RVCT 2.0 Build Tools - Errors and Warnings

Release 1.0 – 27th August 2003

Introduction

This document illustrates the errors and warning messages that are generated by the Build Tools of ARM RealView Compilation Tools 2.0 and later. If you are using ADS (ADS 1.2, 1.1 or 1.0.1) or RVCT 1.2 then please refer to the "ADS 1.2 Build Tools – Errors and Warnings" document instead.
This document is divided into the following sections:

1. ARM C and C++ Compilers (armcc, tcc, armcpp, tcpp)
2. ARM Assembler (armasm)
3. ARM Linker (armlink)
4. ARM ELF Format Conversion Utility (fromelf)
5. ARM Librarian (armar)

The errors and warnings are listed in numeric order. Not all the errors and warnings are yet fully described. The majority of the warnings and errors produced by the build tools are self-explanatory. However, if there are any messages that you do not understand, or which you need more information about, then please contact your supplier, providing as much information as possible about your system and commands used.

Note that this document does not include any reference for errors and warnings emitted by the Licence Management software. For information on this, please see the License Management FAQ at
http://www.arm.com/support/licensemanagement

This document is intended to complement, not replace, the RVCT 2.0 documentation. It should be read in conjunction with the RVCT build tools manuals, in particular the section(s) referring to controlling of warning and error message generation. We would also recommend that you consult the RVCT FAQ at
http://www.arm.com/support/rvct2_faq for further information

Please also ensure that you have the latest "patch" of the build tool(s) you are using. These are downloadable from the appropriate link at http://www.arm.com/support/downloads

2. ARM C/C++ Compilers (armcc, tcc, armcpp, tccp)

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 may point to a potential "bug" in the compiler itself. If such errors are not present in this document, you will be required to supply an example of the source code that causes the error(s), to your tools supplier.

To facilitate the investigation, please 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 may be necessary to preprocess the file (i.e. to take account of #include'd header files, etc). To do this, pass the file through the preprocessor as follows:

```
            armcc <options> –E sourcefile.c > PPsourcefile.c
    or      tcc <options> –E sourcefile.c > PPsourcefile.c
```

where `<options>` are your normal compile switches, (-O1, -g, -I, -D, etc), but *without* -c.

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

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

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

Controlling the Errors and Warnings Messages

This is documented in **RVCT 2.0 Compiler and Libraries Guide Section 2.3.12**. The compiler will normally warn of potential portability problems and other hazards.

When porting legacy code (e.g. in old-style C) to the ARM, many warnings may be reported. It may be tempting to disable all such warnings with "`-w`", however, our recommendation, for portability reasons, is to change the code to make it ANSI compatible, rather than suppressing the warnings.

Some warnings are suppressed by default. To override this, the "`-fx`" switch can be used to enable all those warnings that are suppressed by default.

List of Errors and Warnings Messages

> **1**: `last line of file ends without a new line`
>
> **2**: `last line of file ends with a backslash`
>
> **3**: `#include file "xxxx" includes itself`
>
> **4**: `out of memory`
>
> **5**: `could not open source file "xxxx"`
>
> Example:
> `#include <file.h>`
> `Error:  #5: could not open source file "file.h"`
>
> 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[] = {"\" };
```

9: `nested comment is not allowed`

For example:
```
/*nested
  /*comment*/
```

10: `"#" not expected here`

A '#' character is in an incorrect position

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 should cure the problem, e.g.
```
#endif /* SOMETHING */
```

16: `"xxxx" 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 "xxxx" is undefined

Example when compiled for C++:

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

gives:
```
Error:  #20: identifier "arg" is undefined
```
This is a common error that occurs where there is no prototype for a function.
e.g. when `printf()` is used with no `#include <stdio.h>`, the warning occurs:

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

gives:
```
Error:  #20: identifier "printf" is undefined
```
Example:
```
int foo(void)
{
        int a = 4;
        a = i;
}
```
results in the error:

```
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

digit 8 or 9 found in octal number
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';
```

27: `character value is out of range`

For example:
`char foo[] = {"\xBBBB" };`
gives:
`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: xxxx`

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. For example if the identifier which immediately should follow a preprocessor command is missing, e.g. Missing identifier after #define, results in:

`Error:  #40: expected an identifier`

This also warns about future compatibility with C++.
Example:
`int *new(void *p) { return p; }`
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: `this predefined name may not be redefined`

47: `incompatible redefinition of macro "entity" (declared at line xxxx)`

`Macro` has been defined twice (with different replacement strings).
If you need to do this, undefine the macro (`#undef`) before the second definition.
Example:

```
#define TEST 0
#define TEST 1
int foo()
{
    return TEST;
}
```

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 via a compiler option, but you can use conditional preprocessing. For example:

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

int foo()
{
    return TEST;
}
```

Compiling with "armcc -c foo.c" will define TEST to be 1 (the default).
Compiling with "armcc -c -DTEST_EQUALS_ZERO foo.c" will define 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 can occur with enums, for example, it is common to use enums to represent bit positions in a memory mapped register, like this:

```
typedef enum
{
    Bit0  = 0x00000001,
    Bit1  = 0x00000002,
    Bit2  = 0x00000004,
:
    Bit30 = 0x40000000,
    Bit31 = 0x80000000
} Bits;
```

Strictly this is not allowed, because Bit31 exceeds 0x7FFFFFFF, the upper bound for a signed int, so the compiler will report:
`Error:   #66: enumeration value is out of "int" range`
when compiling with `-strict`. See RVCT 2.0 Compilers and Libraries Guide, section 3.3.4, "Structures, unions, enumerations, and bitfields".

The workaround is to change Bit31 to:
```
    Bit31 = (int)0x80000000
} Bits;
```

An overflow no longer occurs, and so no error is reported. Note, however, that the value of Bit31 is now negative because it is a signed int.

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;
```
gives:
`Warning:  #68-D: integer conversion resulted in a change of sign`

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 will not be able to allocate a size of the structure. As it appears to the compiler that the type is incomplete and reports:
`Error:  #70: incomplete type is not allowed`

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

75: `operand of "*" must be a pointer`

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, e.g. printf(), must follow at least one parameter, e.g 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;

Error:  #84: invalid combination of type specifiers
```

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 not allowed. For example:
```
char name[0] = "Hello";
```
Causes the compiler to report:
```
Error:  #94: the size of an array must be greater than zero
```

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: "xxxx" 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

105: invalid size for bit field

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

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

The compiler reports:
```
Error:  #105: invalid size for bit field
```

106: invalid type for a bit field

Bit fields must have integral type. Example:

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

The compiler reports:
```
Error:  #106: invalid type for a bit field
```

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-kind> "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:

```
int main(void){
int a =0;
break;
```

Results in:
```
Error:  #116: a break statement may only be used within a loop or switch
```

117: non-void <entity-kind> "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-kind> "entity" has no member "xxxx"

136: <entity-kind> "entity" has no field "xxxx"

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);
}
```
Gives:
```
Error:  #140: too many arguments in function call
```

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" will not fit because C strings are always null-terminated (e.g. "Hello\0"). So for the example above 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 will also be raised if there is an implicit cast of non-0 int to pointer, e.g:

```
void foo_func( void )
{
        char *foo=1;
}
```
Gives:
```
#144: a value of type "int" cannot be used to initialize an entity of type
"char *"
```

For the second case this error can be suppressed with the use of the "-Ec" switch.

145: <entity-kind> "entity" may not be initialized

146: too many initializer values

147: declaration is incompatible with <entity-kind> "entity" (declared at line xxxx)

148: <entity-kind> "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 has 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" (declared at line xxxx)

160: name conflicts with previously used external name "xxxx"

161: unrecognized #pragma

163: could not open temporary file "xxxx"

164: name of directory for temporary files is too long ("xxxx")

165: too few arguments in function call

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

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

Gives:
```
#165: too few arguments in function call
```

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`

When attempting to compile some C++ header files with the C compiler instead of the C++ compiler,
```
Error:  #169: expected a declaration
```
is reported.

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){}
```

Reports:
```
Error:  #172: external/internal linkage conflict with previous declaration
```

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

174: `expression has no effect`

175: `subscript out of range`

177: `<entity-kind> "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 `"-Wx"` 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:
```
unsigned long foo = 0x1234;
printf("%0X", 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" may be used for `char`, `short` or `int`. Use "lx" for a long integer, despite both `ints` and `longs` being 32 bits wide on an ARM.

182: could not open source file "xxxx" (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

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

This is warning that the comparison between an unsigned (char, int, etc) value and zero will always evaluate 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 xxxx file

190: invalid intermediate language file

191: type qualifier is meaningless on cast type

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 RVCT 2.0 compiler supports multibyte character sets, such as Unicode. It is possible to use "Escape processing" (as recommended by Kernighan and Richie, section A2.5.2) to encode specific values instead. For example:
```
char *p = "\x12\x34\x56\x78";   // 12 34 56 78
```

There is some example code provided with the RVCT tools which can be found in: "ARM tools directory"\RVCT\Examples\2.0.1\release\windows\unicode.

193: `zero used for undefined preprocessing identifier`

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 "xxxx"`

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 declared implicitly`

This is a common error that occurs where there is no prototype for a function.
Example:
When `printf()` is used with no `#include <stdio.h>`, the warning occurs:

```
void foo(void)
{
        printf("foo");
}
```
gives:
```
Warning:  #223-D: function declared implicitly
```

For ANSI C, this warning can be suppressed with `-Wf` - useful when compiling old-style C in ANSI C mode.

224: `the format string requires additional arguments`

225: `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 allowed for a bit field are `int`, `signed int` and `unsigned int`.
Example:

```
struct X{
char y:2;
};
```
Gives:
```
Error:  #230: nonstandard type for a bit field
```

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 "xxxx" 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-kind> "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 allow a variable to be declared and initialised as separate statements, e.g.

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

In RVCT 2.0 tentative declarations are allowed by default for C code.
The C++ compiler will report:
```
Error:  #247: variable "a" has already been defined
```

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-kind> "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, e.g:
```
typedef int footype;
int x = footype;   // reports Error: #254: type name is not allowed
```

To fix this, create an instance of that type (e.g. a variable of the new type) first, e.g:
```
typedef int footype;
footype bar = 1;
```

```
    int x = bar;
```

255: type definition is not allowed

256: invalid redeclaration of type name "entity" (declared at line xxxx)

257: const <entity-kind> "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 ("xxxx" by default)

262: not a class or struct name

263: duplicate base class name

264: invalid base class

265: <entity-kind> "entity" is inaccessible

For C++ only, the "-Ea" option downgrades access control errors to warnings.
Example:
```
class A { void f() {}; };  // private member
A a;
void g() { a.f(); }        // erroneous access
```
gives:
```
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: implicit 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 "xxxx"

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: "xxxx" 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-kind> "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-kind> "entity" has already been defined

304: no instance of <entity-kind> "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-kind> "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-kind> "entity"

318: override of virtual <entity-kind> "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-kind> "entity" requires a definition

330: <entity-kind> "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" (declared at line xxxx)

