490W `instruction set switching using file extension is deprecated`

C4493E `Function alignment must be a power of 2 and greater than 1`

C4494E `invalid global register number <num>; 1 to <num> allowed`

C4497E `invalid syntax for retention constraint: <text>`

C4498E `option conflicts with an arm linux targeting option: <option>`

Certain options are expected to be used when targeting ARM Linux, for example to select the correct ABI variant options. This message is given to indicate when an incompatible option is specified.

See the following in the *Compiler Reference*:

- *--arm_linux on page 3-16*.

Chapter 3
Assembler Errors and Warnings

The error and warning messages for the assembler, `armasm`, are listed in the following topic:

- *List of the armasm error and warning messages* on page 3-2.

3.1 List of the armasm error and warning messages

The error and warning messages for armasm are:

A1017E `:INDEX: cannot be used on a pc-relative expression`

The :INDEX: expression operator has been applied to a PC-relative expression, most likely a program label. :INDEX: returns the offset from the base register in a register-relative expression.

If you require the offset of a label called <label> within an area called <areaname>, use <label> - <areaname>.

See the following in *Using the Assembler*:

- *Unary operators on page 8-21*.

A1020E `Bad predefine: <directive>`

The operand to the --predefine (-pd) command line option was not recognized. The directive must be enclosed in quotes if it contains spaces, for example on Windows:

```
--predefine "versionnum SETA 5"
```

If the SETS directive is used, the argument to the directive must also be enclosed in quotes, which might require escaping depending on the operating system and shell. For example:

```
--predefine "versionstr SETS \"5A\""
```

A1021U `No input file`

No input file was specified on the command line. This might be because there was no terminating quote on a quoted argument.

A1023E `File "<filename>" could not be opened: <reason>`

A1024E `File "<filename>" could not all be loaded: <reason>`

A1042E `Unrecognized APCS qualifier '<qualifier>'`

There is an error in the argument given to the --apcs command line option. Check the spelling of <qualifier>.

A1051E `Cannot open --depend file '<filename>': <reason>`

A1055E `Cannot open --errors file '<filename>': <reason>`

A1056E `Target cpu '<cpu>' not recognized`

The name given in the --cpu command line option is not a recognized processor name. Check the spelling of the argument.

Use --cpu=list to list the supported processors and architectures.

A1067E `Output file specified as '<filename1>', but it has already been specified as '<filename2>'`

More than one output file, -o filename, has been specified on the command line. Misspelling a command line option can cause this.

A1071E `Cannot open listing file '<filename>': <reason>`

The file given in the --list <filename> command line option could not be opened. This could be because the given name is not valid, there is no space, a read-only file with the same name already exists, or the file is in use by another process. Check that the correct path for the file is specified.

A1072E `The specified listing file '<filename>' must not be a .s or .o file`

The filename argument to the `--list` command line option has an extension that indicates it is a source or object file. This might be because the filename argument was accidentally omitted from the command line. Check that the correct argument is given to the `--list` command line option.

A1073E `The specified output file '<filename>' must not be a source file`

The object file specified on the command line has a filename extension that indicates it is a source file. This might be because the object filename was accidentally omitted from the command line.

A1074E `The specified depend file '<filename>' must not be a source file`

The filename argument to the `--depend` command line option has an extension that indicates it is a source (`.s`) file. This might be because the filename argument was accidentally omitted from the command line. Check that the correct arguments are given.

A1075E `The specified errors file '<filename>' must not be a source file`

The filename argument to the `--errors` command line option has an extension that indicates it is a source (`.s`) file. This might be because the filename argument was accidentally omitted from the command line. Check that the correct arguments are given.

A1085W `Forced user-mode LDM/STM must not be followed by use of banked R8-R14`

The ARM architecture does not permit you to access the banked registers on the instruction following a USER registers LDM or STM. The ARM Architecture Reference Manual says this form of LDM must not be followed by an instruction, which accesses banked registers (a following NOP is a good way to ensure this).

Example:

```
stmib sp, {r0-r14}^ ; Return a pointer to the frame in a1.
mov r0, sp
```

change to:

```
stmib sp, {r0-r14}^ ; Return a pointer to the frame in a1.
nop
mov r0, sp
```

A1088W `Faking declaration of area AREA |$$$$$$$|`

This is given when no `AREA` is given (see A1105E).

A1099E `Structure stack overflow max stack size <max>`

A1100E `Structure stack underflow`

A1105E `Area directive missing`

This is given when no `AREA` is given (see also A1088W).

A1106E `Missing comma`

A1107E `Bad symbol type, expect label`

A1108E `Multiply defined symbol '<name>'`

A1109E `Bad expression type`

A1110E `Expected constant expression`

A constant expression was expected after, for example, `SETA`.

See the following in *Using the Assembler*:

- *Numeric expressions on page 8-16*.

A1111E `Expected constant or address expression`

A1112E `Expected address expression`

A1113E `Expected string expression`

A string expression was expected after, for example, `SETS`.

See the following in *Using the Assembler*:

- *String expressions on page 8-14*.

A1114E `Expected register relative expression`

A1116E `String operands can only be specified for DCB`

A1117E `Register symbol '<name>' already defined`

A1118E `No current macro expansion`

A1119E `MEND not allowed within conditionals`

`MEND` means *END of Macro* (not the English word *mend*).

See the following in *Using the Assembler*:

- *Use of macros on page 5-30*.

A1120E `Bad global name`

A1121E `Global name '<name>' already exists`

A1122E `Locals not allowed outside macros`

A1123E `Bad local name`

A1125E `Unknown or wrong type of global/local symbol '<name>'`

A1126E `Bad alignment boundary, must be a multiple of 2`

A1127E `Bad IMPORT/EXTERN name`

A1128E `Common name '<sym>' already exists`

A1129E `Imported name '<sym>' already exists`

A1130E `Bad exported name`

A1131E `Bad symbol type for exported symbol '<sym>'`

A1132E `REQUIRE directive not supported for <entity> format output`

A1133E `Bad required symbol name`

A1134E `Bad required symbol type, expect (symbol is either external or label) and (symbol is relocatable and absolute)`

A1135E `Area name missing`

AREA names starting with any non-alphabetic character must be enclosed in bars, for example change:

```
AREA 1_DataArea, CODE, READONLY
```

to:

```
AREA |1_DataArea|, CODE, READONLY
```

A1136E `Entry address already set`

A1137E `Unexpected characters at end of line`

This is given when extra characters that are not part of an instruction are found on an instruction line.

For example:

```
ADD r0, r0, r1 comment
```

Can be changed to:

```
ADD r0, r0, r1 ; comment
```

A1138E `String "<string>" too short for operation, length must be > <oplength>`

A1139E `String overflow, string exceeds <max> characters`

A1140E `Bad operand type`

A1141E `Relocated expressions may only be added or subtracted`

A1142E `Subtractive relocations not supported for <entity> format output`

This can occur when subtracting symbols that are in different areas, for example:

```
IMPORT sym1
IMPORT sym2
DCD (sym2 - sym1)
```

A1143E `COMMON directive not supported for %s format output`

A1144E `DCDO directive not supported for %s format output`

A1145E `Undefined exported symbol '<sym>'`

A1146E `Unable to open output file <codeFileName>: <reason>`

A1147E `Bad shift name`

A1148E `Unknown shift name <name>, expected one of LSL, LSR, ASR, ROR, RRX`

A1150E `Bad symbol, not defined or external`

This typically occurs in the following cases:

- when the current file requires an `INCLUDE` of another file to define some symbols, for example:

  ```
  "init.s", line 2: Error: A1150E: Bad symbol
  2 00000000 DCD EBI_CSR_0
  ```

  typically requires a definitions file to be included, for example:

  ```
  INCLUDE targets/eb40.inc
  ```

- when the current file requires `IMPORT` for some symbols, for example:

  ```
  "init.s", line 4: Error: A1150E: Bad symbol
  4 00000000 LDR r0, =||Image$$RAM$$ZI$$Limit||
  ```

  typically requires the symbol to be imported, for example:

  ```
  IMPORT ||Image$$RAM$$ZI$$Limit||
  ```

A1151E `Bad register name symbol`

Example:

```
MCR p14, 3, R0, Cr1, Cr2
```

The coprocessor registers `CR` must be labelled as a lowercase `c` for the code to build. The ARM register can be `r` or `R`:

```
MCR p14, 3, r0, c1, c2
```

or

```
MCR p14, 3, R0, c1, c2
```

A1152E `Unexpected operator`

A1153E `Undefined symbol`

A1154E `Unexpected operand, operator expected`

A1155E `Unexpected unary operator equal to or equivalent to <operator>`

A1156E `Missing open bracket`

A1157E `Syntax error following directive`

A1158E `Illegal line start, should be blank`

Some directives, for example, `ENTRY`, `IMPORT`, `EXPORT`, and `GET` must be on a line without a label at the start of the line. This error is given if a label is present.

A1159E `Label missing from line start`

Some directives, for example, `FUNCTION` or `SETS`, require a label at the start of the line, for example:

```
my_func FUNCTION
```

or

```
label SETS
```

This error is given if the label is missing.

A1160E `Bad local label number`

A numeric local label is a number in the range 0-99, optionally followed by a name.

See the following in *Using the Assembler*:

- *Numeric local labels on page 8-12*.

A1161E `Syntax error following local label definition`

A1162E `Incorrect routine name '<name>'`

A1163E `Unknown opcode <name> , expecting opcode or Macro`

The most common reasons for this are:

- Forgetting to put some white space on the left hand side margin, before the instruction, for example change:

  ```
  MOV PC,LR
  ```

  to

  ```
   MOV PC,LR
  ```

- Use of a hardware floating point instruction without using the `--fpu switch`, for example:

  ```
   FMXR FPEXC, r1 ;
  ```

  must be assembled with armasm `--fpu vfp`

- Mis-typing the opcode:

  ```
   ADDD
  ```

  instead of

  ```
   ADD
  ```

A1164E `Opcode not supported on selected processor`

The processor selected on the armasm command line does not support this instruction. See the *ARM Architecture Reference Manual*, http://infocenter.arm.com/help/topic/com.arm.doc.set.architecture.

A1165E Too many actual parameters, expecting <actual> parameters

A1166E Syntax error following label

A1167E Invalid line start

A1168E Translate not allowed in pre-indexed form

A1169E Missing close square bracket

A1170E Immediate 0x<adr> out of range for this operation, must be below (0x<adr>)

This error is given when DCB, DCW or DCWU directives are used with immediates that are too large.

See the following in the *Assembler Reference*:

- *DCB on page 7-23*
- *DCW and DCWU on page 7-30*.

A1171E Missing close bracket

A1172E Bad rotator <rotator>, must be even and between 0 and 30

A1173E ADR/L cannot be used on external symbols

The ADR and ADRL pseudo-instructions can only be used with labels within the same code area. To load an out-of-area address into a register, use LDR instead.

A1174E Data transfer offset 0x<val> out of range. Permitted values are 0x<mini> to 0x<maxi>

A1175E Bad register range

A1176E Branch offset 0x<val> out of range. Permitted values are 0x<mini> to 0x<maxi>

Branches are PC relative, and have a limited range. If you are using numeric local labels, you can use the ROUT directive to limit their scope. This helps to avoid referring to the wrong label by accident.

See the following in *Using the Assembler*:

- *Numeric local labels on page 8-12*.

A1179E Bad hexadecimal number

A1180E Missing close quote

A1181E Bad operator

A1182E Bad based <base> number

A1183E Numeric overflow

A1184E Externals not valid in expressions

A1185E Symbol missing

A1186E Code generated in data area

An instruction has been assembled into a data area. This can happen if you have omitted the CODE attribute on the AREA directive.

See the following in the *Assembler Reference*:

- *AREA on page 7-13*.

A1187E `Error in macro parameters`

A1188E `Register value <val> out of range.  Permitted values are <mini> to <maxi>`

A1189E `Missing '#'`

A1190E `Unexpected '<entity>'`

A1191E `Floating point register number out of range 0 to <maxi>`

A1192E `Coprocessor register number out of range 0 to 15`

A1193E `Coprocessor number out of range 0 to 15`

A1194E `Bad floating-point number`

A1195W `Small floating point value converted to 0.0`

A1196E `Too late to ban floating point`

A1198E `Unknown operand`

This can occur when an operand is accidentally miss-typed.

For example:

```
armasm init.s -g -PD "ROM_RAM_REMAP SETL {FALS}"
```

must be:

```
armasm init.s -g -PD "ROM_RAM_REMAP SETL {FALSE}"
```

See the following in *Using the Assembler*:

- *Assembly time substitution of variables on page 8-6*.

A1199E `Coprocessor operation out of range 0 to <maxi>`

A1200E `Structure mismatch expect While/Wend`

A1201E `Substituted line too long, maximum length <max>`

A1202E `No pre-declaration of substituted symbol '<name>'`

See the following in *Using the Assembler*:

- *Assembly time substitution of variables on page 8-6*.

A1203E `Illegal label parameter start in macro prototype`

A1204E `Bad macro parameter default value`

A1205E `Register <reg> occurs multiply in list`

A1206E `Registers should be listed in increasing register number order`

This warning is given if registers in, for example, `LDM` or `STM` instructions are not specified in increasing order and the `--checkreglist` option is used.

A1207E `Bad or unknown attribute`

This error is given when an invalid attribute is given in the `AREA` directive. For example:

```
AREA test,CODE,READONLY,HALFWORD
```

`HALFWORD` is invalid, so remove it.

See the following in the *Assembler Reference*:

- *AREA on page 7-13*.

A1209E ADRL cannot be used with PC as destination

A1210E Non-zero data within uninitialized area '<name>'

A1211E Missing open square bracket

A1212E Division by zero

A1213E Attribute <entity> cannot be used with attribute <entity>

A1214E Too late to define symbol '<sym>' as register list

A1215E Bad register list symbol

A1216E Bad string escape sequence

A1217E Error writing to code file <codeFileName>: <reason>

A1219E Bad APSR, CPSR or SPSR designator

For example:

```
MRS r0, PSR
```

It is necessary to specify which status register to use (CPSR or SPSR), such as, for example:

```
MRS r0, CPSR
```

A1220E BLX <address> must be unconditional

A1221E Area attribute '<entity>' not supported for <entity> object file format

A1223E Comdat Symbol '<name>' is not defined

A1224E <entity> format does not allow PC-relative data transfers between areas

A1225E ASSOC attribute is not allowed in non-comdat areas

A1226E SELECTION attribute is not allowed in non-comdat areas

A1227E Comdat Associated area '<name>' undefined at this point in the file

A1228E Comdat Associated area '<name>' is not an area name

A1229E Missing COMDAT symbol

A1230E Missing '}' after COMDAT symbol

A1234E Undefined or Unexported Weak Alias for symbol '<sym>'

A1237E Invalid register or register combination for this operation

A1238E Immediate value must be word aligned when used in this operation

A1240E Immediate value cannot be used with this operation

A1241E Must have immediate value with this operation

A1242E Offset must be word aligned when used with this operation

A1243E Offset must be halfword aligned with this operation

A1244E Missing '!'

A1245E B or BL from Thumb code to ARM code

A1247E BLX from ARM code to ARM code, use BL

This occurs when there is a BLX *label* branch from ARM code to ARM code within this assembler file. This is not permitted because BLX *label* always results in a change of instruction set state. The usual solution is to use BL instead.

A1248E BLX from Thumb code to Thumb code, use BL

This occurs when there is a BLX *label* branch from Thumb code to Thumb code within this assembler file. This is not permitted because BLX *label* always results in a change of instruction set state. The usual solution is to use BL instead.

A1249E Post indexed addressing mode not available

A1250E Pre indexed addressing mode not available for this instruction, use [Rn, Rm]

A1253E Thumb branch to external symbol cannot be relocated: not representable in <fmt>

A1254E Halfword literal values not supported

Example:

```
LDRH R3, =constant
```

Change the LDRH into LDR, which is the standard way of loading constants into registers.

A1256E DATA directive can only be used in CODE areas

A1259E Invalid PSR field specifier, syntax is <PSR>_ where <PSR> is either CPSR or SPSR

A1260E PSR field '<entity>' specified more than once

A1261E MRS cannot select fields, use APSR, CPSR or SPSR directly

This is caused by an attempt to use fields for CPSR or SPSR with an MRS instruction, such as:

```
MRS r0, CPSR_c
```

A1262U Expression storage allocator failed

A1265U Structure mismatch: IF or WHILE unmatched at end of INCLUDE file

A1267E Bad GET or INCLUDE for file <filename>

A1268E Unmatched conditional or macro

A1269U unexpected GET on structure stack

A1270E File "<entity>" not found

A1271E Line too long, maximum line length is <MaxLineLength>

A1272E End of input file

A1273E '\\' should not be used to split strings

A1274W '\\' at end of comment

A1283E Literal pool too distant, use LTORG to assemble it within 1KB

For Thumb code, the literal pool must be within 1KB of the LDR instruction to access it. See A1284E and A1471W.

A1284E Literal pool too distant, use LTORG to assemble it within 4KB

For ARM code, the literal pool must be within 4KB of the LDR instruction to access it. To solve this, add an LTORG directive into your assembler source file at a convenient place.

See the following in *Using the Assembler*:

- *Load addresses to a register using LDR Rd, =label on page 5-17*.

See the following in the *Assembler Reference*:

- *LTORG on page 7-63*.

A1285E Bad macro name

A1286E Macro already exists

A1287E Illegal parameter start in macro prototype

A1288E Illegal parameter in macro prototype

A1289E Invalid parameter separator in macro prototype

A1290E Macro definition too big, maximum length <max>

A1291E Macro definitions cannot be nested

A1310W Symbol attribute not recognized

A1311U macro definition attempted within expansion

A1312E Assertion failed

A1313W Missing END directive at end of file

The assembler requires an END directive to know when the code in the file terminates. You can add comments or other such information in free format after this directive.

A1314W Reserved instruction (using NV condition)

A1315E NV condition not supported on targeted CPU

A1316E Shifted register operand to MSR has undefined effect

A1319E Undefined effect (using PC as Rs)

A1320E Undefined effect (using PC as Rn or Rm in register specified shift)

A1321E Undefined effect (using PC as offset register)

A1322E Unaligned transfer of PC, destination address must be 4 byte aligned, otherwise result is UNPREDICTABLE

This error is reported when you try to use an LDR instruction to load the PC from a non word-aligned address. According to the ARM Architecture Reference Manual, this gives an UNPREDICTABLE result. For example:

```
AREA Example, CODE
LDR pc, [pc, #6] ; Error - offset must be a multiple of 4
END
```

A1323E Reserved instruction (Rm = Rn with post-indexing)

A1324E Undefined effect (PC + writeback)

A1327E Non portable instruction (LDM with writeback and base in register list, final value of base unpredictable)

LDM Operand restriction:

- If the base register <Rn> is specified in <registers>, and base register writeback is specified, the final value of <Rn> is UNKNOWN.

A1328E `Non portable instruction (STM with writeback and base not first in register list, stored value of base unpredictable)`

STM Operand restrictions if <Rn> is specified as <registers> and base register writeback is specified:

- If <Rn> is the lowest-numbered register specified in <register_list>, the original value of <Rn> is stored.
- Otherwise, the stored value of <Rn> is UNKNOWN.

A1329E `Unpredictable instruction (forced user mode transfer with write-back to base)`

This is caused by an instruction such as `PUSH {r0}^` where the ^ indicates access to user registers. The *ARM Architectural Reference Manual* specifies that writeback to the base register is not available with this instruction.

Instead, the base register must be updated separately. For example:

```
SUB sp, sp,#4
STMID sp, {r0}^
```

Another example is replacing `STMFD R0!, {r13, r14}^` with:

```
SUB r0, r0,#8
STM r0, {r13, r14}^
```

See also A1085W

A1331E `Unpredictable instruction (PC as source or destination)`

A1332E `Unpredictable effect (PC-relative SWP)`

A1334E `Undefined effect (use of PC/PSR)`

A1335E `Useless instruction (PC cannot be written back)`

A1337E `Useless instruction (PC is destination)`

A1338E `Dubious instruction (PC used as an operand)`

A1339E `Unpredictable if RdLo and RdHi are the same register`

A1341E `Branch to unaligned destination, expect destination to be <max> byte aligned`

A1342W `<name> of symbol in another AREA will cause link-time failure if symbol is not close enough to this instruction`

A1344I `host error: out of memory`

A1355U `A Label was found which was in no AREA`

Example:

This can occur where no white-space precedes an assembler directive.