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

```
int foo(void);

int bar(void)
{
    int x;
    x = foo();
    return x;
}

extern "C" int foo(void)
{
    return 0;
}
```
Gives:
```
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: "operatorxxxx" 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 "xxxx" matches these operands

350: more than one operator "xxxx" 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-kind> "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-kind> "entity" provides no initializer for:

367: implicitly generated constructor for class "type" cannot initialize:

368: <entity-kind> "entity" defines no constructor to initialize the following:

369: <entity-kind> "entity" has an uninitialized const or reference member

370: <entity-kind> "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 may occur inadvertently when using macros.

Similarly, in C++, this may 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-kind> "entity" matches the required type

387: delete array size expression used (anachronism)

388: "operator->" for class "type" returns invalid type "type"

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 "xxxx" (anachronism)

397: implicitly generated assignment operator cannot copy:

398: cast to array type is nonstandard (treated as cast to "type")

399: <entity-kind> "entity" has an operator newxxxx() but no default operator deletexxxx()

400: <entity-kind> "entity" has a default operator deletexxxx() but no operator newxxxx()

401: destructor for base class "type" is not virtual

403: <entity-kind> "entity" has already been declared

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-kind> "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-kind> "entity" returns incomplete type "type"

410: protected <entity-kind> "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

425: dollar sign ("$") used in identifier

426: temporary used for initial value of reference to non-const (anachronism)

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

432: "enum" declaration is not allowed

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-kind> "entity" is missing

442: too few arguments for <entity-kind> "entity"

443: too many arguments for <entity-kind> "entity"

445: <entity-kind> "entity" is not used in declaring the parameter types of <entity-kind> "entity"

449: more than one instance of <entity-kind> "entity" matches the required type

450: the type "long long" is nonstandard

451: omission of "xxxx" is nonstandard

452: return type may not be specified on a conversion function

456: excessive recursion at instantiation of <entity-kind> "entity"

457: "xxxx" 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 "xxxx" 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

466: "main" is not a valid name for a function template

467: invalid reference to <entity-kind> "entity" (union/nonunion mismatch)

468: a template argument may not reference a local type

469: tag kind of xxxx is incompatible with declaration of <entity-kind> "entity" (declared at line xxxx)

470: the global scope has no tag named "xxxx"

471: <entity-kind> "entity" has no tag member named "xxxx"

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

475: a template argument may not reference a non-external entity

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-kind> "entity" redeclared "inline" after being called

481: invalid storage class for a template declaration

484: invalid explicit instantiation declaration

Example:
```
template template <class T> struct X { }; // is illegal
```

485: <entity-kind> "entity" is not an entity that can be instantiated

486: compiler generated <entity-kind> "entity" cannot be explicitly instantiated

487: inline <entity-kind> "entity" cannot be explicitly instantiated

489: <entity-kind> "entity" cannot be instantiated -- no template definition was supplied

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

493: no instance of <entity-kind> "entity" matches the specified type

494: declaring a void parameter list with a typedef is non-standard

This error may be produced, when the compiler is in ANSI C mode, by a function declaration `f(V)` where V is a void type. In the special syntax `f(<void>)` which 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 "xxxx" may not be redeclared in this scope

497: declaration of "xxxx" hides template parameter

498: template argument list must match the parameter list

500: extra parameter of postfix "operatorxxxx" must be of type "int"

501: an operator name must be declared as a function

502: operator name is not allowed

503: <entity-kind> "entity" cannot be specialized in the current scope

504: nonstandard form for taking the address of a member function

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

510: a template argument may not reference an unnamed type

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-kind> "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-kind> "entity" (declared at line xxxx):

540: support for exception handling is disabled

541: omission of exception specification is incompatible with previous <entity-kind> "entity" (declared at line xxxx)

542: could not create instantiation request file "xxxx"

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:

```
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;
```

Here, '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)."

*) 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;
```

"y" is a POD (Plain Old Data) type, so an alternative would be to not use initialization:
```
case 1:
        int y;
        y = 1;
        z = y + z;
        break;
```

548: transfer of control into an exception handler

549: <entity-kind> "entity" is used before its value is set

550: <entity-kind> "entity" was set but never used

551: <entity-kind> "entity" cannot be defined in the current scope

552: exception specification is not allowed

553: external/internal linkage conflict for <entity-kind> "entity" (declared at line xxxx)

554: <entity-kind> "entity" will not be called for implicit or explicit conversions

555: tag kind of xxxx 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 preprocessor output file

564: cannot open preprocessor output file

568: invalid C output file

569: cannot open C output file

570: error in debug option argument

571: invalid option:

574: invalid number:

575: incorrect host CPU id

576: invalid instantiation mode:

578: invalid error limit:

579: invalid raw-listing output file

580: cannot open raw-listing output file

581: invalid cross-reference output file

582: cannot open cross-reference output file

583: invalid error output file

584: cannot open error output file

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

599: excessive recursive instantiation of <entity-kind> "entity" due to instantiate-all mode

600: strict ANSI 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-kind> "entity"

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

613: invalid error tag:

614: invalid error number:

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

622: invalid precompiled header output file

623: cannot open precompiled header output file

624: "xxxx" is not a type name

625: cannot open precompiled header input file

626: precompiled header file "xxxx" is either invalid or not generated by this version of the compiler

627: precompiled header file "xxxx" was not generated in this directory

628: header files used to generate precompiled header file "xxxx" have changed

629: the command line options do not match those used when precompiled header file "xxxx" was created

630: the initial sequence of preprocessing directives is not compatible with those of precompiled header file "xxxx"

631: unable to obtain mapped memory

632: "xxxx": using precompiled header file "xxxx"

633: "xxxx": creating precompiled header file "xxxx"

634: memory usage conflict with precompiled header file "xxxx"

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 (xxxx bytes required)

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

641: "xxxx" 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: "xxxx" is an unrecognized __declspec attribute

646: a calling convention modifier may not be specified here

647: conflicting calling convention modifiers

648: strict ANSI mode is incompatible with Microsoft mode

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 "xxxx" is not allowed on this declaration

656: transfer of control into a try block

657: inline specification is incompatible with previous "entity" (declared at line xxxx)

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

By default, this results in a call to the library function __pvfn(), which raises the signal SIGPVFN, which is trapped by the default_signal_handler, which displays "Pure virtual fn called" on the console using a semihosting SWI. See RVCT 2.0 Compilers and Libraries Guide, Table 5-18, "Signals used by the C and C++ libraries", and Appendix C.2.

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-kind> "entity" (declared at line xxxx)

678: call of <entity-kind> "entity" (declared at line xxxx) cannot be inlined

679: <entity-kind> "entity" cannot be inlined

680: invalid PCH directory:

681: expected __except or __finally

682: a __leave statement may only be used within a __try

688: "xxxx" not found on pack alignment stack

689: empty pack alignment stack

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

691: <entity-kind> "entity", required for copy that was eliminated, is inaccessible

692: <entity-kind> "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: xxxx 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++

701: an array type is not allowed here

702: expected an "="

703: expected a declarator in condition declaration

704: "xxxx", 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-kind> "entity" is not a variable name

714: __based modifier is not allowed here

715: __based does not precede a pointer operator, __based ignored

716: variable in __based modifier must have pointer type

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-kind> "entity" is not allowed to alter its access

721: nonstandard format string conversion

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

729: invalid combination of DLL attributes

730: <entity-kind> "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-kind> "entity" conflicts with using-declaration of <entity-kind> "entity"

735: using-declaration of <entity-kind> "entity" conflicts with <entity-kind> "entity" (declared at line xxxx)

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

742: <entity-kind> "entity" has no actual member "xxxx"

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-kind> "entity" (declared at line xxxx) 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-kind> "entity"

753: a template-id is not allowed

754: a class-qualified name is not allowed

755: <entity-kind> "entity" may not be redeclared in the current scope

756: qualified name is not allowed in namespace member declaration

757: <entity-kind> "entity" is not a type name

758: explicit instantiation is not allowed in the current scope

759: <entity-kind> "entity" cannot be explicitly instantiated in the current scope

760: <entity-kind> "entity" explicitly instantiated more than once

761: typename may only be used within a template

762: special_subscript_cost option can be used only when compiling C++

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-kind> "entity" is incompatible with that of overridden <entity-kind> "entity"

767: conversion from pointer to smaller integer

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

769: "entity", implicitly called from <entity-kind> "entity", is ambiguous

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

771: "explicit" is not allowed

772: declaration conflicts with "xxxx" (reserved class name)

773: only "()" is allowed as initializer for array <entity-kind> "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-kind> "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: "xxxx", declared in for-loop initialization, may not be redeclared in this scope

780: reference is to <entity-kind> "entity" (declared at line xxxx) -- under old for-init scoping rules it would have been <entity-kind> "entity" (declared at line xxxx)

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

782: definition of virtual <entity-kind> "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-kind> "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-kind> "entity" must precede the first use of <entity-kind> "entity"

790: explicit specialization is not allowed in the current scope

791: partial specialization of <entity-kind> "entity" is not allowed

792: <entity-kind> "entity" is not an entity that can be explicitly specialized

793: explicit specialization of <entity-kind> "entity" must precede its first use

794: template parameter "xxxx" may not be used in an elaborated type specifier

795: specializing <entity-kind> "entity" requires "template<>" syntax

798: option "old_specializations" can be used only when compiling C++

799: specializing <entity-kind> "entity" without "template<>" syntax is nonstandard

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

801: "xxxx" 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-kind> "entity" (declared at line xxxx):

806: omission of exception specification is incompatible with <entity-kind> "entity" (declared at line xxxx)

807: unexpected end of default argument expression

808: default-initialization of reference is not allowed

809: uninitialized <entity-kind> "entity" has a const member

810: uninitialized base class "type" has a const member

811: const <entity-kind> "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

813: option "implicit_extern_c_type_conversion" can be used only when compiling C++

814: strict ANSI mode is incompatible with long preserving rules

815: type qualifier on return type is meaningless

> For example:
> __packed void foo(void) { }
> __packed is ignored here 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

820: option "extern_inline" can be used only when compiling C++

821: extern inline <entity-kind> "entity" was referenced but not defined

822: invalid destructor name for type "type"

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

825: virtual inline <entity-kind> "entity" was never defined

826: <entity-kind> "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-kind> "entity" has no corresponding operator deletexxxx (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-kind> "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:
> ```
> foo(void){
> int a;
> }
> ```
> Gives:
> ```
> Error:  #837-D: omission of explicit type is nonstandard ("int" assumed)
> ```
>
> An `int` result will be assumed. If you want it to return no result, use `void` as the return type. This is widespread in old-style C.
> The "`-Wv`" option suppresses this warning.