Assembler directives must be indented with white-space, for example use:

```
IF :DEF: FOO
; code
ENDIF
```

instead of:

```
IF :DEF: FOO
 ; code
ENDIF
```

Symbols in the left-hand column are assumed to be labels.

A1356E Instruction not supported on targeted CPU

This occurs if you try to use an instruction that is not supported by the default architecture or processor for armasm.

For example:

```
 SMULBB r0,r0,r1 ;
```

can be assembled with:

```
armasm --cpu 5TE
```

The processor selected on the armasm command line does not support this instruction. See the *ARM Architecture Reference Manual*, http://infocenter.arm.com/help/topic/com.arm.doc.set.architecture.

A1406E Bad decimal number

A1407E Overlarge floating point value

A1408E Overlarge (single precision) floating point value

A1409W Small (single precision) floating value converted to 0.0

A1411E Closing '>' missing from vector specifier

A1412E Bad vector length, should be between <min> and <max>

A1413E Bad vector stride, should be between <min> and <max>

A1414E Vector wraps round over itself, length * stride should not be greater than <max>

A1415E VFPASSERT must be followed by 'VECTOR' or 'SCALAR'

A1416E Vector length does not match current vector length <len>

A1417E Vector stride does not match current vector stride

A1418E Register has incorrect type '<type>' for instruction, expect floating point/double register type

A1419E Scalar operand not in a scalar bank

A1420E Lengths of vector operands are different

A1421E Strides of vector operands are different

A1422E This combination of vector and scalar operands is not allowed

A1423E This operation is not vectorizable

A1424E Vector specifiers not allowed in operands to this instruction

A1425E Destination vector must not be in a scalar bank

A1426E Source vector must not be in a scalar bank

A1427E Operands have a partial overlap

A1428E Register list contains registers of varying types

A1429E Expected register list

The assembler reports this when FRAME SAVE and FRAME RESTORE directives are not given register lists.

See the following in the *Assembler Reference*:

- *FRAME RESTORE on page 7-41*
- *FRAME SAVE on page 7-43*.

A1430E Unknown frame directive

A1431E Frame directives are not accepted outside of PROCs/FUNCTIONs

See the following in *Using the Assembler*:

- *Frame directives on page 5-37*.

A1432E Floating-point register type not consistent with selected floating-point architecture

A1433E Only the writeback form of this instruction exists

The addressing mode specified for the instruction did not include the writeback specifier (that is, a '!' after the base register), but the instruction set only supports the writeback form of the instruction. Either use the writeback form, or replace with instructions that have the required behavior.

A1435E {PCSTOREOFFSET} is not defined when assembling for an architecture

{PCSTOREOFFSET} is only defined when assembling for a processor, not for an architecture.

A1437E {ARCHITECTURE} is undefined

{ARCHITECTURE} is only defined when assembling for an architecture, not for a processor.

A1446E Bad or unknown attribute '<attr>'. Use --apcs /interwork instead

Example:

```
AREA test1, CODE, READONLY
AREA test, CODE, READONLY, INTERWORK
```

This code might have originally been intended to work with SDT. The INTERWORK area attribute is now obsolete. To eliminate the warning:

- remove the ", INTERWORK" from the AREA line.
- assemble with 'armasm --apcs /interwork foo.s' instead

A1447W Missing END directive at end of file, but found a label named END

This is caused by the END directive not being indented.

A1448E Deprecated form of PSR field specifier used (use _f)

A1449E Deprecated form of PSR field specifier used (use _c)

A1450E Deprecated form of PSR field specifier used (use _cxsf for future compatibility)

The assembler, armasm, supports the full range of MRS and MSR instructions, in the form:

```
MRS(cond) Rd, CPSR
MRS(cond) Rd, SPSR
MSR(cond) CPSR_fields, Rm
MSR(cond) SPSR_fields, Rm
MSR(cond) CPSR_fields, #immediate
MSR(cond) SPSR_fields, #immediate
```

where `fields` can be any combination of `cxsf`.

Earlier releases of the assembler permitted other forms of the MSR instruction to modify the control field and flags field:

- cpsr or cpsr_all, control and flags field
- cpsr_flg, flags field only
- cpsr_ctl, control field only.

Similar control and flag settings apply for SPSR.

These forms are now deprecated and must not be used. If your legacy code contains them, the assembler reports:

```
Deprecated form of PSR field specifier used (use _cxsf)
```

To avoid the warning, in most cases you can modify your code to use `_c`, `_f`, `_cf` or `_cxsf` instead.

See the following in *Using the Assembler*:

- *Conditional execution in ARM state on page 6-3*
- *Conditional execution in Thumb state on page 6-4*
- *General-purpose registers on page 3-11*
- *Access to the inline barrel shifter on page 3-25*.

See also the FAQ *armasm: use of MRS and MSR instructions ('Deprecated form of PSR field specifier')*, `http://infocenter.arm.com/help/topic/com.arm.doc.faqs/ka3724.html`.

A1454E `FRAME STATE RESTORE directive without a corresponding FRAME STATE REMEMBER`

See the following in *Using the Assembler*:

- *Frame directives on page 5-37*.

See the following in the *Assembler Reference*:

- *FRAME STATE REMEMBER on page 7-44*
- *FRAME STATE RESTORE on page 7-45*.

A1456W `INTERWORK area directive is obsolete. Continuing as if --apcs /inter selected`

Example:

```
AREA test, CODE, READONLY, INTERWORK
```

This code might have originally been intended to work with SDT. The `INTERWORK` area attribute is now obsolete. To eliminate the warning:

1. Remove the `", INTERWORK"` from the `AREA` line.
2. Assemble with `armasm --apcs /interwork foo.s` instead.

A1457E `Cannot mix INTERWORK and NOINTERWORK code areas in same file`

`INTERWORK` and (default) `NOINTERWORK` code areas cannot be mixed in the same file. This code might have originally been intended to work with SDT. The `INTERWORK` area attribute is obsolete in the ARM Compiler toolchain.

Example:

```
AREA test1, CODE, READONLY
  …
AREA test2, CODE, READONLY, INTERWORK
```

To eliminate the error:

1. move the two AREAs into separate assembler files such as, for example, `test1.s` and `test2.s`
2. remove the `", INTERWORK"` from the AREA line in `test2.s`

3. assemble `test1.s` with `armasm --apcs /nointerwork`
4. assemble `test2.s` with `armasm --apcs /interwork`
5. at link time, the linker adds any necessary interworking veneers.

A1458E `DCFD or DCFDU not allowed when fpu is None`

A1459E `Cannot B or BL to a register`

This form of the instruction is not permitted. See the *ARM Architecture Reference Manual* for the permitted forms.

A1461E `Specified processor or architecture does not support Thumb instructions`

It is likely that you are specifying a specific architecture or cpu using the `--cpu` option and then incorporating some Thumb code in the AREA that is generating this error.

For example:

```
armasm --cpu 4 code.s
```

StrongARM is an architecture 4 (not 4T) processor and does not support Thumb code.

A1462E `Specified memory attributes do not support this instruction`

A1463E `SPACE directive too big to fit in area, area size limit 2^32`

A1464W `ENDP/ENDFUNC without corresponding PROC/FUNC`

A1466W `Operator precedence means that expression would evaluate differently in C`

`armasm` has always evaluated certain expressions in a different order to C. This warning might help C programmers from being caught out when writing in assembler.

To avoid the warning, either:

- modify the code to make the evaluation order explicit (that is, add more brackets)
- suppress the warning with `--unsafe` switch.

See the following in *Using the Assembler*:

- *Operator precedence on page 8-29*.

A1467W `FRAME ADDRESS with negative offset <offset> is not recommended`

A1468W `FRAME SAVE saving registers above the canonical frame address is not recommended`

A1469E `FRAME STATE REMEMBER directive without a corresponding FRAME STATE RESTORE`

See the following in *Using the Assembler*:

- *Frame directives on page 5-37*.

See the following in the *Assembler Reference*:

- *FRAME STATE REMEMBER on page 7-44*
- *FRAME STATE RESTORE on page 7-45*.

A1471W `Directive <directive> may be in an executable position`

This can occur with, for example, the LTORG directive (see A1283E & A1284E). LTORG instructs the assembler to dump literal pool DCD data at this position.

To prevent this warning from occurring, the data must be placed where the processor cannot execute them as instructions. A good place for an LTORG is immediately after an unconditional branch, or after the return instruction at the end of a subroutine.

As a last resort, you could add a branch over the LTORG, to avoid the data being executed, for example:

```
 B unique_label
 LTORG
unique_label
```

A1475E `At least one register must be transferred, otherwise result is UNPREDICTABLE`

A1476E `BX r15 at non word-aligned address is UNPREDICTABLE`

A1477E `This register combination results in UNPREDICTABLE behavior`

This error is generated when you are assembling an instruction that has UNPREDICTABLE results on execution. You must rewrite your code to avoid this UNPREDICTABLE behaviour. For example, the following instructions all cause this error when assembling to Thumb, and the target architecture is ARMv6T2 or later:

```
ADD sp, r0, #100 ; error - UNPREDICTABLE use of SP
CMP pc, #1 ; error - UNPREDICTABLE use of PC
PUSH    {r0, pc} ; error - use of an UNPREDICTABLE register combination
```

A1479W `Requested alignment <alignreq> is greater than area alignment <align>, which has been increased`

This is warning about an `ALIGN` directive that has a coarser alignment boundary than its containing `AREA`. This is not permitted. To compensate, the assembler automatically increases the alignment of the containing `AREA` for you. A simple test case that gives the warning is:

```
 AREA test, CODE, ALIGN=3
 ALIGN 16
 mov pc, lr
 END
```

In this example, the alignment of the `AREA (ALIGN=3)` is 2^3=8 byte boundary, but the `mov pc,lr` instruction is on a 16-byte boundary, causing the error.

Note

The two alignment types are specified in different ways.

This warning can also occur when using `AREA ... ALIGN=0` to align a code section on a byte boundary. This is not possible. Code sections can only be aligned on:

- a four-byte boundary for ARM code, so use `"ALIGN=2"`
- a two-byte boundary for Thumb code, so use `"ALIGN=1"`.

See the following in the *Assembler Reference*:

- *ALIGN on page 7-11*
- *AREA on page 7-13*.

A1480W `Macro cannot have same name as a directive or instruction`

A1482E `Shift option out of range, allowable values are from <min> to <max>`

A1484W `Obsolete shift name 'ASL', use LSL instead`

The ARM architecture does not have an ASL shift operation. The ARM barrel shifter only has the following shift types: ROR, ASR, LSR, and LSL.

An arithmetic (that is, signed) shift left is the same as a logical shift left, because the sign bit always gets shifted out.

In ADS 1.0.1 and earlier, the assembler silently converted ASL to LSL. Use the `--unsafe` switch to downgrade this error to a warning.

See the following in the *Assembler Reference*:

- *--unsafe on page 2-82*
- *ASR on page 3-46*.

A1485E `LDM/STM instruction exceeds maximum register count <max> allowed with --split_ldm`

A1486E `ADR/ADRL of a symbol in another AREA is not supported in ELF`

The ADR and ADRL pseudo-instructions can only be used with labels within the same code section. To load an out-of-area address into a register, use LDR instead.

A1487W `Obsolete instruction name 'ASL', use LSL instead`

The Thumb instruction ASL is now faulted. See the corresponding ARM ASL message A1484W.

A1488W `PROC/FUNC at line <lineno> in '<filename>' without matching ENDP/ENDFUNC`

A1489E `<FPU> is undefined`

A1490E `<CPU> is undefined`

{CPU} is only defined by assembling for a processor and not an architecture.

A1491W `Internal error: Found relocation at offset <offset> with incorrect alignment`

This might indicate an assembler fault. Contact your supplier.

A1492E `Immediate 0x<val> out of range for this operation. Permitted values are 0x<mini> to 0x<maxi>`

A1493E `REQUIRE must be in an AREA`

A1495W `Target of branch is a data address`

armasm determines the type of a symbol and detects branches to data. Specify `--diag-suppress 1495` to suppress this warning.

A1496W `Absolute relocation of ROPI address with respect to symbol '<symbol>' at offset <offset> may cause link failure`

For example, when assembling with `--apcs /ropi`:

```
  AREA code, CODE
codeaddr DCD codeaddr
  END
```

because this generates an absolute relocation (`R_ARM_ABS32`) to a PI code symbol.

A1497W `Absolute relocation of RWPI address with respect to symbol '<symbol>' at offset <offset> may cause link failure`

For example, when assembling with `--apcs /rwpi`:

```
  AREA data, DATA
dataaddr DCD dataaddr
  END
```

because this generates an absolute relocation (`R_ARM_ABS32`) to a PI data symbol.

A1498E Unexpected characters following Thumb instruction

For example:

```
ADD r0, r0, r1
```

is accepted as a valid instruction, for both ARM and Thumb, but:

```
ADD r0, r0, r1, ASR #1
```

is a valid instruction for ARM, but not for Thumb, so the unexpected characters are ", ASR #1".

A1499E Register pair is not a valid contiguous pair

A1500E Unexpected characters when expecting '<eword>'

A1501E Shift option out of range, allowable values are 0, 8, 16 or 24

A1502W Register <reg> is a caller-save register, not valid for this operation

A1505E Bad expression type, expect logical expression

A1506E Accumulator should be in form accx where x ranges from 0 to <max>

A1507E Second parameter of register list must be greater than or equal to the first

A1508E Structure mismatch expect Conditional

A1509E Bad symbol type, expect label, or weak external symbol

A1510E Immediate 0x<imm> cannot be represented by 0-255 and a rotation

A1511E Immediate cannot be represented by combination of two data processing instructions

A1512E Immediate 0x<val> out of range for this operation. Permitted values are <mini> to <maxi>

A1513E Symbol not found or incompatible Symbol type for '<name>'

A1514E Bad global name '<name>'

A1515E Bad local name '<name>'

A1516E Bad symbol '<name>', not defined or external

A1517E Unexpected operator equal to or equivalent to <operator>

A1539E Link Order dependency '<name>' not an area

A1540E Cannot have a link order dependency on self

A1541E <code> is not a valid condition code

A1542E Macro names <name1> and <name2>[parameter] conflict

A1543W Empty macro parameter default value

A1544E Invalid empty PSR field specifier, field must contain at least one of c,x,s,f

A1545U Too many sections for one <objfmt> file

A1546W Stack pointer update potentially breaks 8 byte stack alignment

Example:

```
PUSH {r0}
```

The stack must be eight-byte aligned on an external boundary so pushing an odd number of registers causes this warning to be given. This warning is suppressed by default. To enable this warning use `--diag_warning 1546`.

See the following in the *Assembler Reference*:

- *--diag_warning=tag{, tag} on page 2-29*.

A1547W `PRESERVE8 directive has automatically been set`

Example:

```
PUSH {r0,r1}
```

This warning is given because you have not explicitly set the `PRESERVE8` directive, but the assembler has set it automatically. This warning is suppressed by default. To enable this warning use `--diag_warning 1547`.

See the following in the *Assembler Reference*:

- *--diag_warning=tag{, tag} on page 2-29*
- *REQUIRE8 and PRESERVE8 on page 7-76*.

A1548W `Code contains LDRD/STRD indexed/offset from SP but REQUIRE8 is not set`

Example:

```
PRESERVE8
STRD r0,[sp,#8]
```

This warning is given when the `REQUIRE8` directive is not set when required.

See the following in the *Assembler Reference*:

- *REQUIRE8 and PRESERVE8 on page 7-76*.

A1549W `Setting of REQUIRE8 but not PRESERVE8 is unusual`

Example:

```
PRESERVE8 {FALSE}
REQUIRE8
STRD r0,[sp,#8]
```

A1550U `Input and output filenames are the same`

A1551E `Cannot add Comdef area <name> to non-comdat group`

A1560E `Non-constant byte literal values not supported`

A1561E `MERGE and STRING sections must be data sections`

A1562E `Entry size for Merge section must be greater than 0`

A1563W `Instruction stalls CPU for <stalls> cycle(s)`

The assembler can give information about possible interlocks in your code caused by the pipeline of the processor chosen by the `--cpu` option. To do this assemble with `armasm --diag_warning 1563`

Note

If the `--cpu` option specifies a multi-issue processor such as Cortex-A8, the interlock warnings are unreliable.

See also warning `A1746W`.

A1572E `Operator SB_OFFSET_11_0 only allowed on LDR/STR instructions`

A1573E `Operator SB_OFFSET_19_12 only allowed on Data Processing instructions`

A1574E Expected one or more flag characters from "<str>"

A1575E BLX with bit[0] equal to 1 is architecturally UNDEFINED

A1576E Bad coprocessor register name symbol

A1577E Bad coprocessor name symbol

A1578E Bad floating point register name symbol '<sym>'

A1581W Added <no_padbytes> bytes of padding at address <address>

The assembler warns by default when padding bytes are added to the generated code. This occurs whenever an instruction/directive is used at an address that requires a higher alignment, for example, to ensure ARM instructions start on a four-byte boundary after some Thumb instructions, or where there is a DCB followed by DCD.

For example:

```
 AREA Test, CODE, READONLY
 THUMB
ThumbCode
  MOVS r0, #1
  ADR r1, ARMProg
  BX r1
; ALIGN ; <<< add to avoid the first warning
 ARM
ARMProg
 ADD r0,r0,#1
 BX LR
 DCB 0xFF
 DCD 0x1234
 END
```

Results in the warnings:

```
A1581W: Added 2 bytes of padding at address 0x6
8 00000008 ARM
A1581W: Added 3 bytes of padding at address 0x11
13 00000014 DCD 0x1234
```

The warning can also occur when using ADR in Thumb-only code. The ADR Thumb pseudo-instruction can only load addresses that are word aligned, but a label within Thumb code might not be word aligned. Use ALIGN to ensure four-byte alignment of an address within Thumb code.

See the following in the *Assembler Reference*:

- *ADR (PC-relative) on page 3-38*
- *ADR (register-relative) on page 3-40*
- *DCB on page 7-23*
- *DCD and DCDU on page 7-24*
- *ALIGN on page 7-11*.

A1582E Link Order area '<name>' undefined

A1583E Group symbol '<name>' undefined

A1584W Mode <mode> not allowed for this instruction

A1585E Bad operand type (<typ1>) for operator <op>

A1586E Bad operand types (<typ1>, <typ2>) for operator <op>

A1587E Too many registers <count> in register list, maximum of <max>

A1593E Bad Alignment, must match transfer size UIMM * <dt>

A1595E Bad Alignment, must match <st> * <dt>, or 64 when <st> is 4

A1596E Invalid alignment <align> for dt st combination

A1598E Bad Register list length

A1599E Out of range subscript, must be between 0 and <max_index>

A1600E Section type must be within range SHT_LOOS and SHT_HIUSER

A1601E Immediate cannot be represented

A1603E This instruction inside IT block has UNPREDICTABLE results

A1604W Thumb Branch to destination without alignment to <max> bytes

A1606E Symbol attribute <attr1> cannot be used with attribute <attr2>

A1607E Thumb-2 wide branch instruction used, but offset could fit in Thumb-1 narrow branch instruction

A1608W MOV pc,<rn> instruction used, but BX <rn> is preferred

A1609W MOV <rd>,pc instruction does not set bit zero, so does not create a return address

This warning is caused when the current value of the PC is copied into a register while executing in Thumb state. An attempt to create a return address in this fashion fails as bit0 is not set. Attempting to BX to this instruction causes a state change (to ARM).

To create a return address, you can use:

```
MOV r0, pc
ADDS r0, #1
```

This warning can then be safely suppressed with:

```
--diag-suppress 1609
```

A1611E Register list increment of 2 not allowed for this instruction

A1612E <type> addressing not allowed for <instr>

A1615E Store of a single element or structure to all lanes is UNDEFINED

A1616E Instruction, offset, immediate or register combination is not supported by the current instruction set

This can be caused by attempting to use an invalid combination of operands. For example, in Thumb:

```
MOV r0, #1 ; /* Not permitted */
MOVS r0, #1 ; /* Ok */
```

See the following in the *Assembler Reference*:

- Chapter 3 *ARM and Thumb Instructions*.

A1617E Specified width is not supported by the current instruction set

A1618E Specified instruction is not supported by the current instruction set

A1619E Specified condition is not consistent with previous IT

A1620E Error writing to file '<filename>': <reason>

A1621E CBZ or CBNZ from Thumb code to ARM code

A1622E Negative register offsets are not supported by the current instruction set

A1623E Offset not supported by the current instruction set

A1624W Branch from Thumb code to ARM code

A1625W Branch from ARM code to Thumb code

A1626W BL from Thumb code to ARM code

A1627W BL from ARM code to Thumb code

This occurs when there is a branch from ARM code to Thumb code (or vice-versa) within this file. The usual solution is to move the Thumb code into a separate assembler file. Then, at link-time, the linker adds any necessary interworking veneers.

A1630E Specified processor or architecture does not support ARM instructions

ARM M-profile processors, for example Cortex-M3 and Cortex-M1, implement only the Thumb instruction set, not the ARM instruction set. It is likely that the assembly file contains some ARM-specific instructions and is being built for one of these processors.

A1631E Only left shifts of 1, 2 and 3 are allowed on load/stores

A1632E Else forbidden in IT AL blocks

A1633E LDR rx,= pseudo instruction only allowed in load word form

A1634E LDRD/STRD has no register offset addressing mode in Thumb

A1635E CBZ/CBNZ can not be made conditional

A1636E Flag setting MLA is not supported in Thumb

A1637E Error reading line: <reason>

A1638E Writeback not allowed on register offset loads or stores in Thumb

A1639E Conditional DCI only allowed in Thumb mode

A1640E Offset must be a multiple of four

A1641E Forced user-mode LDM/STM not supported in Thumb

A1642W Relocated narrow branch is not recommended

A1643E Cannot determine whether instruction is working on single or double precision values.

A1644E Cannot use single precision registers with FLDMX/LSTMX

A1645W Substituted <old> with <new>

armasm can warn when it substitutes an instruction when assembling.

For example:

- ADD *negative_number* is the same as SUB *positive_number*
- MOV *negative_number* is the same as MVN *positive_number*
- CMP *negative_number* is the same as CMN *positive_number*.

For the Thumb instruction set, UNPREDICTABLE single register LDMs are transformed into LDRs.

This warning is suppressed by default, but can be enabled with `--diag_warning 1645`

For example:

```
AREA foo, CODE
ADD r0, #-1
MOV r0, #-1
CMP r0, #-1
```

When assembled with:

```
armasm --diag_warning 1645
```

the assembler reports:

```
Warning: A1645W: Substituted ADD with SUB
3 00000000 ADD r0, #-1
Warning: A1645W: Substituted MOV with MVN
4 00000004 MOV r0, #-1
Warning: A1645W: Substituted CMP with CMN
5 00000008 CMP r0, #-1
```

and the resulting code generated is:

```
foo
0x00000000: e2400001 ..@. SUB r0,r0,#1
0x00000004: e3e00000 .... MVN r0,#0
0x00000008: e3700001 ..p. CMN r0,#1
```

A1647E `Bad register name symbol, expected Integer register`

An integer (core) register is expected at this point in the syntax.

A1648E `Bad register name symbol, expected Wireless MMX SIMD register`

This message relates to Wireless MMX.

A1649E `Bad register name symbol, expected Wireless MMX Status/Control or General Purpose register`

This message relates to Wireless MMX.

A1650E `Bad register name symbol, expected any Wireless MMX register`

This message relates to Wireless MMX.

A1651E `TANDC, TEXTRC and TORC instructions with destination register other than R15 is undefined`

This message relates to Wireless MMX.

A1652W `FLDMX/FSTMX instructions are deprecated in ARMv6. Please use FLDMD/FSTMD instructions to save and restore unknown precision values.`

A1653E `Shift instruction using a status or control register is undefined`

A1654E `Cannot access external symbols when loading/storing bytes or halfwords`

A1655E `Instruction is UNPREDICTABLE if halfword/word/doubleword is unaligned`

A1656E `Target must be at least word-aligned when used with this instruction`

A1657E `Cannot load a byte/halfword literal using WLDRB/WLDRH =constant`

A1658W `Support for <opt> is deprecated`

The option passed to `armasm` is now deprecated. Use `armasm --help` to view a summary of the available options.

See the following in the *Assembler Reference*:

- *Assembler command-line options on page 2-3*.

A1659E Cannot B/BL/BLX between ARM/Thumb and Thumb-2EE

A1660E Cannot specify scalar index on this register type

A1661E Cannot specify alignment on this register

A1662E Cannot specify a data type on this register type

A1663E A data type has already been specified on this register

A1664E Data type specifier not recognized

A1665E Data type size must be one of 8, 16, 32 or 64

A1666E Data type size for floating-point must be 32 or 64

A1667E Data type size for polynomial must be 8 or 16

A1668E Too many data types specified on instruction

A1669E Data type specifier not allowed on this instruction

A1670E Expected 64-bit doubleword register expression

A1671E Expected 128-bit quadword register expression

A1672E Expected either 64-bit or 128-bit register expression

A1673E Both source data types must be same type and size

A1674E Source operand 1 should have integer type and be double the size of source operand 2

A1675E Data types and sizes for destination must be same as source

A1676E Destination type must be integer and be double the size of source

A1677E Destination type must be same as source, but half the size

A1678E Destination must be untyped and same size as source

A1679E Destination type must be same as source, but double the size

A1680E Destination must be unsigned and half the size of signed source

A1681E Destination must be unsigned and have same size as signed source

A1682E Destination must be un/signed and source floating, or destination floating and source un/signed, and size of both must be 32-bits

A1683E Data type specifiers do not match a valid encoding of this instruction

A1684E Source operand type should be signed or unsigned with size between <min> and <max>

A1685E Source operand type should be signed, unsigned or floating point with size between <min> and <max>

A1686E Source operand type should be signed or floating point with size between <min> and <max>

A1687E Source operand type should be integer or floating point with size between <min> and <max>

A1688E Source operand type should be untyped with size between <min> and <max>

A1689E Source operand type should be <n>-bit floating point

A1690E Source operand type should be signed with size between <min> and <max>

A1691E Source operand type should be integer, floating point or polynomial with size between <min> and <max>

A1692E Source operand type should be signed, unsigned or polynomial with size between <min> and <max>

A1693E Source operand type should be unsigned or floating point with size between <min> and <max>

A1694E Instruction cannot be conditional in the current instruction set

Conditional instructions are not permitted in the specified instruction set. The instruction MOVEQ, for example, is permitted in ARM and 32-bit Thumb code, but not in 16-bit Thumb code.

A1695E Scalar index not allowed on this instruction

A1696E Expected either 32-bit, 64-bit or 128-bit register expression

A1697E Expected either 32-bit or 64-bit VFP register expression

A1698E Expected 32-bit VFP register expression

A1699E 64-bit data type cannot be used with these registers

A1700E Source operand type should be integer with size between <min> and <max>

A1701E 16-bit polynomial type cannot be used for source operand

A1702E Register Dm can not be scalar for this instruction

A1704E Register Dm must be in the range D0-D<upper> for this data type

A1705W Assembler converted Qm register to D<rnum>[<idx>]

A1706E Register Dm must be scalar

A1708E 3rd operand to this instruction must be a constant expression

A1709E Expected ARM or scalar register expression

A1710E Difference between current and previous register should be <diff>

A1711E Scalar registers cannot be used in register list for this instruction

A1712E This combination of LSB and WIDTH results in UNPREDICTABLE behavior

A1713E Invalid field specifiers for APSR: must be APSR_ followed by at least one of n, z, c, v, q or g

A1714E Invalid combination of field specifiers for APSR

A1715E PSR not defined on target architecture

A1716E Destination for VMOV instruction must be ARM integer, 32-bit single-precision, 64-bit doubleword register or 64-bit doubleword scalar register

A1717E Source register must be an ARM integer, 32-bit single-precision or 64-bit doubleword scalar register

A1718E Source register must be an ARM integer register or same as the destination register

A1719W This PSR name is deprecated and may be removed in a future release

A1720E Source register must be a 64-bit doubleword scalar register

A1721E Destination register may not have all-lanes specifier

A1722E Labels not allowed inside IT blocks

A1723W __RELOC is deprecated, please use the new RELOC directive

A1724E RELOC may only be used immediately after an instruction or data generating directive

A1725W 'armasm inputfile outputfile' form of command-line is deprecated

A1726W Decreasing --max_cache below 8MB is not recommended

A1727W Immediate could have been generated using the 16-bit Thumb MOVS instruction

A1728E Source register must be same type as destination register

A1729E Register list may only contain 32-bit single-precision or 64-bit doubleword registers

A1730E Only IA or DB addressing modes may be used with these instructions

A1731E Register list increment of 2 or more is not allowed for quadword registers

A1732E Register list must contain between 1 and 4 contiguous doubleword registers

A1733E Register list must contain 2 or 4 doubleword registers, and increment 2 is only allowed for 2 registers

A1734E Register list must contain <n> doubleword registers with increment 1 or 2

A1735E Post-indexed offset must equal the number of bytes loaded/stored (<n>)

A1736E Number of registers in list must equal number of elements

A1737E PC or SP can not be used as the offset register

A1738E Immediate too large for this operation

A1739W Constant generated using single VMOV instruction; second instruction is a NOP

A1740E Number of bytes in FRAME PUSH or FRAME POP directive must not be less than zero

A1741E Instruction cannot be conditional

A1742E Expected LSL #Imm

A1744E Alignment on register must be a multiple of 2 in the range 16 to 256

A1745W This register combination is DEPRECATED and may not work in future architecture revisions

This warning is generated when all of the following conditions are satisfied:

- you are using a deprecated register combination, for example:
  ```
  PUSH    {r0, pc}
  ```
- you are assembling for a target architecture that supports 32-bit Thumb instructions, in other words ARMv6T2 or later
- you are assembling to ARM code.

Note

- When assembling to Thumb, rather than ARM code, and the target architecture is ARMv6T2 or later, the assembler generates error A1477E instead.
- When assembling for an architecture or processor that does not support 32-bit Thumb instructions, in other words ARM architectures before ARMv6T2, by default no diagnostic is emitted.

A1746W `Instruction stall diagnostics may be unreliable for this CPU`

This warning is shown when you enable message A1563W for a processor that is not modelled accurately by the assembler. It indicates that you cannot rely on the output of A1563W when improving your code.

See also warning `A1563W`.

A1753E `Unrecognized memory barrier option`

A1754E `Cannot change the type of a scalar register`

A1755E `Scalar index has already been specified on this register`

A1756E `Data type must be specified on all registers`

A1757W `Symbol attributes must be within square brackets; Any other syntax is deprecated`

A1758W `Exporting multiple symbols with this directive is deprecated`

A1759E `Specified processor or architecture does not support Thumb-2EE instructions`

A1760W `Build Attribute <from> is '<attr>'`

A1761W `Difference in build attribute from '<diff>' in <from>`

A1762E `Branch offset 0x<val> out of range of 16-bit Thumb branch, but offset encodable in 32-bit Thumb branch`

This is caused when assembling for Thumb if an offset to a branch instruction is too large to fit in a 16-bit branch. The `.W` suffix can be added to the instruction to instruct the assembler to generate a 32-bit branch.

A1763W `Inserted an IT block for this instruction`

This indicates that the assembler has inserted an IT block to permit a number of conditional instructions in Thumb code. For example:

```
MOVEQ r0,r1
```

This warning is off by default. It can be enabled using `--diag_warning A1763`.

A1764W `<name> instructions are deprecated in architecture <arch> and above`

A1765E `Size of padding value on ALIGN must be 1, 2 or 4 bytes`

This is caused when the optional `padsize` attribute is used with an `ALIGN` directive, but has an incorrect size. It does not refer to the parameter to align to. The parameter can be any power of 2 from 2^0 to 2^31

A1766W `Size of padding value for code must be a minimum of <size> bytes; treating as data`

A1767E `Unexpected characters following attribute`

A1768E Missing '='

A1769E Bad NEON or VFP system register name symbol

A1771E Bad floating-point bitpattern when expecting <exp>-bit bitpattern

A1772E Destination type must be signed or unsigned integer, and source type must be 32-bit or 64-bit floating-point

A1773E Floating-point conversion only possible between 32-bit single-precision and 64-bit double-precision types

A1774E Fixed-point conversion only possible for 16-bit or 32-bit signed or unsigned types

A1775E Conversion between these types is not possible

A1776E This operation is not available for 32-bit single-precision floating point types

A1777E <n> is out of range for symbol type; value must be between <min> and <max>

A1778E <n> is out of range for symbol binding; value must be between <min> and <max>

A1779E DCDO cannot be used on READONLY symbol '<key>'

A1780E Unknown ATTR directive

A1781E Tag #<id> cannot be set by using ATTR

A1782E Tag #<id> should be set with ATTR <cmd>

A1783E Attribute scope must be a label or section name

A1784W Reference to weak definition '<sym>' not relocated

A1785E Macro '<macuse>' not found, but '<macdef>' exists

A1786W This instruction using SP is deprecated and may not work in future architecture revisions

This warning is generated when all of the following conditions are satisfied:

- you explicitly use the SP in a deprecated way, for example:
  `ADD sp, r0, #100`
- you are assembling for a target architecture that supports 32-bit Thumb instructions, in other words ARMv6T2 or later
- you are assembling to ARM code.

ARM deprecates the explicit use of the SP in ARM instructions in any way that is not possible in the corresponding Thumb instruction. Such deprecated register uses are still possible in ARM instructions for backwards compatibility and you can suppress this warning by using the assembler's command line option `--diag_suppress=1786`. However, ARM recommends you modify your code, because it might not work in future architecture revisions.

You can replace the deprecated use of the SP shown in the example with a sequence like:

```
ADD r1, r0, #100
MOV sp, r1
```

Note

- When assembling to Thumb, rather than ARM code, and the target architecture is ARMv6T2 or later, the assembler generates error A1477E instead.
- When assembling for an architecture or processor that does not support 32-bit Thumb instructions, in other words ARM architectures before ARMv6T2, by default no diagnostic is emitted.

A1787W `Use of VFP Vector Mode is deprecated in ARMv7`

A1788W `Explicit use of PC in this instruction is deprecated and may not work in future architecture revisions`

This warning is generated when all of the following conditions are satisfied:

- you explicitly use the PC in a deprecated way, for example:
  ```
  CMP pc, #1
  ```
- you are assembling for a target architecture that supports 32-bit Thumb instructions, in other words ARMv6T2 or later
- you are assembling to ARM code.

ARM deprecates most explicit uses of the PC in ARM instructions, although they are still possible for backwards compatibility. You can suppress this warning by using the assembler's command line option `--diag_suppress=1788`. However, ARM recommends you modify your code, because it might not work in future architecture revisions.

Note

- When assembling to Thumb rather than ARM code, and the target architecture is ARMv6T2 or later, the assembler generates error A1477E instead.
- When assembling for an architecture or processor that does not support 32-bit Thumb instructions, in other words ARM architectures before ARMv6T2, by default no diagnostic is emitted.

A1789W `Explicit use of PC in this instruction is deprecated and and may not work in future architecture revisions, except as destination register`

A1790W `Writeback ignored in Thumb LDM loading the base register`

This is caused by incorrectly adding an exclamation mark to indicate base register writeback.

For example:

```
LDM r0!, {r0-r4}
```

is not a legal instruction because r0 is the base register and is also in the destination register list. In this case, the assembler ignores the writeback and generates:

```
LDM r0, {r0-r4}
```

A1791W `Previous value of tag #<id> will be overridden`

A1792E `Undefined build attributes tag`

A1793E `Conversion only possible between 16-bit and 32-bit floating point`

A1794E `Conversion operations require two data types`

A1795E Source and destination vector must contain <n> elements

A1796E Register type not consistent with data type

A1797E Specified FPU is not compatible with CPU architecture

A1798W Output is not WYSIWYG (<output>)

A1799W Output has not been checked for WYSIWYG property

A1800W No output for line

A1801E Instruction is UNPREDICTABLE in current instruction set

A1803E Bad system instruction name

A1804E Bad CP14 or CP15 register name for instruction

A1805W Register is Read-Only

A1806W Register is Write-Only

A1807W Instruction executes as NOP on target CPU

A1808E Generated object file may be corrupt (<reason>)

A1809W Instruction aligns PC before using it; section ought to be at least 4 byte aligned

This warning is generated when all the following conditions apply:

- you are using a PC-relative offset in a Thumb instruction that requires the PC to be word-aligned
- the code section containing this instruction has less than 4-byte alignment
- the instruction is not relocated at link time (because of a relocation emitted by the assembler).

If these conditions are all met, and the code section containing this instruction is not placed at a 4-byte aligned address when linking, the instruction might operate on or with the wrong address at runtime. This is because the instruction aligns the PC to a 4-byte address before using it.

The following example shows an `LDR` instruction in Thumb that is diagnosed by this warning because the section has an alignment of 2 bytes:

```
AREA ||.text||, CODE, READONLY, ALIGN=1
THUMB
LDR r0, [pc, #8]  ; gives warning A1809W
```

A1810E Base register writeback value unclear; use '[rn,#n]!' or '[rn],#n' syntax

A1811E Size of fill value must be 1, 2 or 4 bytes and a factor of fill size

A1812W Instruction cannot be assembled in the opposite instruction set

A1813W 32-bit instruction used where 16-bit could have been used

A1814E No output file

A1815E SHT_ARM_EXIDX sections require a link order dependency to be set

A1816E Unknown opcode '<name>' in CODE16, but exists in THUMB

A1817W ATTR tag #<id> setting ignored in <scope>

A1818W ATTR COMPAT flag <flag> and vendor '<vendor>' setting ignored in <scope>

A1819W `ATTR compatible with tag #<id> setting ignored in <scope>`

A1820E `Register and processor mode not valid for instruction`

A1846E `Invalid field specifiers for CPSR or SPSR: must be followed by at least one of c, x, s or f`

A1847E `Expression requiring more than one relocation not allowed`

This can occur during the assembly of ARM instructions when trying to access data in another area. For example, using:

```
LDR r0, [pc, #label - . - 8]
```

or its equivalent:

```
LDR r0, [pc, #label-{PC}-8]
```

where `label` is defined in a different `AREA`.

Change your code to use the simpler, equivalent syntax:

```
LDR r0, label
```

This works if `label` is either in the same area or in a different area.

A1848W `State change in IT block`

A1875E `Register Rn must be from R0 to R7 in this instruction`

Change the specified register to be in the range R0 to R7.

A1903E `Line not seen in first pass; cannot be assembled`

This occurs if an instruction or directive does not appear in pass 1 but appears in pass 2 of the assembler.

The following example shows when a line is not seen in pass 1:

```
    AREA x,CODE
    [ :DEF: foo
num EQU 42 ; assembler does not see this line during pass 1 because
           ; foo is not defined at this point during pass 1
    ]
foo DCD num
    END
```

A1907W `Test for this symbol has been seen and may cause failure in the second pass.`

This diagnostic is suppressed by default. Enable it to identify situations that might result in errors A1903E, A1909E, or A1908E.

A1908E `Label '<name>' value inconsistent: in pass 1 it was <val1>; in pass 2 it was <val2>`

The following example generates this error because in pass 1 the value of `x` is `0x0004+r9`, and in pass 2 the value of `x` is `0x0000+r0`:

```
    map 0, r0
    if :lnot: :def: sym
        map 0, r9
        field 4
    endif
x   field 4
sym LDR r0, x
```

A1909E `Line not seen in second pass; cannot be assembled`

This occurs if an instruction or directive appears in pass 1 but does not appear in pass 2 of the assembler.

The following example shows when a line is not seen in pass 2:

```
    AREA x,CODE
    [ :LNOT: :DEF: foo
    MOV r1, r2  ; assembler does not see this line during pass 2 because
                ; foo is already defined
    ]
foo MOV r3, r4
    END
```

A1916E Unknown built-in variable '<name>'

A1993E This operator requires a relocation that is not supported in <objfmt>

A1994E This directive is not supported in <objfmt>

A1995E Weak definitions are not supported in <objfmt>

A1996E TYPE must only be used after WEAK on IMPORT

A1997E Expected alias for weak extern symbol

A1998E Comdat Associated area must have Comdat Associative selection type

A1999E Comdat Associated area cannot be another Comdat Associated area

Chapter 4
Linker Errors and Warnings

The following topics describe the error and warning messages for the linker, `armlink`:

4.1 Suppressing armlink error and warning messages

All linker warnings are suppressible with `--diag_suppress` in the same way as for compiler warnings. For example:

```
--diag_suppress 6306
```

Some errors such as L6220E, L6238E and L6784E can be downgraded to a warning by using:

```
--diag_warning
```

4.2 List of the armlink error and warning messages

The error and warning messages for `armlink` are:

L6000U `Out of memory.`

This error is reported by RVCT v4.0 and earlier. For more details on why you might see this error and possible solutions, see the description for error `L6815U`.