838: more than one partial specialization matches the template argument list of <entity-kind> "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-kind> "entity" is not used in template argument list of <entity-kind> "entity"

843: the type of partial specialization template parameter <entity-kind> "entity" depends on another template parameter

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-kind> "entity"

846: this partial specialization would have been made the instantiation of <entity-kind> "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-kind> "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-kind> "entity" is a pure virtual function

859: pure virtual <entity-kind> "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-kind> "entity"

864: xxxx 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 "xxxx" 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

874: option "embedded_c++" can be used only when compiling C++

875: Embedded C++ does not support templates

876: Embedded C++ does not support exception handling

877: Embedded C++ does not support namespaces

878: Embedded C++ does not support run-time type information

879: Embedded C++ does not support the new cast syntax

880: Embedded C++ does not support using-declarations

881: Embedded C++ does not support "mutable"

882: Embedded C++ does not support multiple or virtual inheritance

883: invalid Microsoft version number:

884: pointer-to-member representation "xxxx" has already been set for <entity-kind> "entity"

885: "type" cannot be used to designate constructor for "type"

886: invalid suffix on integral constant

887: operand of __uuidof must have a class or enum type for which __declspec(uuid("...")) has been specified

888: invalid GUID string in __declspec(uuid("..."))

889: option "vla" can be used only when compiling C

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-kind> "entity" is not a template

896: expected a template argument

898: nonmember operator requires a parameter with class or enum type

899: option "enum_overloading" can be used only when compiling C++

901: qualifier of destructor name "type" does not match type "type"

902: type qualifier ignored

903: option "nonstd_qualifier_deduction" can be used only when compiling C++

905: incorrect property specification; correct form is __declspec(property(get=name1,put=name2))

906: property has already been specified

907: __declspec(property) is not allowed on this declaration

908: member is declared with __declspec(property), but no "get" function was specified

909: the __declspec(property) "get" function "xxxx" is missing

910: member is declared with __declspec(property), but no "put" function was specified

911: the __declspec(property) "put" function "xxxx" is missing

912: ambiguous class member reference -- <entity-kind> "entity" (declared at line xxxx) used in preference to <entity-kind> "entity" (declared at line xxxx)

913: missing or invalid segment name in __declspec(allocate("..."))

914: __declspec(allocate) is not allowed on this declaration

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:

918: option "one_instantiation_per_object" can be used only when compiling C++

919: invalid output file: "xxxx"

920: cannot open output file: "xxxx"

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

922: option "one_instantiation_per_object" may not be used when compiling several input files

923: more than one command line option matches the abbreviation "--xxxx":

925: type qualifiers on function types are ignored

926: cannot open definition list file: "xxxx"

927: late/early tiebreaker option can be used only when compiling C++

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:

933: an import directory can be specified only in Microsoft mode

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

935: "typedef" may not be specified here

936: redeclaration of <entity-kind> "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.
> Example:
> main(void){
> int a;
> }
> If compiled with –strict the compiler reports:
>
> `Error: #938-D: return type "int" omitted in declaration of function "main"`
>
> 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 "`-Wv`" option suppresses this warning. For C++, this always results in an error.

939: pointer-to-member representation "xxxx" is too restrictive for <entity-kind> "entity"

940: missing return statement at end of non-void <entity-kind> "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 member template

948: nonstandard local-class friend declaration -- no prior declaration in the enclosing scope

949: specifying a default argument on this declaration is nonstandard

950: option "nonstd_using_decl" can be used only when compiling C++

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

957: option "designators" can be used only when compiling C

958: option "extended_designators" can be used only when compiling C

959: declared size for bit field is larger than the size of the bit field type; truncated to xxxx 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

973: "inline" used as a function qualifier is ignored

974: option "compound_literals" can be used only when compiling C

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-kind> "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-kind> "entity" is not compatible with <entity-kind> "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

1008: a parameter of a template template parameter cannot depend on the type of another template parameter

1009: <entity-kind> "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

1019: __irq 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 "xxxx" 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".
For Example:

    __global_reg(786) int x;

1025: invalid SWI number; 0 to 0xffffff allowed

SWI numbers are limited to the range 0 to 0xffffff for the ARM compilers, and 0 to 0xFF for the Thumb compilers. For standard "semihosting" SWIs, 0x123456 is used for ARM, 0xAB is used for Thumb.

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

1027: __swi_indirect function must have arguments

1028: conflicting global register declaration with <entity-kind> "entity" (declared at line xxxx)

1029: `__packed ignored for non-pointer parameter`

1030: `xxxx "type" previously declared without __packed`

1031: `Definition of "type" in packed "type" must be __packed`

The RVCT 2.0 Compiler Guide, section 3.1.3, 'Type qualifiers', says:
"All substructures of a packed structure must be declared using __packed."
This rule applies for all releases of RVCT, ADS and the earlier SDT 2.5x.

The compiler will fault 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 gives:
```
Error: #1031: Definition of "ChildStruct" in packed "ParentStruct" must be
__packed
```

1032: `Definition of nested anonymous xxxx in packed "type" must be __packed`

1033: `"xxxx" incompatible with function definition`

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

1037: `__global_reg is not valid on this declaration`

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

1039: `conflicting alignment declaration with <entity-kind> "entity" (declared at line xxxx)`

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;
}
```

is not allowed for a local variable foo, so the warning is given.

1042: `<entity-kind> "entity" (declared at line xxxx) cannot be dynamically initialized when compiled position independent`

1043: `<entity-kind> "entity" cannot be const because it contains a mutable member`

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

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

1046: `<entity-kind> "entity" has no member class "xxxx"`

1047: `the global scope has no class named "xxxx"`

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

1052: option "ignore_std" can be used only when compiling C++

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-kind> "entity" cannot be declared in this scope

1064: the reserved identifier "xxxx" may only be used inside a function

1065: this universal character cannot begin an identifier

1066: expected a string literal

1067: unrecognized STDC pragma

1068: expected "ON", "OFF", or "DEFAULT"

1069: a STDC pragma may only appear between declarations in the global scope or before any statements or declarations in a block scope

1070: incorrect use of va_copy

1071: xxxx 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

1076: conversion between real and imaginary yields zero

1077: an initializer cannot be specified for a flexible array member

1078: imaginary *= imaginary sets the left-hand operand to zero

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

1080: a definition is required for inline <entity-kind> "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

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

This error is given by the inline assembler if the coprocessor number is accidentally omitted from an MCR or MRC instruction or is an invalid coprocessor number (not 0-15), e.g:
```
MRC     0, r0, c1, c0, 0
```
instead of:
```
MRC     p15, 0, r0, c1, c0, 0
```

1112: Expected a coprocessor register name

1113: Inline assembler not permitted when generating Thumb code

The Thumb inline assembler was supported in ADS 1.2, but support was withdrawn in RVCT 2.0. ARM inline assembly continues to be supported. The Thumb Instruction Set was designed based on the output of the C compiler, and so there should be no need to write explicitly in Thumb inline assembler. Please contact your supplier if you need more information.

1114: This instruction not supported on target architecture/processor

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-kind> "entity" has no member function "xxxx"

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 `<Number>` is too large to be represented in a signed long, and therefore has been treated as a (signed) `long long`

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

gives the warning:
```
#1134-D: literal treated as "long long"
```

because 2147483648 is one greater than the maximum value allowed for a signed long.
The "`ll`" suffix means that the constant will be treated as a (64-bit) "`long long`" type rather than a signed long. See section 3.2.2 of the RVCT 2.0 Compilers and Libraries Guide.

To eliminate the warning, explicitly add the "`ll`" or "`LL`" suffix to your constants, e.g.:
```
int foo(unsigned int bar)
{
    return (bar == 2147483648LL);
}
```

1135: `literal treated as "unsigned long long"`

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

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-kind> "entity" is incompatible with a declaration in another translation unit

1172: the other declaration is at line xxxx

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

1176: declaration of <entity-kind> "entity" had a different meaning during compilation of "xxxx"

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: "xxxx"

1185: <entity-kind> "entity" already defined during compilation of "xxxx"

1186: <entity-kind> "entity" already defined in another translation unit

1187: a non-static local variable may not be used in a __based specification

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 "xxxx" is corrupted

1194: <entity-kind> "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-kind> "entity"

1197: no instance of <entity-kind> "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-kind> "entity"

1203: invalid attribute for parameter

1204: attribute "xxxx" does not take arguments

1206: expected an attribute name

1207: attribute "xxxx" ignored

1208: attributes may not appear here

1209: invalid argument to attribute "xxxx"

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: xxxx

1216: attribute "xxxx" 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-kind> "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 "xxxx" 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 "xxxx"

1230: modifier letter 'xxxx' ignored in asm operand

1231: unknown asm constraint modifier 'xxxx'

1232: unknown asm constraint letter 'xxxx'

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 10)

1237: too many colons in asm statement

1238: register "xxxx" used more than once

1239: register "xxxx" is both used and clobbered

1240: register "xxxx" clobbered more than once

1241: register "xxxx" has a fixed purpose and may not be used in an asm statement

1242: register "xxxx" 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-kind> "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

1252: more than one preinclude option specified

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: "xxxx" is not a base class member

1258: __super cannot appear after "::"

1259: __super may only be used in a class scope

1260: __super must be followed by "::"

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 xxxx should be defined as a variable

1268: declaration aliased to unknown entity "xxxx"

1269: declaration does not match its alias <entity-kind> "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 ignored

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

1280: the "short_enums" option is only valid in GNU C and GNU C++ modes

1281: invalid export information file "xxxx" at line number xxxx

1282: variable xxxx 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 xxxx has not been defined

1286: LDRD/STRD instruction will be expanded

1287: LDM/STM instruction may be expanded

1288: Implicit ARM register "xxxx" 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 mistyped equality operator (==).
In either case if the operator is intended adding an explicit comparison against 0 may silence the warning.
This warning can be suppressed with -Wa.

Example:

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

Gives:
Warning: #1293-D: assignment in condition

1294: Old-style function xxxx

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 xxxx - 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 "`-Wd`" 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 initialiser may not be portable.
This warns that there is a constant that does not follow the strict rules of ANSI C even though there is a clause to allow it in the ANSI C specification.
Example compiled with –C90 switch:
```
const int foo_table[]  = { (int)"foo", 0, 1, 2};
```
This is not ANSI C standard compliant. Compiling with "`-We`" will suppress the warning.

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 `-W+g`.

1298: Header file is guarded by 'xxxx', but does not #define it

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

1300: xxxx 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 C++, the "`-Wr`" option suppresses the implicit virtual warning.
For example:
```
struct Base { virtual void f(); };
struct Derived : Base { void f(); };
```
gives:
```
Warning:  #1300-D: f inherits implicit virtual
  struct Derived : Base { void f(); };
                              ^
```
Adding the `virtual` keyword in the derived class prevents the warning.

1301: padding inserted in struct xxxx

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 "`-W+s`".
Example:

```
struct X{
char x;
int y;
}
```
gives:

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

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)"

1306: cannot open ASM output file

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

1315: THREADS not allowed in this context

1316: block size specified exceeds the maximum value of xxxx

1317: function returning shared is not allowed

1318: only arrays of a shared type can be dimensioned to a multiple of THREADS

1319: one dimension of an array of a shared type must be a multiple of THREADS when the number of threads is nonconstant

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

1321: parameters may not have shared types

1322: a dynamic THREADS dimension requires a definite block size

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 at line xxxx

1336: the prototype declaration of <entity-kind> "entity" (declared at line xxxx) is ignored after this unprototyped redeclaration

1337: attribute ignored on typedef of class or enum types

1338: <entity-kind> "entity" (declared at line xxxx) must have external C linkage

1339: variable declaration hides declaration in for-initializer

1340: typedef "xxxx" may not be used in an elaborated type specifier

1341: call of zero constant ignored

1342: parameter "xxxx" may not be redeclared in a catch clause of function try block

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

1344: "cc" clobber ignored

1345: "template" must be followed by an identifier

1346: MYTHREAD not allowed in this context

1347: layout qualifier cannot qualify pointer to shared

1348: layout qualifier cannot qualify an incomplete array

1349: declaration of "xxxx" 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 its base class

1353: GNU C++ compilers may use bit field padding

1354: memory usage conflict with precompiled header file "xxxx"

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-kind> "entity" using address of xxxx may cause link failure -ropi

See **1359**

1358: static initialisation of extern const <entity-kind> "entity" using address of xxxx cannot be lowered for ROPI

1359: static initialisation of <entity-kind> "entity" using address of xxxx may cause link failure -rwpi

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

```
char *str = "test";  /* global pointer */
```
when compiled with –apcs /ropi gives:
```
Warning:  #1357-D: static initialisation of variable "str" using address of
string literal may cause link failure -ropi
```
because the global pointer "str" will need to be initialized to the address of the char string "test" in the .constdata section, but absolute addresses cannot be used in a PI system.

```
int bar;
int *foo = &bar;      /* global pointer */
```
when compiled with –apcs /rwpi gives:
```
Warning:  #1359-D: static initialisation of variable "foo" using address of bar
may cause link failure -rwpi
```
because the global pointer "foo" will need to be initialized to the address of "bar" in the .data section, but absolute addresses cannot be used in a PI system.

The workaround is to change your code to avoid use of a global pointer, e.g. use a global array or local pointer instead.

See also ADS FAQ "What does "Error: L6248E: cannot have address type relocation" mean?" at:
http://www.arm.com/support/ads_faq

1360: `static initialisation of extern const <entity-kind> "entity" using address of xxxx cannot be lowered for RWPI`

1361: `Type of result operand is narrower than actual result`

The following old style error and warning messages can still be given:

C2005E: `an __irq function cannot call functions that use stack checking`

Interrupt handlers can be written in C using the compiler keyword __irq, however, this should be used with care. This is because the IRQ handler will be executed in IRQ mode rather than e.g. User mode, so any stack accesses will use the IRQ stack rather than the User stack. Remember to initialize the stack pointer for IRQ mode (SP_IRQ)!   Also, do not compile with '-apcs /swst', because the IRQ function will not be compiled with a stack check. It must not call a subroutine in IRQ mode which has been compiled with stack checking because of the risk that SL (Stack Limit) has not been set up for IRQ mode.

C2012U: `Too many symbols in object file`

In ADS 1.2, RVCT 1.2, RVCT 2.0 and later the compiler limit for the number of symbols in an object is 2^{24}.

C2033E: `R14 corrupted but possibly reused later. This code may not work correctly`

Example:
```
unsigned int foo(void)
{
    unsigned int linkReg;
    __asm{ mov linkReg, r14 }
    return linkReg;
}
```
The compiler is warning that the code may not behave as expected. In particular, r14 may not always contain the "return address" at that point, because the compiler may have inlined the function, or may have pushed LR onto the stack to be able to re-use r14 for temporary storage.

C2056W: `illegal unaligned load or store access - use __packed instead`

C2057E: `-fpu xxx is incompatible with selected -cpu option`

Example: `armcc –cpu arm10200 –fpu fpa`
will fail because the arm10200 contains a vfp, not fpa.

C2059E: `apcs /interwork is only allowed when compiling for processors that support Thumb instructions`

Example:
```
armcc -c -apcs /interwork -cpu strongarm1 main.c
```
will fail because the StrongARM processor does not support Thumb

C2060E: `specified processor or architecture does not support Thumb instructions`
Example:

`tcc -c -cpu strongarm1 main.c`

will fail because the StrongARM processor does not support Thumb

C2067I: `option -zas will not be supported in future releases of the compiler`
The "`-zas`" option is provided for backward compatibility only.
This warning is enabled by default, but may be suppressed with the "`-Wy`" switch.
Refer to the compiler documentation for more information about "`-zas`" and "`-Wy`".

C2068E: `PSR Uninitialised or corrupted use of PSR. This code may not work correctly`
See **C2033E**

C2070W: `Memory access attributes below the minimum requirement for ARM/Thumb`
An invalid `-memaccess` option has been specified

C2075W: `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.

C2619E: `Unbalanced pragma pop, ignored`
"#pragma push" and "#pragma pop" save and restore the current pragma state.
A pop must be paired with a push. An error is given for e.g.:

```
#pragma push
:
#pragma pop
:
#pragma pop
```

C2801W: `unknown option '<option>': ignored`

C2874W: `<name> may be used before being set`

C2885W: `division by zero`
Constant propagation shows that a divide or remainder operator has a second operand with value `0`.
It will be an error if execution reaches this expression.

C3039E: `I/O error on object stream`
C3041U: `I/O error writing`

C3046U: `out of store (for error buffer)`

C3047U: `Too many errors`

C3048U: `out of store [cc_alloc(N)] while compiling -g`
C3049U: `out of store [cc_alloc(N)]`
A storage allocation request by the compiler failed. Compilation of the debugging tables requested
with the `-g` option may require a great deal of memory. Recompiling without `-g`, or with the
program broken into smaller pieces, may help.

C3050U: `Compilation aborted.`

C3051E: `couldn't write file 'filename'`
C3052E: `couldn't read file 'filename'`
C3055E: `internal fault in copy_unparse`

C3056E: `bad option`

C3057E: `bad option`

For example, the switches "-apcs /softfp", " -apcs /narrow", "-apcs /wide" which were supported in SDT, are no longer supported in ADS or RVCT and so must be removed from the compiler command-line.

C3059E: `Missing file argument for '<option>'`

<option> requires a file parameter, e.g. `-errors err.txt`

C3060E: `No argument to compiler option`

C3061E, C3062E, C3063E: `unknown option`

Examples:
```
Error: C3061E: unknown option '-fz': ignored
Error: C3063E: unknown option '-za1': ignored
```
These compiler options were commonly used in build scripts for SDT, however these are no longer supported by ADS or RVCT. You should remove these switches from any makefiles.

C3064E: `Overlong filename: filename`

C3065E: `type of '<variable>' unknown`

C3066E: `The code space needed for this object is too large for this version of the compiler`

C3067E: `Couldn't write installation configuration`

C3074E: `Can't open -via file filename`
C3075E: `Can't open filename for output`

C3078E: `stdin ('-') combined with other files`

C3079E: `command with no effect`

C3396E: `Source file-name 'filename' cannot be configured`

C3403E: `__alloca_state not defined`

C3410W: `Option <option> has been obsoleted`

For example:
```
armcc -dwarf1
Error: C3410W: Option -dwarf1 has been obsoleted
```
Use –dwarf2 instead

C3421W: `write to string literal`

There is a write through a pointer, which has been assigned to point at a literal string. The behaviour is undefined by to the ANSI standard; a subsequent read from the location written may not reflect the write.

C3433E: `<option> selection is incompatible with restrictions in '<constraintfile>'`

The feature-restricted toolkit that uses a 'constraints file' is not installed correctly. Try re-installing.

C3435E: `reference to FOO not allowed`

C3455E: `cannot form pointer to <function> function`

C3463E: `Invalid combination of memory access attributes`

C3464E: `Maximum pointer alignment must be a power of 2`

C3465E: The in-memory file system is obsolete, use the normal include mechanisms

The compiler switches -I and -J control the searching order for header files.

In ADS, "-I- and "-J-" were used to force the searching of the "in-memory file system" (where the ARM-supplied standard C \include header files were built-into the compiler). These allowed you to specify e.g. "armcc -J special_dir -J-", to force the compiler to search "special_dir" before the "in-memory file system".

The "in-memory file system" was a part of the old ADS compiler front-end, but is no longer present in the new RVCT 2.0 front-end.
In RVCT 2.0, use of "-I-" or "-J-" now results in the compiler reporting:
"Fatal error: C3465E: The in-memory file system is obsolete, use the normal include mechanisms".

To convert your ADS makefiles or build scripts to work with RVCT 2.0, you will need to remove any occurences of "-I- or "-J-" and replace them with "-Idir or "-Jdir" as described in the RVCT 2.0 Compiler and Libraries Guide, section 2.2.2, "Included files".

For application header files (#include "file.h"), the Idirs are searched first then Jdirs (and then RVCT20INC or ARMINC if set).
For system header files (#include <file.h>), the Jdirs are searched first (then RVCT20INC or ARMINC if set) then Idirs.

If you need to replace the ARM-supplied system header files (e.g. <stdio.h>) with your own, and force the compiler to search your own include directories before the ADS-supplied \include directories, then compile using -I and -J on the command-line like this (for Windows):

```
-I"path to your application (double quote) headers" -J"path to your system
(angle bracket) headers,%RVCT20INC%"
```

For Unix, replace %RVCT20INC% with $RVCT20INC.

If your application headers and system headers are in the same directory, then you can use just -J on its own like:

```
-J"c:\myinclude,%RVCT20INC%"
```

Note that the environment variable RVCT20INC is, by default, a path containing spaces (e.g. C:\Program Files\ARM\RVCT\Data\..), so needs to be enclosed within double quotes.

RVCT 2.0 provides a new feature - 'Precompiled header files' which offers similar speed benefits to the old "in-memory file system". When compiling single source files with the --pch option, header files are only compiled once. Subsequent compilations are then performed faster by using the precompiled files. Please see Section 2.2.2 and Section 2.2.3 of the RVCT 2.0 Compiler and Libraries Guide for more information.

C3466W: Feedback line ignored, unrecognised pattern

C3473E: unknown CPU <cpu>
C3481E: Bad --diag_style argument style

3. ARM Assembler (armasm) Errors and Warnings

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 wish to obtain the offset of a label called `<label>` within an area called `<areaname>`, use `<label>-<areaname>`. See RVCT 2.0 Assembler Guide, section 3.6.10, "Unary operators"

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 may need to be escaped depending upon operating system and shell. For example:
>   `-predefine "versionstr SETS \"5A\""`

A1021U: `No input file`
>   No input file was specified on the command line. This may be because there was no terminating quote on a quoted argument.

A1023E: `File "<filename>" could not be opened`
>   The given file either did not exist, could not be found, or could not be opened.
>   Check that the correct path for the file is specified.

A1024E: `File "<filename>" could not all be loaded`
>   An error occurred when trying to read the file.
>   A disk error or a file sharing conflict could cause this.

A1042E: `Unrecognized APCS qualifier '<qualifier>'`
>   There is an error in the argument given to the `-apcs` command line option.
>   Check the spelling of `<qualifier>`.

A1046E: `Via file '<filename>' would not open`
>   The file given in the `-via <filename>` command line option either did not exist, could not be found, or could not be opened.
>   Check that the correct path for the file is specified.

A1047E: `Missing argument to '<option>'`
>   No argument was given for the command line option `<-option>`.

A1048E: `Bad fpu specified`
>   The argument to the `-fpu` command line option was not recognized, check the spelling of the argument.

A1051E: `Cannot open -depend file '<filename>'`
>   The file given in the `-depend <filename>` command line option either did not exist, could not be found, or could not be opened. Check that the correct path for the file is specified.

A1055E: `Cannot open -errors file '<filename>'`
>   The file given in the `-errors <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.

A1056E: `Target cpu <cpu> not recognized`

The name given in the `-cpu <cpu>` command line option was not a recognized processor name. Check the spelling of the argument.

A1057E: `Target cpu missing`

No argument was given for the `-cpu (or -proc)` command line option.

A1058E: `Input file specified as '<file1>', but it has already been specified as '<file2>'`

More than one input file has been specified on the command line. Misspelling a command line option can cause this. Only one input file may be given on the command line. To assemble multiple files, it is necessary to invoke the assembler multiple times.

A1063W: `Listing file page length out of range 0 to 255, ignored`

The argument to the `-length` command line option was either negative, or too large. The value of the argument should be 0 (which specifies an unpaged listing), or a positive value less than 256.

A1065E: `Bad value for -maxcache`

Check the argument to the `-maxcache` command line option.

The argument must be a positive integer, either decimal or hexadecimal. Hexadecimal values must be preceded by `0x` or `&`. Note that `&` is recognized by some command line interpreters as a special character, and so may need to be escaped.

A1067E: `Output file specified as '<file1>', but it has already been specified as '<file2>'`

More than one output file has been specified on the command line. Misspelling a command line option can cause this.

A1071E: `Cannot open listing file '<filename>'`

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 may 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 may be because the object filename was accidentally omitted from the command line.

A1074E: `The specified depend file '<filename>' must not be a source file`
A1075E: `The specified errors file '<filename>' must not be a source file`

The filename argument to the `-depend` / `-errors` command line option has an extension that indicates it is a source (.s) file. This may be because the filename argument was accidentally omitted from the command line. Check that the correct arguments are given.

A1077W: `Width out of range 1 to 255, ignored`

The argument to the `-width` command line was either too large or too small.

The `-width` option is ignored, and the default page width is used instead. This warning may be produced because the argument to the `-width` command has been accidentally omitted from the command line, or is not a decimal number.

A1079E: `Unrecognized command line option '<option>'`

        <option> is not a valid command line option. Check the spelling of <option>.

A1080E: `Bad command line operand '<operand>'`

        The only operands that may be specified on the command line are the source file and the object file, in that order. Any subsequent operands will be flagged with this error.

        This error could be caused by attempting to specify multiple source files on the command line (this is not allowed), or by misspelling a command line option or option argument.

A1085E: `Forced user-mode LDM/STM must not be followed by use of banked R8-R14`

        The ARM architecture does not allow you to access the 'banked' registers on the instruction following a 'USER registers' LDM or STM. The ARM ARM (2nd Ed), section 4.1.18, says: "Banked registers: 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`

A1100E: `Structure stack underflow`

A1105E: `Area directive missing`

        This is given when no AREA is given (see **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, e.g. SETA. See the RVCT 2.0 Assembler Guide, section 3.6.3, "Numeric expressions"

A1111E: `Expected constant or address expression`
A1112E: `Expected address expression`

A1113E: `Expected string expression`

        A string expression was expected after, e.g. SETS. See the RVCT 2.0 Assembler Guide, section 3.6.1, "String expressions"

A1114E: `Expected register relative expression`

        Examples:

        The generic form: `LDR r4,[r9,offset]`

        must be rewritten as: `LDR r4,[r9,#offset]`

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 RVCT 2.0 Assembler Guide, section 2.9, "Using macros".

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>`
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, e.g:
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, which are not part of an (ARM) instruction, are found on an instruction line, for example:
`ADD r0, r0, r1 comment`
Could 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 ELF format output`
This can occur 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.

These 'subtractive relocations' were allowed with SDT AOF, but not with ELF, so this error message can sometimes appear when migrating an SDT project to ADS or RVCT.

To resolve this change your code to use the simpler, equivalent syntax:
```
     LDR r0, label
```
This works in both cases of 'label' being either in the same area or in a different area.

A1145E: Undefined exported symbol <sym>

A1146E: Unable to open output file <codeFileName>

A1147E: Bad shift name
A1148E: Unknown shift name <name>, expected one of LSL, LSR, ASR, ROR, RRX

A1149E: Shift option out of range
      Example:
```
mov    r0, r0, LSR #0x0
add    r0, r0, r1, LSR #0x0
```

      Strictly, according to the ARM Architecture Reference Manual, LSR #0 does not exist. You should use LSL #0, or even just omit the shift as:
```
mov    r0, r0
add    r0, r0, r1
```

      Please see the ARM Architecture Reference Manual 2nd edition, section 5.1.7, "Data-processing operands - Logical shift right by immediate"

A1150E: Bad symbol, not defined or external
      This typically occurs in two cases:

      1) when the current file requires another file to be INCLUDEd 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, e.g:
```
     INCLUDE     targets/eb40.inc
```

      2) when the current file requires some symbols to be IMPORTed, for example:
```
"init.s", line 4: Error: A1150E: Bad symbol
   4 00000000       LDR r0, =||Image$$RAM$$ZI$$Limit||
```
typically requires the symbol to be imported, e.g:
```
     IMPORT ||Image$$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 biuld. The ARM Register can be 'r' or 'R', hence:
```
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, e.g. ENTRY, IMPORT, EXPORT, GET must be on a line without a label at the start of the line. This error will be given if a label is present.

A1159E: Label missing from line start

Some directives, e.g. FUNCTION or SETS, require a label at the start of the line, for example:
my_func FUNCTION
or
label SETS
This error will be given if the label is missing.

A1160E: Bad local label number

A local label is a number in the range 0-99, optionally followed by a name. See RVCT 2.0 Assembler Guide, section 3.5.6, "Local labels."

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:
1) Use of a hardware floating point instruction without using the –fpu switch, for example:
       FMXR FPEXC, r1 ; must be assembled with armasm –fpu vfp
   or
       LDFD f0, [r0] ; must be assembled with armasm –fpu fpa
2) 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
3) Mis-typing the opcode, e.g ADDD instead of ADD

A1164E: Opcode not supported on selected processor

The processor selected on the armasm command line does not support this instruction. Check the ARM Architecture Reference Manual, section 4.2, "ARM instructions and architecture versions". This may occur when attempting to use halfword instructions on an old architecture that does not support halfwords, e.g. "STRH r0,[r1]" assembled with "-cpu 3"

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 0xX out of range for this operation must be below (0xX)

This error is given if a MOV or MVN instruction is used with a constant that cannot be assembled. See RVCT 2.0 Assembler Guide, section 2.6.1, "Direct loading with MOV and MVN".

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 may only be used with labels within the same code
    section. 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 "local labels", you can use the
    ROUT directive to limit the scope of local labels, to help avoid referring to a wrong label by
    accident. See RVCT 2.0 Assembler Guide, section 3.5.6, "Local labels".

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

A1187E: Error in macro parameters

A1188E: Register value <val> out of range. Permitted values are <mini> to <maxi>

A1189E: Missing '#'

A1190E: Unexpected '<character>'

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

A1197W: Precision specifier ignored for 'FIX'
    Unlike some of the other FPA instructions, "FIX" has no precision specifier

A1198E: Unknown operand
    This can occur when an operand is accidentally mistyped, for example:
    armasm init.s -g -PD "ROM_RAM_REMAP SETL {FALS}"