L6001U `Could not read from file <filename>.`

L6002U `Could not open file <filename>: <reason>`

This indicates that the linker was unable to open a file specified on the linker command line. This can indicate a problem accessing the file or a fault with the command line specified. Some common occurrences of this message are:

- `L6002U: Could not open file /armlib/{libname}: No such file or directory`

  Either specify the library path with `--libpath` or set the `ARMCC`*`n`*`LIB` environment variable to *`install_directory`*`\lib`.

  See the following in the *Linker Reference*:

  — *--libpath=pathlist on page 2-96*.

  See the following in *Introducing the ARM Compiler toolchain*:

  — *Toolchain environment variables* on page 2-15.

  Note

  In ARM Compiler toolchain v5.0 and later, `armlink` does not require the `ARMCC`*`n`*`LIB` environment variables to be set.

- `Error : armlink : L6002: Could not open file errors=ver.txt`

  Caused by the double-dash (`--`) missing from in front of `errors=ver.txt`. If you do not prefix options with `--` or `-` the linker treats them as input files and fails the link step because it is unable to load all the specified files. The correct switch is `--errors=ver.txt`

L6003U `Could not write to file <filename>.`

An file I/O error occurred while reading, opening, or writing to the specified file.

L6004U `Incomplete library member list <list> for <library>.`

This can occur where there is whitespace in the list of library objects.

The following example fails:

```
armlink x.lib(foo.o, bar.o)
Fatal error: L6004U: Missing library member in member list for x.lib.
```

The following example succeeds:

```
armlink x.lib(foo.o,bar.o)
```

Another less common occurrence is caused by a corrupt library, or possibly a library in an unsupported format.

L6005U `Extra characters on end of member list for <library>.`

L6006U `Overalignment value not specified with OVERALIGN attribute for execution region <regionname>.`

See the following in the *Linker Reference*:

- *Syntax of an input section description on page 4-23*

See the following in *Using the Linker*:

- *Overalignment of execution regions and input sections* on page 8-58.

L6007U `Could not recognize the format of file <filename>.`

The linker can recognize object files in the ELF format, and library files in AR formats. The specified file is either corrupt, or is in a file format that the linker cannot recognize.

L6008U `Could not recognize the format of member <mem> from <lib>.`

The linker can recognize library member objects in the ELF file format. The specified library member is either corrupt, or is in a file format that the linker cannot recognize.

L6009U `File <filename> : Endianness mismatch.`

The endianness of the specified file or object did not match the endianness of the other input files. The linker can handle input of either big endian or little endian objects in a single link step, but not a mixed input of some big and some little endian objects.

L6010U `Could not reopen stderr to file <filename>: <reason>`

An file I/O error occurred while reading, opening, or writing to the specified file.

L6011U `Invalid integer constant : <number>.`

Specifying an illegal integer constant causes this. An integer can be entered in hexadecimal format by prefixing `&`, `0x`, or `0X`. You can use a suffix of `k` or `m` to specify a multiple of 1024 or 1024*1024.

L6015U `Could not find any input files to link.`

The linker must be provided with at least one object file to link.

For example, if you try to link with:

```
armlink lib.a -o foo.axf
```

the linker reports this error.

You must instead use, for example:

```
armlink foo_1.o foo_2.o lib.a -o foo.axf
```

L6016U `Symbol table missing/corrupt in object/library <object>.`

This can occur when linking with libraries built with the GNU tools. This is because GNU `ar` can generate incompatible information.

The workaround is to replace `ar` with `armar` and use the same command line arguments. Alternatively, the error is recoverable by using `armar -s` to rebuild the symbol table.

L6017U `Library <library> symbol table contains an invalid entry, no member at offset 0x<offset>.`

The library might be corrupted. Try rebuilding it.

L6018U `<filename> is not a valid ELF file.`

L6019U `<filename> is not a valid 64 bit ELF file.`

L6020U `<filename> is not a valid 32 bit ELF file.`

L6022U `Object <objname> has multiple <table>.`

The object file is faulty or corrupted. This might indicate a compiler fault. Contact your supplier.

L6024U `Library <library> contains an invalid member name.`

The file specified is not a valid library file, is faulty or corrupted. Try rebuilding it.

L6025U `Cannot extract members from a non-library file <library>.`

The file specified is not a valid library file, is faulty or corrupted. Try rebuilding it.

L6026U `ELF file <filename> has neither little or big endian encoding`

The ELF file is invalid. Try rebuilding it.

L6027U `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) has invalid/unknown type.`

This might indicate a compiler fault. Contact your supplier.

L6028U `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) has invalid offset.`

This might indicate a compiler fault. Contact your supplier.

L6029U `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) is wrt invalid/missing symbol.`

The relocation is with respect to a symbol that is either:

- invalid or missing from the object symbol table
- a symbol that is not suited to be used by a relocation.

This might indicate a compiler fault. Contact your supplier.

L6030U `Overalignment <overalignment> for region <regname> must be at least 4 and a power of 2`

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*
- *Syntax of an input section description on page 4-23*

See the following in *Using the Linker*:

- *Overalignment of execution regions and input sections* on page 8-58.

L6031U `Could not open scatter description file <filename>: <reason>`

An I/O error occurred while trying to open the specified file. This could be because of an invalid filename.

L6032U `Invalid <text> <value> (maximum <max_value>) found in <object>`

L6033U `Symbol <symbolname> in <objname> is defined relative to an invalid section.`

L6034U `Symbol <symbolname> in <objname> has invalid value.`

This is most often caused by a section-relative symbol having a value that exceeds the section boundaries.

L6035U `Relocation #<rel_class>:<rel_number> in ZI Section <objname>(<secname>) has invalid type.`

ZI Sections cannot have relocations other than of type R_ARM_NONE.

L6036U `Could not close file <filename>: <reason>`

An I/O error occurred while closing the specified file.

L6037U `'<arg>' is not a valid argument for option '<option>'.`

The argument is not valid for this option. This could be because of a spelling error, or because of the use of an unsupported abbreviation of an argument.

L6038U `Could not create a temporary file to write updated SYMDEFS.`

An I/O error occurred while creating the temporary file required for storing the SYMDEFS output.

L6039W `Relocation from #<rel_class>:<rel_number> in <objname>(<secname>) with respect to <symname>. Skipping creation of R-type relocation. No corresponding R-type relocation exists for type <rel_type>.`

`--reloc` is used with objects containing relocations that do not have a corresponding R-type relocation.

L6041U `An internal error has occurred (<clue>).`

Contact your supplier.

L6042U `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) is wrt a mapping symbol(#<idx>, Last Map Symbol = #<last>).`

Relocations with respect to mapping symbols are not permitted. This might indicate a compiler fault. Contact your supplier.

L6043U `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) is with respect to an out of range symbol(#<val>, Range = 1-<max>).`

Relocations can only be made wrt symbols in the range (1-n), where n is the number of symbols.

L6047U `The size of this image (<actual_size> bytes) exceeds the maximum allowed for this version of the linker`

L6048U `The linker is unable to continue the link step (<id>). This version of the linker will not create this image.`

L6049U `The linker is unable to continue the link step (<id>). This version of the linker will not link with one or more given libraries.`

L6050U `The code size of this image (<actual_size> bytes) exceeds the maximum allowed for this version of the linker.`

L6058E `Syntax error parsing linker script <script> at line <lineno> : <token>`

The link ld script has a syntax error at the line number.

See the following in *Using the Linker*:

- Chapter 9 *GNU ld script support in armlink*.

L6064E `ELF Executable file <filename> given as input on command line`

This might be because you specified an object file as output from from the compiler without specifying the `-c` compiler option. For example:

```
armcc file.c -o file.o

armlink file.o -o file.axf
```

See the following in the *Compiler Reference*:

- *-c on page 3-31*.

L6065E `Load region <name> (size <size>) is larger than maximum writable contiguous block size of 0x80000000.`

The linker attempted to write a segment larger than 2GB. The size of a segment is limited to 2GB.

L6175E `EMPTY region <regname> cannot have any section selectors.`

L6176E `A negative max_size cannot be used for region <regname> without the EMPTY attribute.`

Only regions with the EMPTY attribute are permitted to have a negative max-size.

L6177E `A negative max_size cannot be used for region <regname> which uses the +offset form of base address.`

Regions using the `+offset` form of base address are not permitted to have a negative max-size.

L6188E `Special section <sec1> multiply defined by <obj1> and <obj2>.`

A special section is one that can only be used once, such as "`Veneer$$Code`".

L6195E `Cannot specify both '<attr1>' and '<attr2>' for region <regname>`

See the following in the *Linker Reference*:

- *Load region attributes on page 4-8*
- *Execution region attributes on page 4-12*
- *Address attributes for load and execution regions on page 4-15*
- *Inheritance rules for load region address attributes on page 4-19*
- *Inheritance rules for execution region address attributes on page 4-20*
- *Inheritance rules for the RELOC address attribute on page 4-21*.

L6200E `Symbol <symbolname> multiply defined by <object1> and <object2>.`

A common example where this occurs:

`Symbol __stdout multiply defined (by retarget.o and stdio.o).`

means that there are two conflicting definitions of `__stdout` present in `retarget.o` and `stdio.o`. The one in `retarget.o` is your own definition. The one in `stdio.o` is the default implementation, which was probably linked-in inadvertently.

`stdio.o` contains a number symbol definitions and implementations of file functions like `fopen`, `fclose`, and `fflush`.

`stdio.o` is being linked-in because it satisfies some unresolved references.

To identify why `stdio.o` is being linked-in, you must use the `verbose` link option switch. For example:

```
armlink [... your normal options...] --verbose --list err.txt
```

Then study `err.txt` to see exactly what the linker is linking in, from where, and why.

You might have to either:

- eliminate the calls like `fopen`, `fclose`, and `fflush`
- re-implement the `_sys_xxxx` family of functions.

See the following in *Using ARM® C and C++ Libraries and Floating-Point Support*:

- *Tailoring input/output functions in the C and C++ libraries on page 2-92*.

L6201E `Object <objname> contains multiple entry sections.`

The input object specifies more than one entry point. Use the `--entry` command-line option to select the entry point to use.

See the following in the *Linker Reference*:

- *--entry=location on page 2-58*.

L6202E `<objname>(<secname>) cannot be assigned to non-root region '<regionname>'`

A root region is a region that has an execution address the same as its load address. The region does not therefore require moving or copying by the scatter-load initialization code.

Certain sections must be placed in a root region in the image, including:

- `__main.o`
- The linker-generated table (`Region$$Table`)
- Scatter-loading (`__scatter*.o`) objects from the library
- Decompressor (`__dc*.o`) objects from the library

If a required section is not placed in a root region, the linker reports, for example:

```
anon$$obj.o(Region$$Table) cannot be assigned to a non-root region 'RAM'.
```

You can use `InRoot$$Sections` to include all required sections in a root region:

```
ROM_LOAD 0x0000 0x4000
{
  ROM_EXEC 0x0000 0x4000 ; root region
  {
    vectors.o (Vect, +FIRST) ; Vector table
    * (InRoot$$Sections) ; All library sections
                         ; that must be in a root region
                         ; for example, __main.o, __scatter*.o,
                         ; dc*.o and * Region$$Table
  }
  RAM 0x10000 0x8000
  {
    * (+RO, +RW, +ZI) ; all other sections
  }
}
```

L6203E `Entry point (<address>) lies within non-root region <regionname>.`

The image entry point must correspond to a valid instruction in a root-region of the image.

L6204E `Entry point (<address>) does not point to an instruction.`

The image entry point you specified with the `--entry` command-line option must correspond to a valid instruction in the root-region of the image.

See the following in the *Linker Reference*:

- *--entry=location on page 2-58*.

L6205E `Entry point (<address>) must be word aligned for ARM instructions.`

This message is displayed because the image entry point you specified with the `--entry` command-line option is not word aligned. For example, you specified `--entry=0x8001` instead of `--entry=0x8000`.

See the following in the *Linker Reference*:

- *--entry=location on page 2-58*.

L6206E `Entry point (<address>) lies outside the image.`

The image entry point you specified with the `--entry` command-line option is outside the image. For example, you might have specified an entry address of `0x80000` instead of `0x8000`, as follows:

```
armlink --entry=0x80000 test.o -o test.axf
```

See the following in the *Linker Reference*:

- *--entry=location on page 2-58*.

L6208E `Invalid argument for --entry command: '<arg>'`

See the following in the *Linker Reference*:

- *--entry=location on page 2-58*.

L6209E `Invalid offset constant specified for --entry (<arg>)`

See the following in the *Linker Reference*:

- *--entry=location on page 2-58*.

L6210E `Image cannot have multiple entry points. (<address1>,<address2>)`

One or more input objects specifies more than one entry point for the image. Use the `--entry` command-line option to select the entry point to use.

See the following in the *Linker Reference*:

- *--entry=location on page 2-58*.

L6211E `Ambiguous section selection. Object <objname> contains more than one section.`

This can occur when using the linker option `--keep` on an assembler object that contains more than one `AREA`. The linker must know which `AREA` you want to keep.

To solve this, use more than one `--keep` option to specify the names of the AREAs to keep, such as:

```
--keep boot.o(vectors) --keep boot.o(resethandler) …
```

Note

Using assembler files with more than one `AREA` might give other problems elsewhere, so this is best avoided.

L6213E `Multiple First section <object2>(<section2>) not allowed. <object1>(<section1>) already exists.`

Only one `FIRST` section is permitted.

L6214E `Multiple Last section <object2>(<section2>) not allowed. <object1>(<section1>) already exists.`

Only one `LAST` section is permitted.

L6215E `Ambiguous symbol selection for --First/--Last. Symbol <symbol> has more than one definition.`

See the following in the *Linker Reference*:

- *--first=section_id on page 2-70*
- *--last=section_id on page 2-93*.

L6216E `Cannot use base/limit symbols for non-contiguous section <secname>`

The exception-handling index tables generated by the compiler are given the section name `.ARM.exidx`. For more information, see *Exception Handling ABI for the ARM Architecture*, http://infocenter.arm.com/help/topic/com.arm.doc.ihi0038-/index.html.

At link time these tables must be placed in the same execution region and be contiguous. If you explicitly place these sections non-contiguously using specific selector patterns in your scatter file, then this error message is likely to occur. For example:

```
LOAD_ROM 0x00000000
{
  ER1 0x00000000
  {
```

```
    file1.o (+RO) ; from a C++ source
    * (+RO)
  }
  ER2 0x01000000
  {
    file2.o (+RO) ; from a C++ source
  }
  ER3 +0
  {
    * (+RW, +ZI)
  }
}
```

This might produce the following error if exception-handling index tables are in both `file1.o` and `file2.o`, because the linker cannot place them in separate regions:

```
Error: L6216E: Cannot use base/limit symbols for non-contiguous section
.ARM.exidx
```

Also, the `.init_array` sections must be placed contiguously within the same region for their base and limit symbols to be accessible.

The corrected example is:

```
LOAD_ROM 0x00000000
{
  ER1 0x00000000
  {
    file1.o (+RO) ; from a C++ source
    * (.ARM.exidx)  ; Section .ARM.exidx must be placed explicitly,
                    ; otherwise it is shared between two regions, and
                    ; the linker is unable to decide where to place it.
    *(.init_array)  ; Section .init_array must be placed explicitly,
                    ; otherwise it is shared between two regions, and
                    ; the linker is unable to decide where to place it.
    * (+RO)
  }
  ER2 0x01000000
  {
    file2.o (+RO) ; from a C++ source
  }
  ER3 +0
  {
    * (+RW, +ZI)
  }
}
```

In the corrected example, the base and limit symbols are contained in `.init_array` in a single region.

For more information, see the following in *Using ARM® C and C++ Libraries and Floating-Point Support*:

- *How C and C++ programs use the library functions on page 2-54*
- *C++ initialization, construction and destruction on page 2-56*.

L6217E `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) with respect to <symbol>. R_ARM_SBREL32 relocation to imported symbol`

L6218E `Undefined symbol <symbol> (referred from <objname>).`

Some common examples where this can occur are:

- User Error. Somebody has referenced a symbol and has either forgotten to define it or has incorrectly defined it.
- `Undefined symbol __ARM_switch8 or __ARM_ll_<xxxx> functions`

The helper functions are automatically generated into the object file by the compiler.

Note

An undefined reference error can, however, still be generated if linking objects from legacy projects where the helper functions are in the `h_xxx` libraries (`h` indicates that these are compiler helper libraries, rather than standard C library code).

Re-compile the object or ensure that these libraries can be found by the linker.

- When attempting to refer to a function/entity in C from a function/entity in C++. This is caused by C++ name mangling, and can be avoided by marking C functions `extern "C"`.
- `Undefined symbol thunk{v:0,-44} to Foo_i::~Foo_i() (referred from Bar_i.o)`

  The symbol `thunk{v:0,-44} to Foo_i::~Foo_i()` is a wrapper function round the regular `Foo_i::~Foo_i()`.

  `Foo_i` is a derived class of some other base class, therefore:

  — it has a base-class vtable for when it is referred to by a pointer to that base class

  — the base-class vtable has an entry for the thunk

  — the destructor thunk is output when the actual (derived class) destructor is output.

  Therefore, to avoid the error, ensure this destructor is defined.
- `Undefined symbol main (referred from kernel.o)`

  The linker is reporting that your application does not include a `main()` function.

  See the following in *Migration and Compatibility*:

  — *C and C++ library changes between RVCT v2.2 and RVCT v3.0* on page 11-5

L6219E `<type> section <object1>(<section1>) attributes {<attributes>} incompatible with neighboring section <object2>(<section2>).`

This error occurs when the default ordering rules used by the linker (RO followed by RW followed by ZI) are violated. This typically happens when one uses `+FIRST` or `+LAST`, for example in a scatter file, attempting to force RW before RO.

L6220E `<type> region <regionname> size (<size> bytes) exceeds limit (<limit> bytes).`

Example:

```
Execution region ROM_EXEC size (4208184 bytes) exceeds limit (4194304 bytes).
```

This can occur where a region has been given an (optional) maximum length in the scatter file, but this size of the code/data being placed in that region has exceeded the given limit. This error is suppressible with `--diag_suppress 6220`.

For example, this might occur when using `.ANY`*num* selectors with the `ALIGN` directive in a scatter file to force the linker to insert padding. You might be able to fix this using the `--any_contingency` option.

See the following in *Using the Linker*:

- *Placing unassigned sections with the .ANY module selector* on page 8-25.

See the following in the *Linker Reference*:

- *--any_contingency on page 2-8*
- *--diag_suppress=tag[,tag,...] on page 2-47*.

L6221E `<type1> region <regionname1> with <addrtype1> range [<base1>,<limit1>) overlaps with <type2> region <regionname2> with <addrtype2> range [<base2>,<limit2>).`

This represents an incorrect scatter file. A non-ZI section must have a unique load address and in most cases must have a unique execution address. This error might be because a load region LR2 with a relative base address immediately follows a ZI execution region in a load region LR1. From RVCT v3.1 onwards, the linker no longer assigns space to ZI execution regions.

See the following in the *Linker Reference*:

- *Scatter files containing relative base address load regions and a ZI execution region on page 4-37*.

L6222E `Partial object cannot have multiple ENTRY sections, <e_oname>(<e_sname>) and <oname>(<sname>).`

Where objects are being linked together into a partially-linked object, only one of the sections in the objects can have an entry point.

Note

It is not possible in this case to use the linker option `--entry` to select one of the entry points.

L6223E `Ambiguous selectors found for <objname>(<secname>) from Exec regions <region1> and <region2>.`

This occurs if the scatter file specifies `<objname>(<secname>)` to be placed in more than one execution region. This can occur accidentally when using wildcards (`*`). The solution is to make the selections more specific in the scatter file.

L6224E `Could not place <objname>(<secname>) in any Execution region.`

This occurs if the linker can not match an input section to any of the selectors in your scatter file. You must correct your scatter file by adding an appropriate selector.

See the following in *Using the Linker*:

- *Section placement with the linker* on page 4-19.

L6225E `Number <str...> is too long.`

L6226E `Missing base address for region <regname>.`

L6227E `Using --reloc with --rw-base without --split is not allowed.`

See the following in the *Linker Reference*:

- *--reloc on page 2-131*
- *--rw_base=address on page 2-138*
- *--split on page 2-153*.