```
       should be:
       armasm init.s -g -PD "ROM_RAM_REMAP SETL {FALSE}"
```
See RVCT 2.0 Assembler Guide, section 3.5.4, "Assembly time substitution of variables"

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 RVCT 2.0 Assembler Guide, section 3.5.4, "Assembly time substitution of variables"

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 e.g. LDM or STM instructions are not specified in increasing order *and* the `-checkreglist` option is used.

A1207E: `Bad or unknown attribute`
Example:
`AREA test,CODE,READONLY,HALFWORD,INTERWORK`
The `HALFWORD` and `INTERWORK` attributes are obsolete - simply remove them.

A1209E: `ADRL can't 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 <Attr1> cannot be used with attribute <Attr2>`

A1214E: `Too late to define symbol <sym> as register list`

A1215E: `Bad register list symbol`

A1216E: `Bad string escape sequence`

A1219E: `Bad CPSR or SPSR designator`
For example:
`    MRS r0, PSR`
It is necessary to specify which status register to use (`CPSR` or `SPSR`), e.g:
`    MRS r0, CPSR`

A1220E: `BLX <address> must be unconditional`

A1234E: `Undefined or Unexported Weak 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 16 bit code to 32 bit code
A1246E: B or BL from 32 bit code to 16 bit code
> This occurs when there is a branch from ARM code (CODE32) to Thumb code (CODE16) (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 will add any necessary interworking veneers.

A1247E: BLX from 32 bit code to 32 bit code, use BL
A1248E: BLX from 16 bit code to 16 bit code, use BL
> This occurs when there is a BLX <label> branch from ARM code to ARM code (or from Thumb code to Thumb code) within this assembler file. This is not allowed because BLX <label> always results in a state change. 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 ARM ELF.
> Branch "B foo" (foo is an extern) is not allowed in Thumb assembler code.

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.

A1255E: Operand to LDRB <value> does not fit in 8 bits

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

A1261E: MRS cannot select fields, use CPSR directly
> This is caused by an attempt to use fields for CPSR or SPSR with an MRS instn, e.g:
> MRS r0, CPSR_c

A1262U: Expression storage allocator failed

A1265U: Structure mismatch, structure not GET
A1267E: Bad GET or INCLUDE for file <filename>

A1268E: Unmatched conditional or macro

A1270E: File "<filename>" 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.
> Refer to the RVCT 2.0 Assembler Guide, section 2.6.2, "Loading with LDR Rd, =const" and section 7.3.1, "LTORG". See **A1471W.**

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`

> The macro definition is invalid.

A1310W: `Symbol attribute not recognized`

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`

A1318W: `TSTP/TEQP/CMNP/CMPP inadvisable in 32-bit PC configurations`

> These obsolete 26-bit architecture instructions are no longer supported.

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`
A1323E: `Reserved instruction (Rm = Rn with post-indexing)`
A1324E: `Undefined effect (PC + writeback)`
A1325E: `Undefined effect (destination same as written-back base)`
A1326E: `Undefined effect (PC used in a non-word context)`

A1327W: `Non portable instruction (LDM with writeback and base in reg. list, final`
`value of base unpredictable)`

> See ARM Architecture Reference Manual (ARM ARM), section 4.1.17, "LDM", Operand restrictions:
> If the base register `<Rn>` is specified in `<registers>`, and base register writeback is specified, the final value of `<Rn>` is UNPREDICTABLE.

A1328W: `Non portable instruction (STM with writeback and base not first in reg. list,`
`stored value of base unpredictable)`

> See ARM Architecture Reference Manual (ARM ARM), section 4.1.42, "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 UNPREDICTABLE.

A1329W: Unsafe instruction (forced user mode transfer with write-back to base)

A1331W: Unsafe instruction (PC as source or destination)
A1332W: Undefined effect (PC-relative SWP)
A1334E: Undefined effect (use of PC/PSR)
A1335W: Useless instruction (PC can't be written back)
A1337W: Useless instruction (PC is destination)
A1338W: Dubious instruction (PC used as an operand)

A1339W: Undefined if any of RdLo, RdHi, Rm are the same register

A1341E: Branch to unaligned destination, expect destination to be `<max>` byte aligned

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
> ```
> not:
> ```
> IF :DEF: FOO
> ; code
> ENDIF
> ```
> Symbols in the left hand column 1 are assumed to be labels, hence the error message.

A1356W: Instruction not supported on targeted CPU

> This will occur if you try to use an instruction that is not supported by armasm's default
> architecture/processor, for example:
> ```
>     SMULBB  r0,r0,r1 ; may be assembled with armasm -cpu 5TE
> ```
> The processor selected on the armasm command line does not support this instruction. Check the
> ARM Architecture Reference Manual, section 4.2, "ARM instructions and architecture versions".

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
A1410E: This floating-point value cannot be specified as an immediate operand,
permitted constants are 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 0.5 and 10.0
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 for instruction, expect floating point/double
register type
A1419E: Scalar operand not in first 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 first bank`
A1426E: `Source vector must not be in first bank`
A1427E: `Operands have a partial overlap`
A1428E: `Register list contains registers of varying types`
A1429E: `Expected register list`
> The VFP instructions are malformed. See RVCT 2.0 Assembler Guide, section 6, "Vector Floating-point Programming"

A1430E: `Unknown frame directive`
A1431E: `Frame directives are not accepted outside of PROCs/FUNCTIONs`
> Invalid FRAME directive. See RVCT 2.0 Assembler Guide, 7.5, "Frame description directives"

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 (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 desired behaviour.

A1434E: `Architecture attributes '<attr1>' and '<attr2>' conflict`

A1435E: `{PCSTOREOFFSET} is not defined when assembling for an architecture`
> `{PCSTOREOFFSET}` is only defined when assembling for a processor, not for an architecture.

A1436E: `Memory access attributes '<attr1>' and '<attr2>' conflict`
> Use of incorrect `-memaccess` arguments

A1446E: `Bad or unknown attribute 'INTERWORK'. Use -apcs /interwork instead`
> Example:
> ```
>     AREA test1, CODE, READONLY
>     AREA test, CODE, READONLY, INTERWORK
> ```
>
> This code may have originally been intended to work with SDT. The `INTERWORK` area attribute is now obsolete. To eliminate the warning:
> a) remove the "`, INTERWORK`" from the AREA line.
> b) assemble with '`armasm -apcs /interwork foo.s`' instead

A1447W: `Missing END directive at end of file, but a label named END exists. Perhaps you intended to put this in a column other than 1.`

A1448W: `Deprecated form of PSR field specifier used (use _f)`
A1449W: `Deprecated form of PSR field specifier used (use _c)`
A1450W: `Deprecated form of PSR field specifier used (use _cxsf for future compatibility)`
> The ARM 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".
>
> Note that `MSR CPSR_c, #immediate` is a legitimate instruction (despite what is written in early versions of the ARM ARM), so a sequence of two instructions like:
> ```
>     MOV r0, #0x1F
>     MSR CPSR_c, r0
> ```

as commonly found in boot code, can be combined into one instruction, like:

```
MSR CPSR_c, #0x1F ; go to System mode, IRQ & FIQ enabled
```

Earlier releases of the assembler allowed 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
```

and similarly for SPSR.

These forms are now deprecated, so should not be used. If your legacy code contains them, the assembler will report "Deprecated form of PSR field specifier used (use _cxsf)"

To avoid the warning, in most cases you should simply modify your code to use '_c', '_f', '_cf' or '_cxsf' instead.

For more information, see FAQ "armasm: Use of MRS and MSR instructions ('Deprecated form of PSR field specifier')" at: http://www.arm.com/support/faq

A1454E: `FRAME STATE RESTORE directive without a corresponding FRAME STATE REMEMBER`
Invalid FRAME directive. See RVCT 2.0 Assembler Guide, 7.5, "Frame description directives"

A1456W: `INTERWORK area directive is obsolete. Continuing as if -apcs /inter selected.`
Example:

```
        AREA test, CODE, READONLY, INTERWORK
```

This code may have originally been intended to work with SDT. The INTERWORK area attribute is now obsolete. To eliminate the warning:

a) remove the "`, INTERWORK`" from the AREA line.

b) 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 same file. This code may have originally been intended to work with SDT. The INTERWORK area attribute is obsolete in RVCT.

Example:

```
        AREA test1, CODE, READONLY
        :
        AREA test2, CODE, READONLY, INTERWORK
```

To eliminate the error:

a) move the two AREAs into separate assembler file, e.g. test1.s and test2.s

b) remove the "`, INTERWORK`" from the AREA line in test2.s

c) assemble test1.s with `'armasm -apcs /nointerwork'`

d) assemble test2.s with `'armasm -apcs /interwork'`

e) at link time, the linker will add 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 allowed – consult the ARM ARM for the allowed forms.

A1460W: `<badopt> is deprecated and will not be supported in future releases of the assembler: use -cpu`
Simply change "`-arch`" or "`-proc`" to "`-cpu`"

A1461E: `specified processor or architecture does not support Thumb instructions`
Example:

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, added in ADS 1.1, may help C programmers from being caught out when writing in assembler.
To avoid the warning, modify the code to make the evaluation order explicit (i.e. add more brackets), or suppress the warning with '`-unsafe`' switch.
See RVCT 2.0 Assembler Guide, section 3.6.9, "Operator precedence".

A1467W: `FRAME 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`
Invalid FRAME directive. See RVCT 1.2 Assembler Guide, 7.5, "Frame description directives"

A1471W: `directive LTORG may be in an executable position`
This can occur with e.g. the `LTORG` directive (see **A1283E** & **A1284E)**. `LTORG` instructs the assembler to dump literal pool DCD data at this position. The data must be placed where the processor cannot execute them as instructions, otherwise this warning is given. 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
```

A1472E: `Cannot load constraints file for feature set <filename>`
A1473E: `<option> selection is incompatible with restrictions in '<constraintfile>'`
The feature-restricted toolkit that uses a 'constraints file' is not installed correctly. Try re-installing.

A1475W: `At least one register must be transferred, otherwise result is UNPREDICTABLE.`

A1476W: `BX r15 at non word-aligned address is UNPREDICTABLE`

A1477W: `This register combination results in UNPREDICTABLE behaviour`

A1479W: `Requested ALIGN is greater than area alignment <align>, which has been increased`
This is warning about an ALIGN directive which has a coarser alignment boundary than its containing AREA, which is not allowed. 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 will be on a 16 byte boundary, hence the error. (Note the difference in how the two

alignment types are specified). These two types of alignment control are described in detail in the RVCT 2.0 Assembler Guide, section 7.7.1, "ALIGN" and 7.7.2, "AREA".

A1480W: `Macro cannot have same name as a directive or instruction`

A1481E: `Object file format does not support this area alignment`

A1482E: `Shift option out of range, allowable values are from <min> to <max>`

A1483E: `Obsolete command line option '<option>'`
`<option>` is no longer a valid command line option.

A1484E: `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 4 shift types: ROR, ASR, LSR, and LSL. An arithmetic (i.e. signed) shift left is the same as a logical shift left, because the sign bit always gets shifted out. Earlier versions of the assembler would silently convert ASL to LSL. This error can be downgraded to a warning by using the "`-unsafe`" switch.

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 may only be used with labels within the same code section. To load an out-of-area address into a register, use LDR instead.

A1487E: `Obsolete instruction name 'ASL', use LSL instead`
The Thumb instruction ASL is now faulted. See the corresponding ARM ASL message A1484E.

A1488W: `PROC/FUNC at line <lineno> without matching ENDP/ENDFUNC`

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`
A1494E: `Via file limit exceeded.. This might be due to recursive via files`
A1495E: `Target of branch is a data address`

A1496E: `Absolute relocation of ROPI address with respect to symbol <X> at offset 0x<X> 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.

A1497E: `Absolute relocation of RWPI address with respect to symbol <X> at offset 0x<X> 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>

A1572E: Operator SB_OFFSET_11_0 only allowed on LDR/STR instuctions

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

A1579E: Bad command line argument <arg1> <arg2>

A1581W: Added <no_padbytes> bytes of padding at address <address>

A1582E: Link Order area <name> undefined

A1583E: Group symbol <name> undefined

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> conflict

A1543W: Empty macro parameter default value

A1544W: Invalid empty PSR field specifier, field must contain at least one of c,x,s,f

A1545E: Too many sections for one ELF file

4. ARM Linker (armlink) Errors and Warnings

All linker warnings are suppressible with "--diag_suppress" in the same way as for compiler warnings, e.g "--diag_suppress 6306".

Two downgradeable errors are also suppressible: L6220E and L6238E.

L6000U: `Out of memory.`
This may occur when linking very large objects/libraries together, or you have very large regions defined in your scatter-file. In these cases, your workstation may run out of (virtual) memory.

L6001U: `Could not read from file <filename>.`
L6002U: `Could not open file <filename>.`
L6003U: `Could not write to file <filename>.`
An file I/O error occurred while reading/opening/writing to the specified file.

L6004U: `Missing library member in member list for <library>.`
L6005U: `Extra characters on end of member list for <library>.`

L6007U: `Could not recognize the format of file <filename>.`
The linker can recognize object files in ELF and AOF formats, and library files in AR and ALF 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 written in ELF and AOF file formats. The specified library member is either corrupt, or is written in a file format that the linker cannot recognize.

L6009U: `File <filename> : Endianness mismatch.`
The endianness of the specified file/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>.`
An file I/O error occurred while reading /opening/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 '`&`' or '`0x`' or '`0X`'. A suffix of '`k`' or '`m`' can be used to specify a multiple of `1024` or `1024*1024`.

L6012U: `Missing argument for option '-<option>'.`
The specified option requires an argument.

L6013U: `Relocation #NN in <objname>(<secname>) has invalid/unknown type.`
See **L6027U**

L6014U: `Unrecognised option -<option>.`
The linker does not recognize this option. This could be due to a spelling error, or due to the use of an unsupported abbreviation of an option.

L6015U: `Could not find any input files to link.`
The linker must be provided with at least one object file to link.
Example:
If you try to link with

```
armlink -o foo.axf
```
you will get the above error. Instead, you must use, for example:
```
armlink foo_1.o foo_2.o -o foo.axf
```

L6016U: `Symbol table missing/corrupt in object/library <object>.`
The specified object (or library) does not have a valid symbol table. It may be corrupt - try rebuilding it.

L6017U: `Library <library> symbol table contains an invalid entry.`
The library may 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.`

L6021U: `Symbol <symbol> has unsupported attribute <attribute>.`
L6022U: `Object <objname> has multiple <table>.`
L6023U: `<objecttype> object <objname> does not contain any <part>.`
The object file is faulty or corrupted. This may indicate a compiler fault – please contact your supplier.

L6024U: `Library <library> contains an invalid member name.`
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`

L6027U: `Relocation #NN in <objname>(<secname>) has invalid/unknown type.`
This may occur in rare cases when linking legacy SDT AOF objects with the RVCT linker. For backward compatibility, the linker is able to accept most object files in the SDT AOF format and libraries in the SDT ALF format. These formats are obsolete and will not be supported in the future. However, not all AOF relocations are recognized in RVCT. Some obscure AOF relocations cannot be translated into ELF, and are faulted. If so, the linker may report e.g.:
Error : (Fatal) L6027U: Relocation #17 in obj.o (SYMBOL_NAME) has invalid/unknown type.
To resolve this, the object/library must be rebuilt with RVCT.

L6028U: `Relocation #NN in <objname>(<secname>) has invalid offset.`
The relocation has an invalid offset. This may indicate a compiler fault – please contact your supplier.

L6029U: `Relocation #NN in <objname>(<secname>) is wrt invalid/missing symbol.`
The relocation is with respect to a symbol, which is either invalid or missing from the object symbol table, or is a symbol that is not suited to be used by a relocation. This may indicate a compiler fault – please contact your supplier.

L6030U: `AOF area <objname>(<areaname>) has unsupported attribute <attribute>.`
The object file was built with SDT, probably in assembler, using the old (ARM proprietary) AOF format, which is now obsolete. The linker is able to recognise most AOF relocations, but not all the old attributes. To resolve this, the object/library must be rebuilt with RVCT.

L6031U: `Could not open scatter description file <filename>.`
An I/O error occurred while trying to open the specified file. This could be due to 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.`

The object file is faulty or corrupted. This may indicate a compiler fault – please contact your supplier.

L6034U: `Symbol <symbolname> in <objname> has invalid value.`
This can be caused by a section relative symbol having a value that exceeds the section boundaries. This may indicate a compiler fault – please contact your supplier.

L6035U: `Relocation #NN 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>.`
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 due to a spelling error, or due to 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.

L6039U: `Multiple -entry options cannot be specified.`
Only one instance of `-entry` is permitted, to specify the unique entry point for the ELF image.

L6040U: `Object <objname> contains corrupt symbol table entry for symbol <symbolname>.`
The object file is faulty or corrupted. This may indicate a compiler fault – please contact your supplier.

L6041U: `An internal error has occurred (<clue>).`
Contact your supplier.

L6042U: `Relocation #NN in <objname>(<secname>) is wrt a mapping symbol`
Relocations with respect to mapping symbols are not allowed. This may indicate a compiler fault – please contact your supplier.

L6043U: `Relocation Relocation #<rel_number> in <objname>(<secname>) is wrt 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.

L6046U: `Recursive via file inclusion depth of <limit> reached`

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 allowed 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 allowed to have a negative max-size.

L6188E: `Special section <sec1> multiply defined by <obj1> and <obj2>.`

L6195E: `Cannot specify both '<attr1>' and '<attr2>' for region <regname>`

L6199E: `Number string '<number>' contains invalid character(s) '<badchar>'.`
Number must not contain characters that are not valid digits for the base.

L6200E: `Symbol <symbol> multiply defined (by <object1> and <object2>).`
There are two common examples where this occurs:

1) `Symbol __semihosting_swi_guard multiply defined (by use_semi.o and use_no_semi.o).`
This error is reported when functions that use semihosting SWIs are linked in from the C library, in the presence of the `__use_no_semihosting_swi` guard. See the RVCT 2.0 Compilers and Libraries Guide, section 5.2.2, "Building an application for a nonsemihosted environment" and RVCT 2.0 Developer Guide, Section 2.3.2, "Avoiding C library semihosting".

To resolve this, you must provide your own implementations of these C library functions. The RVCT 2.0 \emb_sw_dev directory contains examples of how to re-implement some of the more common SWI-using functions - see the file `retarget.c`.

To identify which SWI-using functions are being linked-in from the C libraries:
1. Link with '`armlink -verbose -errors err.txt`'
2. Search err.txt for occurrences of '`__I_use_semihosting_swi`'
For example:
```
:
Loading member sys_exit.o from c_a__un.l.
            reference :   __I_use_semihosting_swi
            definition:   _sys_exit
:
```
This shows that the SWI-using function _sys_exit is being linked-in from the C library. To prevent this, you will need to provide your own implementation of this function.

2) `Symbol __stdout multiply defined (by retarget.o and stdio.o).`
This means that there are two conflicting definitions of `__stdout` present – one in `retarget.o`, the other in `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`, `fflush`, etc. `stdio.o` is being linked-in because it satisfies some unresolved references.

To identify why stdio.o is being linked-in, you must link with the linker's "verbose" switch, e.g.:
```
   armlink [... your normal options...] -verbose -errors err.txt
```
Then study `err.txt`, so see exactly what the linker is linking-in, from where, and why.

To move forward, the user may have to either:
- Eliminate the calls like `fopen`, `fclose`, `fflush`, etc, or
- Re-implement the `_sys_xxxx` family of functions.

See the RVCT 2.0 Compilers and Libraries Guide, section 5.10, "Tailoring the input/output functions".

L6201E: `Object <objname> contains multiple entry sections.`

L6202E: `Section <secname> cannot be assigned to a non-root region.`

Certain sections must be placed in root region in the image. __main.o and the two linker-generated tables (Region$$Table and ZISection$$Table) must be in a root region. If not, the linker will report:

```
L6202E: Section Region$$Table cannot be assigned to a non-root region.
L6202E: Section ZISection$$Table cannot be assigned to a non-root region.
```

To fix this, use a root region in the scatter-file containing, as a minimum, these three lines:

```
LOAD_ROM 0x0000 0x4000   ; start address and length
{
    EXEC_ROM 0x0000 0x4000 ; root (load = exec address)
    {
        __main.o (+RO)           ; copying code
        * (Region$$Table)        ; RO/RW addresses to copy
        * (ZISection$$Table)     ; ZI addresses to zero
        :
    }
    :
}
```

L6203E: Entry point (<address>) lies within non-root region <regionname>.
L6204E: Entry point (<address>) does not point to an instruction.
L6205E: Entry point (<address>) must be word aligned for ARM instructions.
L6206E: Entry point (<address>) lies outside the image.
  The image entry point must correspond to a valid instruction in the root-region of the image.

L6207E: Invalid argument for -keep/-first/-last command

L6208E: Invalid argument for -entry command

L6209E: Invalid offset constant specified for -entry (<arg>)

L6210E: Image cannot have multiple entry points. (<address1>,<address2>)
  An ELF image can have only one unique entry point. Specify the unique entry point with -entry.

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 needs to know which AREA you would like to keep.
  To solve this, specify the names of the AREAs that you wish to keep, using more than one -keep option, for example: -keep boot.o(vectors) -keep boot.o(resethandler)...
  Note that using assembler files with more than one AREA may give other problems elsewhere, so this is best avoided.

L6212E: <symbolname> multiply defined (by <object1> and <object2>) at different offsets in a COMMON section.
  See **L6200E**.

L6213E: Multiple First section <object2>(<section2>) not allowed. <object1>(<section1>) already exists.
  Only one FIRST section is allowed.

L6214E: Multiple Last section <object2>(<section2>) not allowed. <object1>(<section1>) already exists.
  Only one LAST section is allowed.

L6215E: Ambiguous symbol selection for -First/-Last. Symbol <symbol> has more than one definition.

L6216E: Cannot use base/limit symbols for non-contiguous section <secname>

Certain sections must be placed contiguously within the same region, for their base/limit symbols to be accessible.

For example, C$$pi_ctorvec, C$$pi_dtorvec and C$$ddtorvec are AREAs generated by the C++ compiler which must be placed in the same region with __cpp_initialise and __cpp_finalise. This can be done by specifying the sections in a scatter-loading description file:

```
LOAD_ROM 0x0000000
{
    EXEC_ROM 0x0000000
    {
        cpp_initialise.o(+RO)
        cpp_finalise.o(+RO)
      * (C$$pi_ctorvec)
      * (C$$pi_dtorvec)
      * (C$$ddtorvec)
      ....
    }

    RAM 0x0100000
    {
      * (+RW,+ZI)
    }
}
```

L6217E: Section <objname>(<secname>) contains R_ARM_SBREL32 relocation (#NN) wrt imported symbol <sym>

L6218E: Undefined symbol <symbol> (referred from <objname>).
<objname> refers to <symbol>, but <symbol> is not defined anywhere. You must either provide a definition of <symbol> or remove the reference to <symbol>.
There are two common examples where this occurs:

1) Undefined symbol Image$$ZI$$Limit (referred from sys_stackheap.o).
It is most likely that you have not re-implemented __user_initial_stackheap(). The RVCT 2.0 \emb_sw_dev directory contains examples of how to re-implement __user_initial_stackheap() - see the file retarget.c. Please see ADS FAQ "Re-implement __user_initial_stackheap() when using Scatterloading" at:
http://www.arm.com/support/ads_faq
Please see also chapter 5 in the RVCT 2.0 Developer Guide - 5.9.4 __user_initial_stackheap()

2) Undefined symbol __rt_embeddedalloc_init (referred from entry.o)
The function __rt_embeddedalloc_init() was used in SDT embedded projects to set up a heap. This is no longer needed in RVCT projects, so the call to it must be removed. You should also remove your implementation of __rt_heapdescriptor() (if there is one).

L6219E: <type> section <object1>(<section1>) attributes {<attributes>} incompatible with neighbouring section <object2>(<section2>).

L6220E: Load/Execution region <regionname> size (<size> bytes) exceeds limit (<limit> bytes).
Example:
L6220E: 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".

L6221E: <type1> region <regionname1> overlaps with <type2> region <regionname2>.

L6222E: Partial object cannot have multiple ENTRY sections
Where objects are being linked together into a partially-linked object, only one of the sections may have an entry point. You will need to rebuild your objects to achieve this. Note: It is not possible here 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 will occur 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.

L6225E: Number <str...> is too long.

L6226E: Missing base address for region <regname>.

L6227E: Using -reloc with -rw-base without -split is not allowed.

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: <str> must follow a single selector.