L6228E `Expected '<str1>', found '<str2>'.`

L6229E `Scatter description <file> is empty.`

L6230E `Multiple execution regions (<region1>,<region2>) cannot select <secname>.`

L6231E `Missing module selector.`

L6232E `Missing section selector.`

L6233E `Unknown section selector '+<selector>'.`

L6234E `<ss> must follow a single selector.`

For example, in a scatter file:

```
:
* (+FIRST, +RO)
:
```

`+FIRST` means place this (single) section first. Selectors that can match multiple sections (for example, `+RO` or `+ENTRY`) are not permitted to be used with `+FIRST` (or `+LAST`). If used together, the error message is generated.

L6235E `More than one section matches selector - cannot all be FIRST/LAST.`

See the following in *Using the Linker*:

- *Placing sections with FIRST and LAST attributes* on page 4-21.

See the following in the *Linker Reference*:

- *Syntax of an input section description on page 4-23*.

L6236E `No section matches selector - no section to be FIRST/LAST.`

The scatter file specifies a section to be `+FIRST` or `+LAST`, but that section does not exist, or has been removed by the linker because it believes it to be unused. Use the linker option `--info unused` to reveal which objects are removed from your project. Example:

```
ROM_LOAD 0x00000000 0x4000
{
 ROM_EXEC 0x00000000
 {
   vectors.o (Vect, +First) << error here
   * (+RO)
 }
 RAM_EXEC 0x40000000
 {
   * (+RW, +ZI)
 }
}
```

Some possible solutions are:

- Ensure `vectors.o` is specified on the linker command-line.
- Link with `--keep vectors.o` to force the linker not to remove this, or switch off this optimization entirely, with `--no_remove`. ARM does not recommend this.
- ARM recommends that you add the `ENTRY` directive to `vectors.s`, to tell the linker that it is a possible entry point of your application such as, for example:

  ```
  AREA Vect, CODE
  ENTRY ; define this as an entry point
  Vector_table
  ...
  ```

  Then link with `--entry Vector_table` to define the real start of your code.

See the following in *Using the Linker*:

- *Placing sections with FIRST and LAST attributes* on page 4-21.

See the following in the *Linker Reference*:

- *--entry=location on page 2-58*
- *--info=topic[,topic,...] on page 2-79*

- *--keep=section_id on page 2-88*
- *--remove, --no_remove on page 2-133*
- *Syntax of an input section description on page 4-23*.

See the following in the *Assembler Reference*:

- *ENTRY on page 7-32*.

L6237E `<objname>(<secname>) contains relocation(s) to unaligned data.`

L6238E `<objname>(<secname>) contains invalid call from '<attr1>' function to '<attr2>' function <sym>.`

This linker error is given where a stack alignment conflict is detected in object code. The *ABI for the ARM Architecture* suggests that code maintains eight-byte stack alignment at its interfaces. This permits efficient use of LDRD and STRD instructions (in ARM Architecture 5TE and later) to access eight-byte aligned `double` and `long long` data types.

Symbols such as `~PRES8` and `REQ8` are *Build Attributes* of the objects:

- `PRES8` means the object PREServes eight-byte alignment of the stack
- `~PRES8` means the object does NOT preserve eight-byte alignment of the stack (~ meaning NOT)
- `REQ8` means the object REQuires eight-byte alignment of the stack.

This link error typically occurs in two cases:

- Where assembler code (that does not preserve eight-byte stack alignment) calls compiled C/C++ code (that requires eight-byte stack alignment).
- Where attempting to link legacy objects that were compiled with older tools with objects compiled with recent tools. Legacy objects that do not have these attributes are treated as `~PRES8`, even if they do actually happen to preserve eight-byte alignment.

For example:

```
Error: L6238E: foo.o(.text) contains invalid call from '~PRES8' function
to 'REQ8' function foobar
```

This means that there is a function in the object `foo.o` (in the section named `.text`) that does not preserve eight-byte stack alignment, but which is trying to call function `foobar` that requires eight-byte stack alignment.

A similar warning that might be encountered is:

```
Warning: L6306W: '~PRES8' section foo.o(.text) should not use the address
of 'REQ8' function foobar
```

where the address of an external symbol is being referred to.

There are two possible solutions to work-around this issue:

- Rebuild all your objects/libraries.

  If you have any assembler files, you must check that all instructions preserve eight-byte stack alignment, and if necessary, correct them.

  For example, change:

  ```
  STMFD sp!, {r0-r3, lr} ; push an odd number of registers
  ```

  to

  ```
  STMFD sp!, {r0-r3, r12, lr} ; push even number of registers
  ```

  The assembler automatically marks the object with the `PRES8` attribute if all instructions preserve eight-byte stack alignment, so it is no longer necessary to add the `PRESERVE8` directive to the top of each assembler file.

- If you have any legacy objects/libraries that cannot be rebuilt, either because you do not have the source code, or because the old objects must not be rebuilt (for example, for qualification/certification reasons), then you must inspect the legacy objects to check whether they preserve eight-byte alignment or not.

  Use `fromelf -c` to disassemble the object code. C/C++ code compiled with ADS 1.1 or later normally preserves eight-byte alignment, but assembled code does not.

  If your objects do indeed preserve eight-byte alignment, then the linker error L6238E can be suppressed with the use of `--diag_suppress 6238` on the linker command line.

  By using this, you are effectively guaranteeing that these objects are `PRES8`.

  The linker warning L6306W is suppressible with `--diag_suppress 6306`.

See also *8 Byte Stack Alignment*, http://infocenter.arm.com/help/topic/com.arm.doc.faqs/ka4127.html.

L6239E `Cannot call non-interworking <t2> symbol '<sym>' in <obj2> from <t1> code in <obj1>(<sec1>)`

Example:

```
Cannot call non-interworking ARM symbol 'ArmFunc' in object foo.o from THUMB code in bar.o(.text)
```

This problem can be caused by `foo.c` not being compiled with the option `--apcs /interwork`, to enable ARM code to call Thumb code (and Thumb to ARM) by linker-generated interworking veneers.

L6241E `<objname>(<secname>) cannot use the address of '<attr1>' function <sym> as the image contains '<attr2>' functions.`

When linking with `'--strict'`, the linker reports conditions that might fail as errors, for example:

```
Error: L6241E: foo.o(.text) cannot use the address of '~IW' function main as the image contains 'IW' functions.
```

`IW` means *interworking*, and `~IW` means *non-interworking*.

L6242E `Cannot link object <objname> as its attributes are incompatible with the image attributes.`

Each object file generated by the compilation tools includes a set of attributes that indicates the options that it was built with. The linker checks the attributes of each object file it processes. If it finds attributes that are incompatible with those of object files it has loaded previously, it generates this error.

There are three common reasons for this error, each of which produces a different message:

- ```
  Error: L6242E: Cannot link object foo.o as its attributes are incompatible with the image attributes.
  require four-byte alignment of eight-byte datatypes clashes with require eight-byte alignment of eight-byte data types.
  ```

  This can occur when you try to link objects built using RVCT 2.0 or later with objects built using ADS or RVCT 1.2. In ADS and RVCT 1.2, `double` and `long long` data types were 4-byte aligned (unless you used the `-Oldrd` compiler option or the `__align` keyword). In RVCT 2.0, the ABI changed, so that `double` and `long long` data types are 8-byte aligned.
  ```

This change means that ADS and RVCT 1.2 objects and libraries using `double` or `long long` data types are not directly compatible with objects and libraries built using RVCT 2.0 or later, and so the linker reports an attribute clash.

To use RVCT 2.x or 3.0 C objects with legacy ADS C objects, compile the RVCT 2.x or 3.0 C code with the `--apcs /adsabi` command line option. This option was deprecated in RVCT 2.2 and removed from RVCT 3.1.

- `Error: L6242E: Cannot link object foo.o as its attributes are incompatible with the image attributes.`
  `... pure-endian double clashes with mixed-endian double.`

  This can occur when you are linking objects built using the ARM Compiler toolchain, RVCT or ADS with legacy SDT objects or objects built using either of the compiler options `--fpu softfpa` or `--fpu fpa`. SDT used a non-standard format for little-endian `double` and big-endian `long long`. However ADS and RVCT use industry-standard `double` and `long long` types, except for when the `--fpu softfpa` or `--fpu fpa` options are used. (These options are only supported in RVCT 2.1 and earlier). If you attempt to link object files that use the different formats for little-endian `double` and big-endian `long long` then the linker reports this error.

  ARM recommends you rebuild your entire project using RVCT or the ARM Compiler toolchain. If you do not have the source code for an object or library, then try recompiling your code with `--fpu softfpa`.
- `Error: L6242E: Cannot link object foo.o as its attributes are incompatible with the image attributes.`
  `... FPA clashes with VFP.`

  This error typically occurs when you attempt to link objects built with different `--fpu` options. ARM recommends you rebuild your entire project using the same `--fpu` options.

See *Are legacy objects and libraries compatible with my project?*, http://infocenter.arm.com/help/topic/com.arm.doc.faqs/ka3639.html.

L6243E `Selector only matches removed unused sections - no section to be FIRST/LAST.`

All sections matching this selector have been removed from the image because they were unused. For more information, `use --info unused`.

L6244E `<type> region <regionname> address (<addr>) not aligned on a <align> byte boundary.`

L6245E `Failed to create requested ZI section '<name>'.`

L6248E `<objname>(<secname>) in <attr1> region '<r1>' cannot have <rtype> relocation to <symname> in <attr2> region '<r2>'.`

This error can occur when you are trying to build position-independent (PI) code. Consider, for example the following code:

```
#include <stdio.h>
char *str = "test";
int main(void)
{
    printf ("%s",str);
}
```

When you compile and link this using:

```
armcc -c --apcs /ropi/rwpi pi.c
armlink --ropi --rwpi pi.o
```

the linker reports the following error message:

```
Error: L6248E: pi.o(.data) in PI region 'ER_RW' cannot have address type
relocation to .conststring in PI region 'ER_RO'.
```

This is because the compiler generates a global pointer `str` that must be initialized to the address of the string in the `.conststring` section. However, absolute addresses cannot be used in a PI system, so the link step fails.

To resolve this, you must re-write the code to avoid the explicit pointer. You can do this using either of the following methods:

- Use a global array instead of a global pointer, for example:

```
#include <stdio.h>
const char str[] = "test";
int main(void)
{
    printf ("%s",str);
}
```

- Use a local pointer instead of a global pointer, for example:

```
#include <stdio.h>
int main(void)
{
    char *str = "test";
    printf ("%s",str);
}
```

Note

If you are using an array of pointers, such as:

```
char * list[] = {"zero", "one", "two"};
```

the linker reports a separate error for each element in the array. In this case, ARM recommends you declare a two dimensional array for the list, with the first dimension as the number of elements in the array, and the second dimension as the maximum size of an element in the array, for example:

```
char list[3][5] = {"zero", "one", "two"};
```

You must change the `printf()` statement to, for example:

```
printf("%s", list[1]);
```

See compiler error number 1359.

L6249E `Entry point (<address>) lies within multiple sections.`

L6250E `Object <objname> contains illegal definition of special symbol <symbol>.`

L6251E `Object <objname> contains illegal reference to special symbol <symbol>.`

L6252E `Invalid argument for --xreffrom/--xrefto command: '<arg>'`

L6253E `Invalid SYMDEF address: <number>.`

L6254E `Invalid SYMDEF type : <type>.`

The content of the symdefs file is invalid.

See the following in *Using the Linker*:

- *Symdefs file format* on page 7-22.

L6255E `Could not delete file <filename>: <reason>`

An I/O error occurred while trying to delete the specified file. The file was either read-only, or was not found.

L6257E `<object>(<secname>) cannot be assigned to overlaid Execution region '<ername>'.`

This message indicates a problem with the scatter file.

See the following in the *Linker Reference*:

- Chapter 4 *Formal syntax of the scatter file*.

L6258E `Entry point (<address>) lies in an overlaid Execution region.`

This message indicates a problem with the scatter file.

See the following in the *Linker Reference*:

- Chapter 4 *Formal syntax of the scatter file*.

L6259E `Reserved Word '<name>' cannot be used as a <type> region name.`

`<name>` is a reserved word, so choose a different name for your region.

L6260E `Multiple load regions with the same name (<regionname>) are not allowed.`

This message indicates a problem with the scatter file.

See the following in the *Linker Reference*:

- Chapter 4 *Formal syntax of the scatter file*.

L6261E `Multiple execution regions with the same name (<regionname>) are not allowed.`

This message indicates a problem with the scatter file.

See the following in the *Linker Reference*:

- Chapter 4 *Formal syntax of the scatter file*.

L6263E `<addr> address of <regionname> cannot be addressed from <pi_or_abs> Region Table in <regtabregionname>`

The Region Table contains information used by the C-library initialization code to copy, decompress, or create ZI. This error message is given when the scatter file specifies an image structure that cannot be described by the Region Table.

The error message is most common when PI and non-PI Load Regions are mixed in the same image.

L6265E `Non-PI Section <obj>(<sec>) cannot be assigned to PI Exec region <er>.`

This might be caused by explicitly specifying the wrong ARM library on the linker command-line. Either:

- remove the explicit specification of the ARM library
- replace the library, for example, `c_t.l`, with the correct library.

L6266E `RWPI Section <obj>(<sec>) cannot be assigned to non-PI Exec region <er>.`

A file compiled with `--apcs=/rwpi` is placed in an Execution Region that does not have the PI attribute.

L6271E `Two or more mutually exclusive attributes specified for Load region <regname>`

This message indicates a problem with the scatter file.

L6272E `Two or more mutually exclusive attributes specified for Execution region <regname>`

This message indicates a problem with the scatter file.

L6273E `Section <objname>(<secname>) has mutually exclusive attributes (READONLY and ZI)`

This message indicates a problem with the object file.

L6275E `COMMON section <obj1>(<sec1>) does not define <sym> (defined in <obj2>(<sec2>))`

Given a set of `COMMON` sections with the same name, the linker selects one of them to be added to the image and discards all others. The selected `COMMON` section must define all the symbols defined by any rejected `COMMON` section, otherwise, a symbol that was defined by a rejected section would become undefined again. The linker generates an error if the selected copy does not define a symbol that a rejected copy does. This error is normally caused by a compiler fault. Contact your supplier.

L6276E `Address <addr> marked both as <s1>(from <sp1>(<obj1>) via <src1>) and <s2>(from <sp2>(<obj2>) via <src2>).`

The image cannot contain contradictory mapping symbols for a given address, because the contents of each word in the image are uniquely typed as ARM (`$a`) or THUMB (`$t`) code, DATA (`$d`), or NUMBER. It is not possible for a word to be both ARM code and DATA. This might indicate a compiler fault. Contact your supplier.

L6277E `Unknown command '<cmd>'.`

L6278E `Missing expected <str>.`

L6279E `Ambiguous selectors found for <sym> ('<sel1>' and '<sel2>').`

L6280E `Cannot rename <sym> using the given patterns.`

See the following in the *Linker Reference*:

- *RENAME on page 3-5*.

L6281E `Cannot rename both <sym1> and <sym2> to <newname>.`

See the following in the *Linker Reference*:

- *RENAME on page 3-5*.

L6282E `Cannot rename <sym> to <newname> as a global symbol of that name exists (defined) in <obj>).`

See the following in the *Linker Reference*:

- *RENAME on page 3-5*.

L6283E `Object <objname> contains illegal local reference to symbol <symbolname>.`

An object cannot contain a reference to a local symbol, because local symbols are always defined within the object itself.

L6285E `Non-relocatable Load region <lr_name> contains R-Type dynamic relocations. First R-Type dynamic relocation found in <object>(<secname>) at offset 0x<offset>.`

This error occurs where there is a PI reference between two separate segments, if the two segments can be moved apart at runtime. When the linker sees that the two sections can be moved apart at runtime it generates a relocation (an R-Type relocation) that can be resolved if the sections are moved from their statically linked address. However the linker faults this relocation (giving error L6285E) because PI regions must not have relocations with respect to other sections as this invalidates the criteria for being position independent.

L6286E `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) with respect to {symname|%s}. Value(<val>) out of range(<range>) for (<rtype>)`

This error typically occurs in the following situations:

- In handwritten assembler code, where the limited number of bits for a field within the instruction opcode is too small to refer to a far away symbol.

  For example, an LDR or STR instruction where the offset is too large for the instruction (+/-4095 for ARM state LDR/STR instruction).

- When the linker is having difficulty placing veneers around a large code section in your image.

  When the linker places a veneer near a very large section it must decide whether to place the veneer before or after the large section. When the linker has placed the veneer it might have to place more veneers, which could be placed between the original veneer and its target. This would increase the distance between the veneer and its target.

  The linker automatically allows for modest increases in distances between veneers and their targets. However, a large number of veneers placed between the original veneer and its target might result in the target moving out of range. If this occurs the linker generates message L6286E.

  To work around this you can move large code sections away from areas where the linker is placing many veneers. This can be done either by placing large sections in their own regions or by placing them first in the region they are located in using the +FIRST directive in the scatter-loading description file.

  For example:

```
LOAD 0x0A000000 0x1000000
{
   ROM1 +0x0
   {
      *(+RO)
   }
}
```

  Can be changed to:

```
LOAD 0x0A000000 0x1000000
{
   ROM1 +0x0
   {
      *(+RO)
   }
   ROM1A +0x0
   {
      large.o (+RO)
   }
}
```

- When .ARM.exidx exception-handling index tables are placed in different execution regions, or too far from exception handling code.

  The .ARM.exidx exception-handling index tables must be located in a single execution region. Also, the distance from the these tables to the C++ code that uses C++ exception handling must be within the range -0x40000000 to 0x3fffffff. Otherwise, the linker reports the following error:

```
L6286: Value(0x9ff38980) out of range(-0x40000000 - 0x3fffffff) for
relocation #0 (R_ARM_PREL31), wrt symbol xxx in XXXX.o(.ARM.exidx)
```

  This behavior is specified in the *ARM Exception Handling ABI* (EHABI). The EHABI states that the R_ARM_PREL31 relocation, which .ARM.exidx uses, does not use the highest bit (bit 31) for calculating the relocation.

  The most likely cause of this is that C++ code that must access the .ARM.exidx sections, has been split and placed into separate execution regions, outside of the valid range (-0x40000000 - 0x3fffffff).

To resolve this error, if you have memory between the separated execution regions, place the .ARM.exidx section there with the selector *(.ARM.exidx). For example:

```
LOAD_ROM 0x00000000
{
   ER1 0x00000000 ;the distance from ER2 to ER1 is out of range
   {
      file1.o (+RO) ; from a C++ source
      * (+RO)
   }
   ERx 0x30000000
   {
      *(.ARM.exidx) ; ARM.exidx to ER1 and ER2 both in range
   }
   ER2 0x60000000
   {
      file2.o (+RO) ; from a C++ source
   }
   ER3 +0
   {
      * (+RW, +ZI)
   }
```

Otherwise, try placing the code into an execution region close enough to the tables (within the range of -0x40000000 - 0x3fffffff).

In other cases, make sure you have the latest patch installed from *Downloads*, https://silver.arm.com/browse.

For more information see *Value out of range for relocation*, http://infocenter.arm.com/help/topic/com.arm.doc.faqs/ka3553.html.

L6287E `Illegal alignment constraint (<align>) specified for <objname>(<secname>).`

An illegal alignment was specified for an ELF object.

L6291E `Cannot assign Fixed Execution Region <ername> Load Address:<addr>. Load Address must be greater than or equal to next available Load Address:<load_addr>.`

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*.

L6292E `Ignoring unknown attribute '<attr>' specified for region <regname>.`

This error message is specific to execution regions with the FIXED attribute. FIXED means make the load address the same as the execution address. The linker can only do this if the execution address is greater than or equal to the next available load address within the load region.

See the following in *Using the Linker*:

- *Using the FIXED attribute to create root regions* on page 8-17.

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*.

L6294E `<type> region <regionname> spans beyond 32 bit address space (base <base>, size <size> bytes).`

This error message relates to a problem with the scatter file.

L6295E `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) with respect to <symname> SBREL relocation requires image to be RWPI`

L6296E `Definition of special symbol <sym1> is illegal as symbol <sym2> is absolute.`

See L6188E.

L6300W `Common section <object1>(<section1>) is larger than its definition <object2>(<section2>).`

This might indicate a compiler fault. Contact your supplier.

L6301W `Could not find file <filename>: <reason>`

The specified file was not found in the default directories.

L6302W `Ignoring multiple SHLNAME entry.`

There can be only one `SHLNAME` entry in an edit file. Only the first such entry is accepted by the linker. All subsequent `SHLNAME` entries are ignored.

L6304W `Duplicate input file <filename> ignored.`

The specified filename occurred more than once in the list of input files.

L6305W `Image does not have an entry point. (Not specified or not set due to multiple choices.)`

The entry point for the ELF image was either not specified, or was not set because there was more than one section with an entry point linked-in. You must use linker option `--entry` to specify the single, unique entry, for example:

```
--entry 0x0
```

or

```
--entry <label>
```

The label form is typical for an embedded system.

L6306W `'<attr1>' section <objname>(<secname>) should not use the address of '<attr2>' function <sym>.`

See L6238E.

L6307W `Relocation #<rel_class>:<rel_num> in <objname>(<secname>) with respect to <sym>. Branch to unaligned destination.`

L6308W `Could not find any object matching <membername> in library <libraryname>.`

The name of an object in a library is specified on the link-line, but the library does not contain an object with that name.

L6309W `Library <libraryname> does not contain any members.`

A library is specified on the linker command-line, but the library does not contain any members.

L6310W `Unable to find ARM libraries.`

This is most often caused by incorrect arguments to `--libpath` or an invalid value for the environment variable `ARMCC`*`n`*`LIB`, if defined.

Set the the correct path with either the `--libpath` linker option or the `ARMCC`*`n`*`LIB` environment variable. The default path for a Windows installation is:

`install_directory``\lib`

Ensure this path does not include:

- `\armlib`
- `\cpplib`

- any trailing slashes (\) at the end. These are added by the linker automatically.

Use `--verbose` or `--info libraries` to display where the linker is attempting to locate the libraries.

See the following in the *Linker Reference*:

- *--info=topic[,topic,...] on page 2-79*
- *--libpath=pathlist on page 2-96*
- *--verbose on page 2-183*.

See the following in *Introducing the ARM Compiler toolchain*:

- *Toolchain environment variables* on page 2-15.

L6311W `Undefined symbol <symbol> (referred from <objname>).`

See L6218E.

L6312W `Empty <type> region description for region <region>`

L6313W `Using <oldname> as an section selector is obsolete. Please use <newname> instead.`

For example, use of `IWV$$Code` within the scatter file is obsolete. Replace `IWV$$Code` with `Veneer$$Code`.

L6314W `No section matches pattern <module>(<section>).`

Example:

```
No section matches pattern foo.*o(ZI).
```

This can occur for the following reasons:

- The file `foo.o` is mentioned in your scatter file, but it is not listed on the linker command-line. To resolve this, add `foo.o` to the link-line.
- You are trying to place the ZI data of `foo.o` using a scatter file, but `foo.o` does not contain any ZI data. To resolve this, remove the `+ZI` attribute from the foo.o line in your scatter file.
- You have used `__attribute__((at(`*address*`)))` in your source code to place code and data at a specific address. You have also specified `*(.ARM.__AT_`*address*`)` in a scatter file, but you have not specified the address as eight hexadecimal digits. For example, if you specify `__attribute__((at(0x10000)))` in your source code, then you must specify the section name as `*(.ARM.__AT_0x00010000)` in the scatter file.

See the following in *Using the Linker*:

- *Placing functions and data at specific addresses* on page 8-18
- *Using __at sections to place sections at a specific address* on page 8-37.

L6315W `Ignoring multiple Build Attribute symbols in Object <objname>.`

An object can contain at most one absolute `BuildAttribute$$...` symbol. Only the first such symbol from the object symbol table is accepted by the linker. All subsequent ones are ignored.

L6316W `Ignoring multiple Build Attribute symbols in Object <objname> for section <sec_no>`

An object can contain at most one `BuildAttribute$$...` symbol applicable to a given section. Only the first such symbol from the object symbol table is accepted by the linker. All subsequent ones are ignored.

L6317W `<objname>(<secname>) should not use the address of '<attr1>' function <sym> as the image contains '<attr2>' functions.`

L6318W `<objname>(<secname>) contains branch to a non-code symbol <sym>.`

This warning means that in the (usually assembler) file, there is a branch to a non-code symbol (in another `AREA`) in the same file. This is most likely a branch to a label or address where there is data, not code.

For example:

```
AREA foo, CODE
B bar
AREA bar, DATA
DCD 0
END
```

This results in the message:

`init.o(foo) contains branch to a non-code symbol bar.`

If the destination has no name:

```
BL 0x200 ; Branch with link to 0x200 bytes ahead of PC
```

the following message is displayed:

`bootsys.o(BOOTSYS_IVT) contains branch to a non-code symbol <Anonymous Symbol>.`

This warning can also appear when linking objects generated by GCC. GCC uses linker relocations for references internal to each object. The targets of these relocations might not have appropriate mapping symbols that permit the linker to determine whether the target is code or data, so a warning is generated. By contrast, `armcc` resolves all such references at compile-time.

L6319W `Ignoring <cmd> command. Cannot find section <objname>(<secname>).`

For example, when building a Linux application, you might have:

`--keep *(.init_array)`

on the linker command-line in your makefile, but this section might not be present when building with no C++, in which case this warning is reported:

`Ignoring --keep command. Cannot find section *(.init_array)`

You can often ignore this warning, or suppress it with `--diag_suppress 6319`.

L6320W `Ignoring <cmd> command. Cannot find argument '<argname>'.`

L6323W `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) with respect to <sym>. Multiple variants exist. Using the <type> variant to resolve ambiguity`

L6324W `Ignoring <attr> attribute specified for Load region <regname>.`

This attribute is applicable to execution regions only. If specified for a Load region, the linker ignores it.

L6325W `Ignoring <attr> attribute for region <regname> which uses the +offset form of base address.`

This attribute is not applicable to regions using the `+offset` form of base address. If specified for a region, which uses the `+offset` form, the linker ignores it.

A region that uses the `+offset` form of base address inherits the `PI`, `RELOC`, or `OVERLAY` attributes from either:

- the previous region in the description
- the parent load region if it is the first execution region in the load region.

See the following in the *Linker Reference*:

- *Inheritance rules for load region address attributes on page 4-19*
- *Inheritance rules for execution region address attributes on page 4-20*
- *Inheritance rules for the RELOC address attribute on page 4-21*.

L6326W `Ignoring ZEROPAD attribute for non-root execution region <ername>.`

`ZEROPAD` only applies to root execution regions. A root region is a region whose execution address is the same as its load address, and so does not require moving or copying at run-time.

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*.

L6329W `Pattern <module>(<section>) only matches removed unused sections.`

All sections matching this pattern have been removed from the image because they were unused. For more information, use `--info unused`.

See the following in *Using the Linker*:

- *Elimination of unused sections* on page 5-4.

See the following in the *Linker Reference*:

- *--info=topic[,topic,...] on page 2-79*.

L6330W `Undefined symbol <symbol> (referred from <objname>). Unused section has been removed.`

This means that an unused section has had its base and limit symbols referenced. For more information, use `--info unused`.

For example, when using a scatter file to place code and data with RVCT 2.1 or later, the linker reports this error if the scatter file includes the linker-generated table `ZISection$$Table`. In RVCT 2.1, a new region tables format was introduced which no longer contains `ZISection$$Table`.

See the following in *Using the Linker*:

- *Elimination of unused sections* on page 5-4.

See the following in the *Linker Reference*:

- *--info=topic[,topic,...] on page 2-79*.

L6331W `No eligible global symbol matches pattern <pat>.`

L6332W `Undefined symbol <sym1> (referred from <obj1>). Resolved to symbol <sym2>.`

L6334W `Overalignment <overalignment> for region <regname> cannot be negative.`

See the following in *Using the Linker*:

- *Overalignment of execution regions and input sections* on page 8-58.

L6335W `ARM interworking code in <objname>(<secname>) may contain invalid tailcalls to ARM non-interworking code.`

The compiler is able to perform tailcall optimization for improved code size and performance. However, there is a problematic sequence for Architecture 4T code where a Thumb IW function calls (by a veneer) an ARM IW function, which tailcalls an ARM not-IW function. The return from the ARM not-IW function can pop the return address off the stack into the PC instead of using the correct `BX` instruction. The linker can detect this situation and report this warning.

Thumb IW tailcalls to Thumb not-IW do not occur because Thumb tailcalls with `B` are so short ranged that they can only be generated to functions in the same ELF section which must also be Thumb.

The warning is pessimistic in that an object *might* contain invalid tailcalls, but the linker cannot be sure because it only looks at the attributes of the objects, not at the contents of their sections.

To avoid the warning, either recompile your entire code base, including any user libraries, with `--apcs /interwork`, or manually inspect the ARM IW function to check for tailcalls (that is, where function calls are made using an ordinary branch `B` instruction), to check whether this is a real problem. This warning can be suppressed with `--diag_suppress L6335W`.

L6337W `Common code sections <o1>(<s1>) and <o2>(<s2>) have incompatible floating-point linkage`

L6339W `Ignoring RELOC attribute for execution region <er_name>.`

Execution regions cannot explicitly be given `RELOC` attribute. They can only gain this attribute by inheriting from the parent load region or the previous execution region if using the `+offset` form of addressing.

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*.

L6340W `options first and last are ignored for link type of <linktype>`

The `--first` and `--last` options are meaningless when creating a partially-linked object.

L6366E `<object> attributes<attr> are not compatible with the provided cpu and fpu attributes<cli> <diff>.`

L6367E `<object>(<section>) attributes<attr> are not compatible with the provided cpu and fpu attributes<cli> <diff>`

L6368E `<symbol> defined in <object>(<section>) attributes<attr> are not compatible with the provided cpu and fpu attributes<cli> <diff>`

L6369E `<symbol> defined in <object>(ABSOLUTE) are not compatible with the provided cpu and fpu Attributes<cli> <diff>`

L6370E `cpu <cpu> is not compatible with fpu <fpu>`

See the following in the *Linker Reference*:

- *--cpu=name on page 2-38*
- *--fpu=name on page 2-75*.

L6371E `Adding attributes from cpu and fpu: <attrs>`

L6372E `Image needs at least one load region.`

L6373E `libattrs.map file not found in System Library directory <dir>. Library selection may be impaired.`

L6384E `No Load Execution Region of name <region> seen yet at line <line>.`

This might be because you have used the current base address in a limit calculation in a scatter file. For example:

```
ER_foo 0 ImageBase(ER_foo)
```

L6385W `Addition overflow on line <line>`

L6386E `Exec Region Expressions can only be used in base address calculations on line <line>`

L6387E `Load Region Expressions can only be used in ScatterAssert expressions on line <line>`

See the following in the *Linker Reference*:

- *ScatterAssert function and load address related functions on page 4-39*.

L6388E `ScatterAssert expression <expr> failed on line <line>`

See the following in the *Linker Reference*:

- *ScatterAssert function and load address related functions on page 4-39*.

L6389E `Load Region <name> on line <line> not yet complete, cannot use operations that depend on length of region`

L6390E `Conditional operator (expr) ? (expr) : (expr) on line <line> has no : (expr).`

See the following in the *Linker Reference*:

- *About Expression evaluation in scatter files on page 4-31*
- *Expression rules in scatter files on page 4-33*.

L6404W `FILL value preferred to combination of EMPTY, ZEROPAD and PADVALUE for Execution Region <name>.`

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*.

L6405W `No .ANY selector matches Section <name>(<objname>).`

See the following in *Using the Linker*:

- *Placing unassigned sections with the .ANY module selector* on page 8-25.

L6406W `No space in execution regions with .ANY selector matching Section <name>(<objname>).`

This occurs if there is not sufficient space in the scatter file regions containing `.ANY` to place the section listed. You must modify your scatter file to ensure there is sufficient space for the section.

See the following in *Using the Linker*:

- *Placing unassigned sections with the .ANY module selector* on page 8-25.

L6407W `Sections of aggregate size 0x<size> bytes could not fit into .ANY selector(s).`

This warning identifies the total amount of image data that cannot be placed in any `.ANY` selectors.

For example, `.ANY(+ZI)` is placed in an execution region that is too small for the amount of `ZI` data:

```
ROM_LOAD 0x8000
{
    ROM_EXEC 0x8000
    {
        .ANY(+RO,+RW)
    }
    RAM +0 0x{...} <<< region max length is too small
    {
        .ANY(+ZI)
    }
}
```

See the following in *Using the Linker*:

- *Placing unassigned sections with the .ANY module selector* on page 8-25.

L6408W `Output is --fpic yet section <sec> from <obj> has no FPIC attribute.`

L6409W `Output is --fpic yet object <obj> has no FPIC attribute.`

L6410W Symbol <sym> with non STV_DEFAULT visibility <vis> should be resolved statically, cannot use definition in <lib>.

L6411W No compatible library exists with a definition of startup symbol <name>.

L6412W Disabling merging for section <sec> from object <obj>, non R_ARM_ABS32 relocation from section <srcsec> from object <srcobj>

L6413W Disabling merging for section <sec> from object <obj>, Section contains misaligned string(s).

L6414E --ropi used without --rwpi or --rw-base.

See the following in the *Linker Reference*:

- *--ropi on page 2-135*
- *--rw_base=address on page 2-138*
- *--rwpi on page 2-139*.

L6415E Could not find a unique set of libraries compatible with this image. Suggest using the --cpu option to select a specific library.

See the following in the *Linker Reference*:

- *--cpu=name on page 2-38*.

L6416E Relocation <type> at <relclass>:<idx> <objname>(<secname>) cannot be veneered as it has an offset <offset> from its target.

L6417W Relocation #<rel_class>:<rel_number> in <objname>(<secname>) is with respect to a reserved tagging symbol(#<idx>).

L6418W Tagging symbol <symname> defined in <objname>(<secname>) is not recognized.

L6419W Undefined symbol <symbol> (referred from <objname>) imported.

L6420E Ignoring <oepname>(<secname>:<secnum>) as it is not of a recognized type.

L6422U PLT generation requires an architecture with ARM instruction support.

For the linker to generate PLT, you must be using a target that supports the ARM instruction set. For example, the linker cannot generate PLT for a Cortex-M3 target.

L6423E Within the same collection, section <secname> cannot have different sort attributes.

L6424E Within the same collection, section <secname1> and section <secname2> cannot be separated into different execution regions.

L6425E Within the same collection, section <secname> cannot have their section names with different length.

L6426E Within the same collection, section <secname> cannot have its name duplicated.

L6427E Cannot rename <sym> to <newname> as it has already been renamed to <name>).

L6429U Attempt to set maximum number of open files to <val> failed with error code <error>.

An attempt to increase the number of file handles armlink can keep open at any one time has failed.

L6431W Ignoring incompatible enum size attribute on Symbol <symbol> defined in <object>(<section>).

L6432W Ignoring incompatible enum size attribute on Object <object>(<section>).

L6433W Ignoring incompatible enum size attribute on object <object>.

L6434W Ignoring incompatible wchar_t size attribute on Symbol <symbol> defined in <object>(<section>).

L6435W Ignoring incompatible wchar_t size attribute on Section <object>(<section>).

L6436W Ignoring incompatible wchar_t size attribute on object <object>.

L6437W Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <armsym>. Branch relocation to untyped symol in object <armobjname>, target state unknown.

L6438E __AT section <objname>(<secname>) address <address> must be at least 4 byte aligned.

L6439W Multiply defined Global Symbol <sym> defined in <objname>(<secname>) rejected in favour of Symbol defined in <selobj>(<selsec>).

L6440E Unexpected failure in link-time code generation

L6441U System call to get maximum number of open files failed <error>.

L6442U Linker requires a minimum of <min> open files, current system limit is <max> files.

L6443W Data Compression for region <region> turned off. Region contains reference to symbol <symname> which depends on a compressed address.

The linker requires the contents of a region to be fixed before it can be compressed and cannot modify it after it has been compressed. Therefore a compressible region cannot refer to a memory location that depends on the compression process.

L6444I symbol visibility : <symname> set to <visibility>.

L6445I symbol visibility : <symname> merged to <set_vis> from existing <old_vis> and new <new_vis>.

L6447E SHT_PREINIT_ARRAY sections are not permitted in shared objects.

L6448W While processing <filename>: <message>

L6449E While processing <filename>: <message>

L6450U Cannot find library <libname>.

L6451E <object> built permitting Thumb is forbidden in an ARM-only link.

L6452E <object>(<section>) built permitting Thumb is forbidden in an ARM-only link.

L6453E <symbol> defined in <object>(<section>) built permitting Thumb is forbidden in an ARM-only link.

L6454E <symbol> defined in <object>(ABSOLUTE) built permitting Thumb is forbidden in an ARM-only link.

L6455E Symbol <symbolname> has deprecated ARM/Thumb Synonym definitions (by <object1> and <object2>).

L6459U Could not create temporary file.

L6462E Reference to <sym> from a shared library only matches a definition with Hidden or Protected Visibility in Object <obj>.

L6463U Input Objects contain <archtype> instructions but could not find valid target for <archtype> architecture based on object attributes. Suggest using --cpu option to select a specific cpu.

See the following in the *Linker Reference*:

- *--cpu=name on page 2-38*.

L6464E Only one of --dynamic_debug, --emit-relocs and --emit-debug-overlay-relocs can be selected.

See the following in the *Linker Reference*:

- *--dynamic_debug on page 2-50*
- *--emit_debug_overlay_relocs on page 2-54*
- *--emit_relocs on page 2-57*.

L6467W Library reports remark: <msg>

L6468U Only --pltgot=direct or --pltgot=none supported for --base_platform with multiple Load Regions containing code.

See the following in the *Linker Reference*:

- *--base_platform on page 2-18*
- *--pltgot=type on page 2-120*.

L6469E --base_platform does not support RELOC Load Regions containing non RELOC Execution Regions. Please use +0 for the Base Address of Execution Region <ername> in Load Region <lrname>.

See the following in the *Linker Reference*:

- *--base_platform on page 2-18*
- *Inheritance rules for the RELOC address attribute on page 4-21*.

L6470E PLT section <secname> cannot be moved outside Load Region <lrname>.

L6471E Branch Relocation <rel_class>:<idx> in section <secname> from object <objname> refers to ARM Absolute <armsym> symbol from object <armobjname>, Suppress error to treat as a Thumb address.

Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <armsym>. Branch refers to ARM Absolute Symbol defined in <armobjname>, Suppress error to treat as a Thumb address.

L6475W IMPORT/EXPORT commands ignored when --override_visibility is not specified

The symbol you are trying to export, either with an EXPORT command in a steering file or with the --undefined_and_export command-line option, is not exported becuase of low visibility.

See the following in the *Linker Reference*:

- *--override_visibility on page 2-114*
- *--undefined_and_export=symbol on page 2-174*
- *EXPORT on page 3-2*.

L6616E Cannot increase size of RegionTable <sec_name> from <obj_name>

L6617E Cannot increase size of ZISectionTable <sec_name> from <obj_name>

L6629E Unmatched parentheses expecting) but found <character> at position <col> on line <line>

This message indicates a parsing error in the scatter file.

L6630E `Invalid token start expected number or ( but found <character> at position <col> on line <line>`

This message indicates a parsing error in the scatter file.

L6631E `Division by zero on line <line>`

This message indicates an expression evaluation error in the scatter file.

L6632W `Subtraction underflow on line <line>`

This message indicates an expression evaluation error in the scatter file.

L6634E `Pre-processor command in  '<filename>'too long, maximum length of <max_size>`

This message indicates a problem with pre-processing the scatter file.

L6635E `Could not open intermediate file '<filename>' produced by pre-processor: <reason>`

This message indicates a problem with pre-processing the scatter file.

L6636E `Pre-processor step failed for '<filename>'`

This message indicates a problem with pre-processing the scatter file.

L6637W `No input objects specified. At least one input object or library(object) must be specified.`

At least one input object or library(object) must be specified.

L6638U `Object <objname> has a link order dependency cycle, check sections with SHF_LINK_ORDER`

L6640E `PDTTable section not least static data address, least static data section is <secname>`

Systems that implement shared libraries with RWPI use a *process data table* (PDT). It is created at static link time by the linker and must be placed first in the data area of the image.

This message indicates that the scatter file does not permit placing the PDT first in the data area of the image.

To avoid the message, adjust your scatter file so that the PDT is placed correctly. This message can also be triggered if you accidentally build object files with `--apcs=/rwpi`.

L6642W `Unused virtual function elimination might not work correctly, because <obj_name> has not been compiled with --vfe`

L6643E `The virtual function elimination information in section <sectionname> refers to the wrong section.`

This message might indicate a compiler fault. Contact your supplier.

L6644E `Unexpectedly reached the end of the buffer when reading the virtual function elimination information in section <oepname>(<sectionname>).`

This message might indicate a compiler fault. Contact your supplier.

L6645E `The virtual function elimination information in section <oepname>(<sectionname>) is incorrect: there should be a relocation at offset <offset>.`

This message might indicate a compiler fault. Contact your supplier.

L6646W `The virtual function elimination information in section <oepname>(<sectionname>) contains garbage from offset <offset> onwards.`

This message might indicate a compiler fault. Contact your supplier.

L6647E `The virtual function elimination information for <vcall_objectname>(<vcall_sectionname>) incorrectly indicates that section <curr_sectionname>(object <curr_objectname>), offset <offset> is a relocation (to a virtual function or RTTI), but there is no relocation at that offset.`

This message might indicate a compiler fault. Contact your supplier.

L6649E `EMPTY region <regname> must have a maximum size.`

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*.

L6650E `Object <objname> Group section <sectionidx> contains invalid symbol index <symidx>.`

L6651E `Section <secname> from object <objname> has SHF_GROUP flag but is not member of any group.`

L6652E `Cannot reverse Byte Order of Data Sections, input objects are <inputendian> requested data byte order is <dataendian>.`

L6654E `Rejected Local symbol <symname> referred to from a non group member <objname>(<nongrpname>)`

This message might indicate a compiler fault. Contact your supplier.

L6656E `Internal error: the vfe section list contains a non-vfe section called <oepname>(<secname>).`

This message might indicate a compiler fault. Contact your supplier.

L6664W `Relocation #<rel_class>:<rel_number> in <objname>(<secname>) is with respect to a symbol(#<idx> before last Map Symbol #<last>).`

L6665W `Neither Lib$$Request$$armlib Lib$$Request$$cpplib defined, not searching ARM libraries.`

This reproduces the warning:

```
  AREA Block, CODE, READONLY
  EXPORT func1
  ;IMPORT || Lib$$Request$$armlib||
  IMPORT printf
func1
  LDR r0,=string
  BL printf
  BX lr
  AREA BlockData, DATA
string DCB "mystring"
  END
```

The linker has not been told to look in the libraries and so cannot find the symbol `printf`.

This also causes an error:

```
L6218E: Undefined symbol printf (referred from L6665W.o).
```

If you do not want the libraries, then ignore this message. Otherwise, to fix both the error and the warning uncomment the line:

```
IMPORT || Lib$$Request$$armlib||
```

L6679W Data in output ELF section #<sec> '<secname>' was not suitable for compression (<data_size> bytes to <compressed_size> bytes).

L6682E Merge Section <oepname>(<spname>) is a code section

L6683E Merge Section <oepname>(<spname>) has an element size of zero

L6684E Section <spname> from object <oepname> has SHF_STRINGS flag but not SHF_MERGE flag

L6685E Section <spname> from object <oepname> has a branch reloc <rel_idx> to a SHF_MERGE section

L6688U Relocation #<rel_class>:<rel_idx> in <oepname>(<spname>) references a negative element

L6689U Relocation #<rel_class>:<rel_idx> in <oepname>(<spname>). Destination is in the middle of a multibyte character

L6690U Merge Section <spname> from object <oepname> has no symbols

L6703W Section <er> implicitly marked as non-compressible.

L6707E Padding value not specified with PADVALUE attribute for execution region <regionname>.

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*.

L6708E Could not process debug frame from <secname> from object <oepname>.

L6709E Could not associate fde from <secname> from object <oepname>.

L6713W Function at offset <offset> in <oepname>(<secname>) has no symbol.

L6714W Exception index table section .ARM.exidx from object <oepname> has no data.

L6720U Exception table <spname> from object <oepname> present in image, --noexceptions specified.

See the following in the *Linker Reference*:

- *--exceptions, --no_exceptions on page 2-60*.

L6721E Section #<secnum> '<secname>' in <oepname> is not recognized and cannot be processed generically.

L6725W Unused virtual function elimination might not work correctly, because there are dynamic relocations.

L6728U Link order dependency on invalid section number <to> from section number <from>.

L6730W Relocation #<rel_class>:<index> in <objname>(<secname>) with respect to <name>. Symbol has ABI type <type>, legacy type <legacy_type>.

A change in the linker behavior gives warnings about strict compliance with the ABI.

Note

The following example produces a warning only if linking with a toolchain that is compliant with an earlier version of the *Application Binary Interface* (ABI). The ARM Compiler toolchain v4.1 and later does not give this warning.

Example:

```
    AREA foo, CODE, READONLY
    CODE32
    ENTRY
    KEEP
func proc
    NOP
    ENDP
    DCD foo
    END
```

The warning is related to how the assembler marks sections for interworking. Previously, the section symbol `foo` would be marked as ARM or Thumb code in the ELF file. The `DCD foo` in this example would therefore also be marked as subject to interworking.

However, the ABI specifies that only functions are subject to interworking and marked as ARM or Thumb. The linker therefore warns that it is expecting `DCD <number>`, which does not match the symbol type (ARM, or Thumb if you use `CODE16`) of the area section.

The simplest solution is to move the data into a separate data area in the assembly source file.

Alternatively, you can use `--diag_suppress 6730` to suppress this warning.

L6731W `Unused virtual function elimination might not work correctly, because the section referred to from <secname> does not exist.`

L6733W `<objname>(<secname>) contains offset relocation from <lr1name> to <lr2name>, load regions must be rigidly relative.`

L6738E `Relocation #<rel_class>:<relocnum> in <oepname>(<secname>) with respect to <wrtsym> is a GOT-relative relocation, but _GLOBAL_OFFSET_TABLE_ is undefined.`

Some GNU produced images can refer to the symbol named `_GLOBAL_OFFSET_TABLE_`. If there are no `GOT Slot` generating relocations and the linker is unable to pick a suitable address for the `GOT base` the linker issues this error message.

L6739E `Version '<vername>' has a dependency to undefined version '<depname>'.`

L6740W `Symbol '<symname>' versioned '<vername>' defined in '<symverscr>' but not found in any input object.`

L6741E `Versioned symbol binding should be 'local:' or 'global:'.`

L6742E `Symbol '<symname>' defined by '<oepname>'.  Cannot not match to default version symbol '<defversym>'`

L6743E `Relocation #<rel_class>:<index> in <oepname>(<spname>) with respect to <symname> that has an alternate def. Internal consistency check failed`

L6744E `Relocation #<rel_class>:<index> <oepname>(<spname>) with respect to undefined symbol <symname>. Internal consistency check:`

L6745E `Target CPU <cpuname> does not Support ARM, <objname>(<secname>) contains ARM code`

L6747W `Raising target architecture from <oldversion> to <newversion>.`

If the linker detects objects that specify the obsolete ARMv3, it upgrades these to ARMv4 to be usable with ARM libraries.

L6748U `Missing dynamic array, symbol table or string table in file <oepname>.`

L6751E No such sorting algorithm <str> available.

L6753E CallTree sorting needs Entry Point to lie within a CallTree Sort ER.

L6761E Removing symbol <symname>.

L6762E Cannot build '<type>' PLT entries when building a <imgtype>.

L6763W '<optname>' cannot be used when building a shared object or DLL. Switching it off

L6764E Cannot create a PLT entry for target architecture 4T that calls Thumb symbol <symname>

L6765W Shared object entry points must be ARM-state when linking architecture 4T objects.

This can occur when linking with GNU C libraries. The GNU startup code `crt1.o` does not have any build attributes for the entry point, so the linker cannot determine which execution state (ARM or Thumb) the code runs in. Because the GNU C library startup code is ARM code, you can safely ignore this warning, or suppress it with `--diag_suppress 6765`.

L6766W PLT entries for architecture 4T do not support incremental linking.

L6769E Relocation #<rel_class>:<relocnum> in <oepname>(<secname>) with respect to <wrtsym>. No GOTSLOTexists for symbol.

L6770E The size and content of the dynamic array changed too late to be fixed.

L6771W <oepname>(<secname>) contains one or more address-type relocations in RO data. Making section RW to be dynamically relocated at run-time.

L6772W IMPORT <symname> command ignored when building --sysv.

See the following in the *Linker Reference*:

- *--sysv on page 2-169*
- *IMPORT on page 3-4*.

L6774W <objname>(<secname>) has debug frame entries of a bad length.

L6775W <objname>(<secname>) has FDEs which use CIEs which are not in this section.

L6776W The debug frame in <objname>(<secname>) does not describe an executable section.

L6777W The debug frame in <objname>(<secname>) has <actual> relocations (expected <expected>)

L6778W The debug frame in <objname>(<secname>) uses 64-bit DWARF.

L6780W <origvis> visibility removed from symbol '<symname>' through <impexp>.

L6781E Value(<val>) Cannot be represented by partition number <part> for relocation #<rel_class>:<rel_number> (<rtype>, wrt symbol <symname>) in <objname>(<secname>)

Relocation #<rel_class>:<rel_number> in <objname>(<secname>) with respect to <symname>. Value(<val>) Cannot be represented by partition number <part> for relocation type >rtype>

L6782W Relocation #<rel_class>:<relnum> '<rtype>' in <oepname> may not access data correctly alongside <pltgot_type> PLT entries

L6783E `Mapping symbol #<symnum> '<msym>' in <oepname>(<secnum>:<secname>) defined at the end of, or beyond, the section size (symbol offset=0x<moffset>, section size=0x<secsize>)`

This indicates that the address for a section points to a location at the end of or outside of the ELF section. This can be caused by an empty inlined data section and indicates there might be a problem with the object file. You can use `--diag_warning 6783` to suppress this error.

L6784E `Symbol #<symnum> '<symname>' in <oepname>(<secnum>:<secname>) with value <value> has size 0x<size> that extends to outside the section.`

The linker encountered a symbol with a size that extends outside of its containing section. This message is only a warning by default in the RVCT 2.2 SP1 and later toolchains. Use `--diag_warning 6784` to suppress this error.

L6785U `Symbol '<symname>' marked for import from '<libname>' already defined by '<oepname>'`

L6786W `Mapping symbol #<symnum> '<msym>' in <oepname>(<secnum>:<secname>) defined at unaligned offset=0x<moffset>`

L6787U `Region table handler '<handlername>' needed by entry for <regionname> was not found.`

L6788E `Scatter-loading of execution region <er1name> to [<base1>,<limit1>) will cause the contents of execution region <er2name> at [<base2>,<limit2>) to be corrupted at run-time.`

This occurs when scatter-loading takes place and an execution region is put in a position where is overwrites partially or wholly another execution region (which can be itself or another region).

For example, this works:

```
LOAD_ROM 0x0000 0x4000
{
 EXEC1 0x0000 0x4000
 {
 * (+RO)
 }
 EXEC2 0x4000 0x4000
 {
 * (+RW,+ZI)
 }
}
```

This generates the error:

```
LOAD_ROM 0x0000 0x4000
{
 EXEC1 0x4000 0x4000
 {
 * (+RW,+ZI)
 }
 EXEC2 0x0000 0x4000
 {
 * (+RO)
 }
```

and reports:

```
Error: L6788E: Scatter-loading of execution region EXEC2 will cause the
contents of execution region EXEC2 to be corrupted at run-time.
```

See the following in *Using the Linker*:

- Chapter 8 *Using scatter files*.

L6789U Library <library> member <filename> : Endianness mismatch.

L6790E Relocation #<rel_class>:<relnum> in <objname>(<secname>) with respect to <symname>. May not IMPORT weak reference through GOT-generating relocation

L6791E Unknown personality routine <pr> at 0x<offset> <oepname>(<secname>).

L6792E Descriptor at offset 0x<offset> <oepname>(<secname>).

L6793E Expecting Landing pad reference at offset 0x<offset> in cleanup descriptor <oepname>(<secname>).

L6794E Expecting Landing pad reference at offset 0x<offset> in catch descriptor <oepname>(<secname>).

L6795E Expecting RTTI reference at offset 0x<offset> in catch descriptor <oepname>(<secname>).

L6796E Descriptor at offset 0x<offset> <oepname>(<secname>) overruns end of section.

L6797E Data at Offset 0x<offset> in exception table <oepname>(<secname>) overruns end of section

L6798E Expecting RTTI reference at offset 0x<offset> in Function Specification descriptor <oepname>(<secname>).

L6799E Expecting Landing Pad reference at offset 0x<offset> in Function Specification descriptor <oepname>(<secname>).

A landing pad is code that cleans up after an exception has been raised. If the linker detects old-format exception tables, it automatically converts them to the new format.

This message does not appear unless you are using a later version of the linker with an earlier version of the compiler.

L6800W Cannot convert generic model personality routine at 0x<offset> <oepname>(<secname>).

A personality routine unwinds the exception handling stack. If the linker detects old-format exception tables then it automatically converts them to the new format. This message indicates a fault in the compiler. Contact your supplier.

L6801E <objname>(<secname>) containing <secarmthumb> code cannot use the address of '~IW (The user intended not all code should interwork)' <funarmthumb> function <sym>.

The linker can diagnose where a non-interworking (`~IW`) function has its address taken by code in the other state. This error is disabled by default, but can be enabled by linking with `--strict`. The error can be downgraded to a warning with `--diag_warning 6801` and subsequently suppressed completely if required with `--diag_suppress 6801`

Where code, for example, in `a.c` uses the address of a non-interworking function in `t.c`:

```
armcc -c a.c
armcc --thumb -c t.c
armlink t.o a.o --strict
```

reports:

```
Error: L6801E: a.o(.text) containing ARM code cannot use the address of
'~IW' Thumb function foo.
```

L6802E `Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <armsym>. Thumb Branch to non-Thumb symbol in <armobjname>(<armsecname>).`

L6803W `Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <armsym>. Thumb Branch is unlikely to reach target in<armobjname>(<armsym>).`

L6804W `Legacy use of symbol type STT_TFUNC detected`

L6805E `Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <armsym>. Branch to untyped Absolute symbol in <armobjname>, target state unknown`

L6806W `Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <othersym>. Branch to untyped symbol in <otherobjname>(<othersecname>), ABI requires external code symbols to be of type STT_FUNC.`

L6807E `Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <othersym>. Branch to untyped symbol in same section. State change is required.`

L6809W `Relocation <rel_class>:<idx> in <objname>(<secname>) is of deprecated type <rtype>, please see ARMELF for ABI compliant alternative`

L6810E `Relocation <rel_class>:<idx> in <objname>(<secname>) is of obsolete type <rtype>`

Relocation errors and warnings are most likely to occur if you are linking object files built with previous versions of the ARM tools.

To show relocation errors and warnings use the `--strict_relocations` switch. This option enables you to ensure ABI compliance of objects. It is off by default, and deprecated and obsolete relocations are handled silently by the linker.

See the following in the *Linker Reference*:

- *--strict_relocations, --no_strict_relocations on page 2-159*.

L6812U `Unknown symbol action type, please contact your supplier.`

L6813U `Could not find Symbol <symname> to rename to <newname>.`

See the following in the *Linker Reference*:

- *RENAME on page 3-5*.

L6815U `Out of memory. Allocation Size:<alloc_size> System Size:<system_size>.`

This error is reported by ARM Compiler toolchain v4.1 and later. It provides information about the amount of memory available and the amount of memory required to perform the link step.

This error occurs because the linker does not have enough memory to link your target object. This is not common, but might be triggered for a number of reasons, such as:

- linking very large objects or libraries together
- generating a large amount of debug information
- having very large regions defined in your scatter file.

In these cases, your workstation might run out of virtual memory.

This issue might also occur if you use the `FIXED` scatter-loading attribute. The `FIXED` attribute forces an execution region to become a root region in ROM at a fixed address. The linker might have to add padding bytes between the end of the previous execution region and the `FIXED` region, to generate the ROM image. The

linker might run out of memory if large amounts of padding are added when the address of the FIXED region is far away from the end of the execution region. The link step might succeed if the gap is reduced.

See the following in the *Linker Reference*:

- *Execution region attributes on page 4-12*.

See the following in the *Using the Linker*:

- *Using the FIXED attribute to create root regions* on page 8-17.

While the linker can generate images of almost any size, it requires a larger amount of memory to run and finish the link. Try the following solutions to improve link-time performance, to avoid the `Out of memory` error:

1. Shut down all non-essential applications and processes when you are linking.

   For example, if you are running under Eclipse, try running your linker from the command-line, or exiting and restarting Eclipse between builds.

2. Use the 64-bit version of the linker.

   If you are using a 64-bit operating system, it is possible to make use of a 64-bit version of armlink. See the following in the *Introducing the ARM Compiler toolchain*:

   - *Changing to the 64-bit linker* on page 2-7.

3. Use the `--no_debug` linker option.

   This command tells the linker to create the object without including any debug information. See the following in the *Linker Reference*:

   - *--debug, --no_debug on page 2-41*.

   Note

   It is not possible to perform source level debugging if you use this option.

4. Reduce debug information.

   If you do not want to use the `--no_debug` option, there are other methods you can use to try and reduce debug information. See the following in *Using the Compiler*:

   - *Methods of reducing debug information in objects and libraries on page 6-22*.

   You can also use the fromelf utility to strip debug information from objects and libraries that you do not have to debug. See the following in *Using the fromelf Image Converter*:

   - *--strip=option[,option,...] on page 4-70*.

5. Use partial linking.

   You can use partial linking to split the link stage over a few smaller operations. Doing this also stops duplication of the object files in memory in the final link.

   See the following in the *Linker Reference*:

   - *--partial on page 2-118*.

6. Increase memory support on Windows operating systems.

   On some Windows operating systems it is possible to increase the virtual address space from 2GB (the default) to 3GB. For more information, see the following Microsoft article:

   - Memory Support and Windows Operating Systems, `http://msdn.microsoft.com/en-us/windows/hardware/gg487508.aspx`

7. Use the `--no_eager_load_debug` linker option.

   This option is available in RVCT 4.0 build 697 and later. It causes the linker to remove debug section data from memory after object loading. This lowers the peak memory usage of the linker at the expense of some linker performance, because much of the debug data has to be loaded again when the final image is written.

   See the following in the *Linker Reference*:

   - *--eager_load_debug, --no_eager_load_debug on page 2-52*.

If you are still experiencing the same problem, raise a support case.

L6828E `Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <symname>, Branch source address <srcaddr> cannot reach next available pool at [<pool_base>,<pool_limit>). Please use the --veneer_pool_size option to increase the contingency.`

The `--veneer_inject_type=pool` veneer generation model requires branches to veneers in the pool to be able to reach the pool limit, which is the highest possible address a veneer can use. If a branch is later found that cannot reach the pool limit, and `armlink` is able to fit all the veneers in the pool into the lower pool limit, then `armlink` reduces the pool limit to accomodate the branch. Error message L6828 is issued only if `armlink` is unable to lower the pool limit.

See the following in the *Linker Reference*:

- *--veneer_inject_type=type on page 2-180*.

L6898E `Relocation #<rel_class>:<idx> in <objname>(<secname>) with respect to <armsym>. ARM branch to non-ARM/Thumb symbol in <armobjname>(<armsecname>).`

L6899E `Existing SYMDEFS file '<filename> 'is read-only.`

L6900E `Expected parentheses to specify priority between AND and OR operators.`

L6901E `Expected symbol name.`

L6902E `Expected a string.`

L6903E `Cannot execute '<text>' in '<clause>' clause of script.`

L6904E `Destination symbol of rename operation clashes with another rename.`

L6905E `Source symbol of rename operation clashes with another rename.`

L6906E `(This is the rename operation which it clashes with.)`

L6907E `Expected an expression.`

L6910E `Expected a phase name.`

L6912W `Symbol <symname> defined at index <idx> in <oepname>(<secname>), has ABI symbol type <symtype> which is inconsistent with mapping symbol type <maptype>.`

L6913E `Expected execution region name.`

L6914W `option <spurious> ignored when using --<memoption>.`

L6915E `Library reports error: <msg>`

The message is typically one of the following:

- `Error: L6915E: Library reports error: scatter-load file declares no heap or stack regions and __user_initial_stackheap is not defined.`

  or

`Error: L6915E: Library reports error: The semihosting __user_initial_stackheap cannot reliably set up a usable heap region if scatter loading is in use`

It is most likely that you have not re-implemented `__user_setup_stackheap()` or you have not defined `ARM_LIB_STACK` or `ARM_LIB_HEAP` regions in the respective scatter file.

Note

`__user_setup_stackheap()` supersedes the deprecated function `__user_initial_stackheap()`.

See the following in *Developing Software for ARM® Processors*:

— *Placing the stack and heap on page 3-13*.

See the following in *C and C++ Libraries and Floating-Point Support Reference*:

— *__user_setup_stackheap() on page 2-60*

— *Legacy function __user_initial_stackheap() on page 2-71*.

See the following in *Using the Linker*:

— *Reserving an empty region* on page 8-54.

- `Error: L6915E: Library reports error: __use_no_semihosting was requested but <function> was referenced.`

  Where `<function>` represents `__user_initial_stackheap`, `_sys_exit`, `_sys_open`, `_sys_tmpnam`, `_ttywrch`, `system`, `remove`, `rename`, `_sys_command_string`, `time`, or `clock`

  This error can appear when retargeting semihosting-using functions, to avoid any `SVC/BKPT` instructions being linked-in from the C libraries.

  Ensure that no semihosting-using functions are linked in from the C library by using:

  `#pragma import(__use_no_semihosting)`

  See the following in *Using C and C++ Libraries and Floating-Point Support*:

  — *Using the libraries in a nonsemihosting environment on page 2-36*.

  If there are still semihosting-using functions being linked in, the linker reports this error.

  To resolve this, you must provide your own implementations of these C library functions.

  The `emb_sw_dev` directory contains examples of how to re-implement some of the more common semihosting-using functions. See the file `retarget.c`.

  See *Using C and C++ Libraries and Floating-Point Support* for more information on using semihosting-using C library functions.

  Note

  The linker does not report any semihosting-using functions such as, for example, `__semihost()`, in your own application code.

  To identify which semihosting-using functions are still being linked-in from the C libraries:

  — Link with `armlink --verbose --list err.txt`

  — Search `err.txt` for occurrences of `__I_use_semihosting`

  For example:

```
…
Loading member sys_exit.o from c_4.l.
reference : __I_use_semihosting
definition: _sys_exit
…
```

This shows that the semihosting-using function `_sys_exit` is linked-in from the C library. To prevent this, you must provide your own implementation of this function.

- `Error: L6915E: Library reports error:__use_no_heap was requested, but <reason> was referenced`

  If `<reason>` represents `malloc`, `free`, `__heapstats`, or `__heapvalid`, the use of `__use_no_heap` conflicts with these functions.
- `Error: L6915E: Library reports error:__use_no_heap_region was requested, but <reason> was referenced`

  If `<reason>` represents `malloc`, `free`, `__heapstats`, `__heapvalid`, or `__argv_alloc`, the use of `__use_no_heap_region` conflicts with these functions.

L6916E `Relocation #<rel_class>:<idx> in <oepname>(<spname>). R_ARM_CALL for conditional BL instruction).`

L6917E `Relocation #<rel_class>:<idx> in <oepname>(<spname>). R_ARM_JUMP24 for BLX instruction.`

L6918W `Execution region <ername> placed at 0x<eraddr> needs padding to ensure alignment <spalign> of <oepname>(<spname>).`

L6922E `Section <objname>(<secname>) has mutually exclusive attributes (READONLY and TLS)`

L6923E `Relocation #<rel_class>:<idx> in <oepname>(<spname>) with respect to <symname>. TLS relocation <type> to non-TLS symbol in <symobjname>(<symsecname>).`

L6924E `Relocation #<rel_class>:<idx> in <oepname>(<spname>) with respect to <symname>. Non-TLS relocation <type> to STT_TLS symbol in <symobjname>(<symsecname>).`

L6925E `Ignoring <token> attribute for region <region>. MemAccess support has been removed.`

L6926E `Relocation #<rel_class>:<idx> in <oepname>(<spname>) has incorrect relocation type <rtype> for instruction encoding 0x<bl>.`

L6927E `Relocation #<rel_class>:<idx> in <oepname>(<spname>) has incorrect relocation type <rtype> for instruction encoding 0x<bl1><bl2>.`

L6932W `Library reports warning: <msg>`

See the following in *Migration and Compatibility*:

- *Linker changes between RVCT v3.1 and RVCT v4.0* on page 9-8.

L6935E `Debug Group contents are not identical, <name> with signature sym <sig> from objects (<new>) and (<old>)`

L6936E `Multiple RESOLVE clauses in library script for symbol '<sym>'.`

L6937E `Multiple definitions of library script function '<func>'.`

L6939E `Missing alignment for region <regname>.`

L6940E `Alignment <alignment> for region <regname> must be at least 4 and a power of 2 or MAX.`

L6941W chmod system call failed for file <filename> error <perr>

L6942E Execution Region <ername> contains multiple <type>, sections:

L6966E Alignment <alignment> for region <regname> cannot be negative.

L6967E Entry point (<address>) points to a THUMB instruction but is not a valid THUMB code pointer.

L6968E Could not parse Linux Kernel version \"<kernel>\".

L6969W Changing AT Section <name> type from RW to RO in <ername>.

L6971E <objname>(<secname>) type <type> incompatible with <prevobj>(<prevname>) type <prevtype> in er <ername>.

You might see this message when placing `__at` sections with a scatter file. For example, the following code in `main.c` and the related scatter file gives this error:

```
int variable __attribute__((at(0x200000)));
```

```
LR1 0x0000 0x20000
{
    ER1 0x0 0x2000
    {
        *(+RO)
    }
    ER2 0x8000 0x2000
    {
        main.o
    }
    RAM 0x200000 (0x1FF00-0x2000)
    {
        *(+RW, +ZI)
    }
}
```

The variable has the type ZI, and the linker attempts to place it at address `0x200000`. However, this address is reserved for RW sections by the scatter file. This produces the error:

```
Error: L6971E: stdio_streams.o(.data) type RW incompatible with
main.o(.ARM.__AT_0x00200000) type ZI in er RAM.
```

To fix this change the address in your source code, for example:

```
int variable __attribute__((at(0x210000)));
```

See the following in *Using the Linker*:

- *Placing functions and data at specific addresses* on page 8-18
- *Using __at sections to place sections at a specific address* on page 8-37.

L6972E <objname>(<secname>) with required base 0x<required> has been assigned base address 0x<actual>.

L6973E Error placing AT section at address 0x<addr> in overlay ER <ername>.

For example, you attempted to use `__attribute__((at(`*address*`)))` to place a section when building a DLL or application with an overlay region. `__attribute__((at(`*address*`)))` requires that you specify a fixed location in a scatter file with `.ARM.__at_`*address*. In this case, you must also specify the `--no_autoat` linker option.

See the following in *Using the Linker*:

- *Using __at sections to place sections at a specific address* on page 8-37.

See the following in the *Linker Reference*:

- *--autoat, --no_autoat on page 2-17*.

L6974E `AT section <name> does not have a base address.`

See the following in *Using the Linker*:

- *Using __at sections to place sections at a specific address* on page 8-37.

L6975E `<objname>(<secname>) cannot have a required base and SHF_MERGE.`

L6976E `<objname>(<secname>) cannot have a required base and SHF_LINK_ORDER.`

L6977E `<objname>(<secname>) cannot be part of a contiguous block of sections`

L6978W `<objname>(<secname>) has a user defined section type and a required base address.`

L6979E `<objname>(<secname>) with required base address cannot be placed in Position Independent ER <ername>.`

L6980W `FIRST and LAST ignored for <objname>(<secname>) with required base address.`

See the following in *Using the Linker*:

- *Placing sections with FIRST and LAST attributes* on page 4-21.

L6981E `__AT incompatible with BPABI and SystemV Image types`

See the following in *Using the Linker*:

- *Restrictions on placing __at sections* on page 8-38.

L6982E `AT section <objname>(<spname>) with base <base> limit <limit> overlaps address range with AT section <obj2name>(<sp2name>) with base <base2> limit <limit2>.`

See the following in *Using the Linker*:

- *Using __at sections to place sections at a specific address* on page 8-37.

L6983E `AT section <objname>(<spname>) with required base address <base> out of range for ER <ername> with base <erbase> and limit <erlimit>.`

This can occur if you specify `__attribute__((at(address)))` in your code, `.ARM.__at_address` in your scatter file, and `--no_autoat` option on the linker command-line. In this case, the address part of `.ARM.__at_address` must be specified as eight hexadecimal digits. For example:

```
int x1 __attribute__((at(0x4000)));  // defined in function.c

; scatter file
LR1 0x0
{
    ...
        function.o(.ARM.__at_0x00004000)
    ...
}
```

See the following in *Using the Linker*:

- *Using __at sections to place sections at a specific address* on page 8-37.

See the following in the *Linker Reference*:

- *--autoat, --no_autoat on page 2-17*.

L6984E `AT section <objname>(<spname>) has required base address <base> which is not aligned to section alignment <alignment>.`

See the following in *Using the Linker*:

- *Using __at sections to place sections at a specific address* on page 8-37.

L6985E `Unable to automatically place AT section <objname>(<spname>) with required base address <base>. Please manually place in the scatter file using the --no_autoat option.`

See the following in *Using the Linker*:

- *Using __at sections to place sections at a specific address* on page 8-37.

See the following in *Linker Reference*:

- *--autoat, --no_autoat on page 2-17*.

Chapter 5
ELF Image Converter Errors and Warnings

The following topic describes the error and warning messages for the ELF image converter, `fromelf`:

5.1 List of the fromelf error and warning messages

The error and warning messages for `fromelf` are:

Q0105E `Load region #<segindex> extends beyond top of address space.`

Q0106E `Out of Memory.`

Q0107E `Failed writing output file '<filename>': <reason>`

Q0108E `Could not create output file '<filename>': <reason>`

Q0119E `No output file specified.`

Q0120E `No input file specified.`

Q0122E `Could not open file '<filename>': <reason>`

If `<reason>` is Invalid argument, this might be because you have invalid characters on the command-line. For example, on Windows you might have used the escape character \ when specifying a filter with an archive file:

```
fromelf --elf --strip=all t.a\(test*.o\) -o filtered/
```

On Windows, use:

```
fromelf --elf --strip=all t.a(test*.o) -o filtered/
```

See the following in *Using the fromelf Image Converter*:

- *input_file on page 4-48*.

Q0128E `File i/o failure.`

This error can occur if you specify a directory for the `--output` command-line option, but you did not terminate the directory with a path separator. For example, `--output=my_elf_files/`.

See the following in *Using the fromelf Image Converter*:

- *--output=destination on page 4-57*.

Q0129E `Not a 32 bit ELF file.`

Q0130E `Not a 64 bit ELF file.`

Q0131E `Invalid ELF identification number found.`

This error is given if you attempt to use fromelf on a file which is not in ELF format, or which is corrupted. Object (`.o`) files and executable (`.axf`) files are in ELF format.

Q0132E `Invalid ELF section index found <idx>.`

Q0133E `Invalid ELF segment index found <idx>.`

Q0134E `Invalid ELF string table index found <idx>.`

Q0135E `Invalid ELF section entry size found.`

Q0136E `ELF Header contains invalid file type.`

Q0137E `ELF Header contains invalid machine name.`

Q0138E `ELF Header contains invalid version number.`

See Q0131E.

Q0147E `Failed to create Directory <dir>: <reason>`

If <reason> is File exists, this might be because you have specified a directory that has the same name as a file that already exists. For example, if a file called `filtered` already exists, then the following command produces this error:

```
fromelf --elf --strip=all t.a(test*.o) -o filtered/
```

The path separator character / informs fromelf that `filtered` is a directory.

See the following in *Using the fromelf Image Converter*:

- *--output=destination on page 4-57*.

Q0171E `Invalid st_name index into string table <idx>.`

See Q0131E.

Q0172E `Invalid index into symbol table <idx>.`

See Q0131E.

Q0186E `This option requires debugging information to be present`

The `--fieldoffsets` option requires the image to be built with dwarf debug tables.

Q0425W `Incorrectly formed virtual function elimination header in file`

This might indicate a compiler fault. Contact your supplier.

Q0426E `Error reading vtable information from file`

This might indicate a compiler fault. Contact your supplier.

Q0427E `Error getting string for symbol in a vtable`

This might indicate a compiler fault. Contact your supplier.

Q0433E `Diagnostic style <style> not recognised`

Q0440E `No relocation sections for <secname>`

Q0447W `Unknown Diagnostic number (<num>)`

Q0448W `Read past the end of the compressed data while decompressing section '<secname>' #<secnum> in <file>`

This might indicate an internal fault. Contact your supplier.

Q0449W `Write past the end of the uncompressed data buffer of size <bufsize> while decompressing section '<secname>' #<secnum> in <file>`

This might indicate an internal fault. Contact your supplier.

Q0450W `Section '<secname>' #<secnum> in file <file> uses a mixture of legacy and current ABI relocation types.`

Q0451W `Option '--strip symbols' used without '--strip debug' on an ELF file that has debug information.`

Q0452W `Option '--strip filesymbols' used without '--strip debug' on an ELF file that has debug information.`

Q0453W `Stripping path names from '<path1>' and '<path2>' produces a duplicate file name '<filename>'.`

Q0454E `In ELF file: <details>`

Chapter 6
Librarian Errors and Warnings

The following topic describes the error and warning messages for the ARM Librarian, `armar`:

- *List of the armar error and warning messages* on page 6-2.

6.1 List of the armar error and warning messages

The error and warning messages for `armar` are:

L6800U `Out of memory`

L6825E `Reading archive '<archive>' : <reason>`

L6826E `'<archive>' not in archive format`

L6827E `'<archive>': malformed symbol table`

L6828E `'<archive>': malformed string table`

L6829E `'<archive>': malformed archive (at offset <offset>)`

L6830E `Writing archive '<archive>' : <reason>`

L6831E `'<member>' not present in archive '<archive>'`

L6832E `Archive '<archive>' not found : <reason>`

L6833E `File '<filename>' does not exist`

L6835E `Reading file '<filename>' : <reason>`

L6836E `'<filename>' already exists, so will not be extracted`

L6838E `No archive specified`

L6839E `One of the actions -[<actions>] must be specified`

L6840E `Only one action option may be specified`

L6841E `Position '<position>' not found`

L6842E `Filename '<filename>' too long for file system`

L6843E `Writing file '<filename>' : <reason>`

L6874W `Minor variants of archive member '<member>' include no base variant`

Minor variants of the same function exist within a library. Find the two equivalent objects and remove one of them.

L6875W `Adding non-ELF object '<filename>' to archive '<name>'`

Chapter 7
Other Errors and Warnings

The following topic describes other error and warning messages that might be displayed by the tools:

- *List of other error and warning messages* on page 7-2.

Note

These error messages can be produced by any of the tools.

When the message is displayed, the *X* prefixing the message number is replaced by the appropriate letter relating to the application. For example, the code `X3900U`, is displayed as `L3900U` by the linker when you have specified an unrecognized option.

7.1 List of other error and warning messages

Other error and warning messages that might be displayed by the tools are:

***X*3900U** `Unrecognized option '<dashes><option>'.`

`<option>` is not recognized by the tool. This could be because of a spelling error or the use of an unsupported abbreviation of an option.

***X*3901U** `Missing argument for option '<option>'.`

***X*3902U** `Recursive via file inclusion depth of <limit> reached in file '<file>'.`

***X*3903U** `Argument '<argument>' not permitted for option '<option>'.`

Possible reasons include malformed integers or unknown arguments.

***X*3904U** `Could not open via file '<file>'.`

***X*3905U** `Error when reading from via file '<file>'.`

***X*3906U** `Malformed via file '<file>'.`

***X*3907U** `Via file '<file>' command too long for buffer.`

***X*3908U** `Overflow: '<string>' will not fit in an integer.`

***X*3910W** `Old syntax, please use '<hyphens><option><separator><parameter>'.`

***X*3912W** `Option '<option>' is deprecated.`

***X*3913W** `Could not close via file '<file>'.`

***X*3915W** `Argument '<argument>' to option '<option>' is deprecated`

***X*3916U** `Unexpected argument for option '<dashes><option>'`

***X*3917U** `Concatenated options cannot have arguments: -<option> <arg>`

***X*9905E** `cannot use --apcs=/hardfp without floating point hardware`

***X*9906E** `cannot use --apcs=/hardfp with fpu <fpu_option>`

***X*9907E** `unable to select no floating point support`

***X*9908E** `--fpmode=none overrides --fpu choice`

Appendix A
Revisions for the Errors and Warnings Reference

The following technical changes have been made to the *Errors and Warnings Reference*:

Table A-1 Differences between Issue H and Issue I

Change	Topics affected
Changes to the old-style compiler messages: • Updated message ID numbers to reflect renumbering of C3nnn to C4nnn. • Removed obsolete compiler messages C3000E, C3015E, C3055U, and C3064E.	*List of the old-style armcc error and warning messages* on page 2-70
Changes to the compiler messages: • Added more detail for compiler errors and warnings 95, 1043, 1083, 1296, 1559.	*List of the armcc error and warning messages* on page 2-4
Changes to the linker messages: • Added more detail for linker errors and warnings L6202E, L6330W. • Changed the article linked to from linker error L6238E.	*List of the armlink error and warning messages* on page 4-3
Changes to the assembler messages: • Where appropriate, changed the term *local label* to *numeric local label*. • Improved the description of assembler warning A1746W.	*List of the armasm error and warning messages* on page 3-2

Table A-2 Differences between Issue G and Issue H

Change	Topics affected
Changes to the compiler messages: • added error and warning messages 2902 to 3040 inclusive, which are new in ARM Compiler 5.0 • added error and warning messages 3049 to 3081 inclusive, which are new in ARM Compiler 5.01.	*List of the armcc error and warning messages* on page 2-4
Enhanced the description of the linker error message L6973E.	*List of the armlink error and warning messages* on page 4-3

Table A-3 Differences between Issue E and Issue F

Change	Topics affected
Changes to the compiler messages: • changed the version number component of internal compiler errors from two to three digits • removed messages 1113 and 2523 because they are no longer valid • added the remarks 2813 and 2815.	• *Internal errors and other unexpected failures* on page 2-2 • *List of the armcc error and warning messages* on page 2-4
Changes to the assembler messages: • corrected the E, W or U suffix in many error and warning message codes • added more detail for A1322E, A1477E, A1745W, A1786W, A1788W and A1809W.	*List of the armasm error and warning messages* on page 3-2
Changes to the linker messages: • added cross references to the descriptions for L6218E and L6932W • added more detail for L6242E and L6248E.	*List of the armlink error and warning messages* on page 4-3
Changed the `ARMCC`*nn*`LIB` and `ARMCC`*nn*`INC` environment variables to `ARMCC`*n*`LIB` and `ARMCC`*n*`INC`.	Various topics
Where appropriate: • changed Thumb-1 to 16-bit Thumb • changed Thumb-2 to 32-bit Thumb.	Various topics

Table A-4 Differences between Issue D and Issue E

Change	Topics affected
Changes to the compiler messages: • added more detail for error 1558.	*List of the armcc error and warning messages* on page 2-4
Changes to the assembler messages: • removed messages A1588E, A1589E, A1597E, A1613E, A1614E, and A1646W, because they are not reachable.	*List of the armasm error and warning messages* on page 3-2

Table A-5 Differences between Issue C and Issue D

Change	Topics affected
Changes to the compiler messages: • added messages C3335E, C3336W, C3337E, and C3338W • removed error C3053W, because `--profile` is deprecated • added cross references to various messages.	*List of the old-style armcc error and warning messages* on page 2-70
Changes to the assembler messages: • added messages A1903E, A1907W, A1908E, and A1909E • added cross references to A1450W.	*List of the armasm error and warning messages* on page 3-2
Changes to the linker messages: • added L6064E • corrected examples for L6216E • added L6815U • updated the description of L6002U • updated the description of L6310W • added cross references to various messages.	*List of the armlink error and warning messages* on page 4-3
Changes to the fromelf messages: • added cross references to various messages.	*List of the fromelf error and warning messages* on page 5-2

Table A-6 Differences between Issue B and Issue C

Change	Topics affected
Added the linker error L6058E.	*List of the armlink error and warning messages* on page 4-3
Added the linker error L6828E.	*List of the armlink error and warning messages* on page 4-3

Table A-7 Differences between Issue A and Issue B

Change	Topics affected
Added more detail for compiler errors 631 and 634.	*List of the armcc error and warning messages* on page 2-4
Removed the assembler error A1590E.	*List of the armasm error and warning messages* on page 3-2
Added more detail for the assembler warning A1746W.	*List of the armasm error and warning messages* on page 3-2
Added the Linker error L6065E.	*List of the armlink error and warning messages* on page 4-3

Table A-7 Differences between Issue A and Issue B (continued)

Change	Topics affected
Added more detail for linker errors L6220E, L6221E, L6384E, L6915E, and L6971E.	*List of the armlink error and warning messages* on page 4-3
Added cross-references for linker errors L6224E and L6469E.	*List of the armlink error and warning messages* on page 4-3
Added more detail for fromelf errors Q0122E, Q0128E, and Q0147E.	*List of the fromelf error and warning messages* on page 5-2