```
e.g. in a scatter file:
:
* (+FIRST, +RO)
:
```

+FIRST means "place this (single) section first", therefore selectors which can match multiple sections (e.g. +RO, +ENTRY, etc) are not allowed to be used with +FIRST (or +LAST), hence the error message.

L6235E: More than one section matches selector - cannot all be FIRST/LAST.

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:
a) ensure `vectors.o` is specified on the linker command-line.
b) link with "`-keep vectors.o`" to force the linker not to remove this, or switch off this optimization entirely, with `-noremove` [not recommended]
c) [Recommended] Add the `ENTRY` directive to `vectors.s`, to tell the linker that it is a possible entry point of your application, e.g.:

```
        AREA Vect, CODE
        ENTRY    ; define this as an entry point
Vector_table
        ...
```

and then link with "`-entry 0x0`" to define the real start of your code.

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" demands that code maintains 8-byte stack alignment at its interfaces. This allows efficient use of LDRD and STRD instructions (in ARM Architecture 5TE and later) to access 8-byte-aligned "double" and "long long" data types.

Symbols like '~PRES8' and 'REQ8' are "Build Attributes" of the objects. PRES8 means the object PREServes 8-byte alignment of the stack. ~PRES8 means the object does NOT preserve 8-byte alignment of the stack (~ meaning NOT). REQ8 means the object REQuires 8-byte alignment of the stack.

This link error typically occurs in two cases:

1) where assembler code (that does not preserve 8-byte stack alignment) calls compiled C/C++ code (that requires 8-byte stack alignment).

2) when attempting to link legacy SDT/ADS objects with RVCT 2.x objects. Legacy SDT/ADS objects that do not have these attributes are treated as '~PRES8', even if they do actually happen to preserve 8-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 8-byte stack alignment, but which is trying to call function foobar that requires 8-byte stack alignment.

A similar warning that may 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:

1) Rebuild all your objects/libraries using RVCT 2.x.
If you have any assembler files, you will need to:

i) check that all instructions preserve 8-byte stack alignment, and if necessary, correct them.
e.g. change:
```
    STMFD sp!, {r0-r3, lr}   ; push an odd number of registers
```
to
```
    STMFD sp!, {r0-r3, r12, lr}   ; push an even number of registers
```
and:

ii) add the PRESERVE8 directive to the top of each assembler file.
e.g. change:
```
    AREA Init, CODE, READONLY
```
to:
```
    PRESERVE8
    AREA Init, CODE, READONLY
```

(the PRES8 attribute applies to the whole object, not just the code section).

2) 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 (e.g. for qualification/certification reasons), then you must inspect the legacy objects to check whether they preserve 8-byte alignment or not. Use "fromelf -c" to disassemble the object code. C/C++ code compiled with ADS 1.1 or later will normally preserve 8-byte alignment, but assembled code will not.

If your objects do indeed preserve 8-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 saying "I guarantee that these objects are PRES8". The linker warning L6306W is suppressible with "--diag_suppress 6306".

More information about linking with legacy objects/libraries and the "--apcs /adsabi" is given at:
http://www.arm.com/support/rvct2_faq

L6239E: Cannot call ARM/THUMB symbol '<sym>' in non-interworking object <obj2> from ARM/THUMB code in <obj1>(<sec1>)
> Example:
> L6239E: Cannot call ARM symbol 'ArmFunc' in non-interworking object foo.o from THUMB code in bar.o(.text)

> This problem may be caused by foo.c not being compiled with the option "-apcs /interwork", to enable ARM code to call Thumb code (and vice-versa) via 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", '~IW' means "non-interworking"

L6242E: Cannot link object <objname> as its attributes are incompatible with the image attributes.

There are two common reasons for this error message:

1. `Error: L6242E: Cannot link object foo.o as its attributes are incompatible`
    `with the image attributes.`
    `... require 4-byte alignment of 8-byte datatypes clashes with require`
    `8-byte alignment of 8-byte datatypes.`

    This can occur when linking RVCT 2.0 objects with legacy objects built with SDT, ADS or RVCT 1.2. The Application Binary Interface (ABI) has changed between ADS and RVCT 2.0.

    In SDT, ADS and RVCT 1.2, "double" and "long long" data types were 4-byte aligned (unless -Oldrd or __align were used). In RVCT 2.0, "double" and "long long" data types are now 8-byte aligned, according to the new EABI. This allows efficient use of LDRD and STRD instructions in ARM Architecture 5TE and later.

    These changes mean that legacy SDT/ADS/RVCT1.2 objects/libraries using "double" or "long long" data types are not directly compatible with RVCT 2.0 objects/libraries, and so the linker will report an attribute clash.

    Some compatibility is made possible, with some restrictions, by way of the new "--apcs /adsabi" switch in RVCT 2.0. To allow RVCT 2.0 C objects to be used with legacy SDT/ADS C objects, compile the RVCT 2.0 C code with "--apcs /adsabi". If you have C++ code, more detail is given about "--apcs /adsabi" at:
        http://www.arm.com/support/rvct2_faq

2. `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 linking RVCT 2.0 or ADS objects with legacy objects built with SDT. The byte order of 'double' and 'long long' types changed between SDT and ADS.

    In SDT, the formats of little-endian 'double' and big-endian 'long long' are nonstandard. The ADS/RVCT compilers and assembler support industry-standard 'double' and 'long long' types in both little-endian and big-endian formats.

    If you try to link an ADS/RVCT object with an SDT object, all built with the normal defaults, the linker will report an attribute clash.

    Again, the recommended solution is to rebuild your entire project with RVCT. If you do not have the source code for an object or library, then try recompiling your RVCT code with '-fpu softfpa'.

3. `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 attempting to link objects built with different -fpu options. The recommended solution is to rebuild your entire project with RVCT, with the same -fpu options.

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: `Load/Execution region <regionname> address (<addr>) not aligned on a <align>`
`byte boundary.`

L6245E: Failed to create requested ZI section '<name>'.

L6246E: Invalid memory access attributes '<attr>' specified for Execution region <region>

L6247E: Memory attributes of <objname>(<secname>) incompatible with those of parent Execution region <regname>.

L6248E: <objname>(<secname>) in <attr1> region '<r1>' cannot have <rtype> relocation to <symname> in <attr2> region '<r2>'.

> Example:
> L6248E: foo.o(areaname) in ABSOLUTE region 'ER_RO' cannot have address/offset type relocation to symbol in PI region 'ER_ZI'.
> See Compiler #**1359**. See also ADS FAQ "What does "Error: L6248E: cannot have address type relocation" mean?" at: http://www.arm.com/support/ads_faq

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

L6253E: Invalid SYMDEF address: <number>.
L6254E: Invalid SYMDEF type : <type>.

> The content of the symdefs file is invalid.

L6255E: Could not delete file <filename>.

> An I/O error occurred while trying to delete the specified file. The file was either read-only, or was not found.

L6256E: Could not rename file <oldname> to <newname>

> An I/O error occurred while trying to rename the specified file. File specified by newname may already exist.

L6257E: Section <secname> cannot be assigned to an overlaid Execution region.

L6258E: Entry point (<address>) lies in an overlaid Execution region.

L6259E: Reserved Word '<name>' cannot be used as a Load/Execution region name.

L6260E: Multiple load regions with the same name (<regionname>) are not allowed.
L6261E: Multiple execution regions with the same name (<regionname>) are not allowed.

L6262E: Cannot relocate wrt symbol <symbol> (defined at non-zero offset in COMMON section <objname>(<secname>)).

> Relocations to a COMMON section are permitted only through a section relative symbol with zero offset. This error may indicate a compiler fault – please contact your supplier.

L6263E: Cannot express Region Table entry for <regionname> as either address or offset.
L6264E: Cannot express ZISection Table entry for <regionname> as either address or offset.

L6265E: Non-RWPI Section <obj>(<sec>) cannot be assigned to PI Exec region <er>.
L6266E: RWPI Section <obj>(<sec>) cannot be assigned to non-PI Exec region <er>.

This may be caused by explicitly specifying the (wrong) ARM library on the linker command-line. You should not normally need to specify any ARM libraries explicitly e.g. (`c_t__ue.b`) on the link-line.

L6268E: Non-word aligned address <addr> specified for region <regname>.

L6269E: Missing expected <ch>.

L6271E: Two or more mutually exclusive attributes specified for Load region <regname>
L6272E: Two or more mutually exclusive attributes specified for Execution region <regname>

L6273E: Section <object2>(<section2>) has mutually exclusive attributes (READONLY and ZI)

L6274E: Ignoring unknown <attr> attribute <subattr> specified for region <regname>.

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 which was defined by the rejected section now becomes undefined again. The linker will generate an error if the selected copy does not define a symbol that a rejected copy does. This error would normally be caused by a compiler fault – please contact your supplier.

L6276E: Address <addr> marked both as <s1>(from <obj1>) and <s2>(from <obj2>).
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 may indicate a compiler fault – please 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.
L6281E: Cannot rename both <sym1> and <sym2> to <newname>.
L6282E: Cannot rename <sym1> to <newname> as a global symbol of that name exists (defined) in <obj>).
The RENAME command in the steering file is invalid.

L6283E: Object <objname> contains illegal local reference to symbol <symbolname>.
An object cannot contain a reference to a local symbol, since local symbols are always defined within the object itself.

L6284E: Cannot have multiple definitions of macro <macro_name>
Each macro can be defined only once. Multiple definitions of a macro (even using same value) are not permitted.

L6285E: Non-relocatable Load region <lr_name> contains R-Type dynamic relocations.

L6286E: Value(<val>) out of range(<range>) for relocation #<rel_number> (<rtype>, wrt symbol <symname>) in <objname>(<secname>)
means that you have exceeded the limited number of bits for a field within the instruction opcode. For instance, it may be because of an LDR or STR where the offset is too large for the instruction

(+/-4095 for ARM state LDR/STR instruction). It can also occur for a branch instruction where the linker has been unable to insert a "long-branch veneer" at an appropriate place, e.g. because an execution region is too large (Thumb execution regions are currently limited to <4MB). The RVCT 2.0.1 Linker is now able to support large Thumb execution regions.

The RVCT 2.0.1 patches can be downloaded from: http://www.arm.com/support/downloads

L6287E: `Illegal alignment constraint (<align>) specified for <objname>(<secname>).`
An illegal alignment was specified for an ELF object (see **L6334W**).

L6288E: `cannot load constraints file for feature set <filename>`
L6290E: `<option> selection is incompatible with restrictions in '<constraintfile>'`
The feature-restricted toolkit that uses a 'constraints file' is not installed correctly. Try re-installing.

L6291E: `Base address <addr> lies in the previous exec region or before the start of the load region`

L6292E: `Ignoring unknown attribute '<attr>' specified for region <regname>.`

L6293E: `FIXED is incompatible with relative base <offset> for region <regname>.`

L6294E: `Load/Execution region <regionname> spans beyond 32 bit address space (base <base>, size <size> bytes).`
The region has overflowed the 2^32 address limit - make the region smaller.

L6295E: `SB Relative relocation requires image to be RWPI`

L6296E: `Definition of special symbol <sym1> is illegal as symbol <sym2> is absolute.`

L6297E: `Definition of special symbol <sym1> is illegal as symbol <sym2> has synonyms (defined by <obj1>, <obj2>).`

L6298E: `Invalid definition of macro <macro_name>`
A macro definition is invalid if the macro name or value is missing.

L6299E: `Undefined macro <macro_name>`
A macro needs to be defined before it can be used. No definition of the specified macro was found.

L6300W: `Common section <object1>(<section1>) is larger than its definition <object2>(<section2>).`

L6301W: `Could not find file <filename>.`
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.

L6303W: `Symbol <symbol> multiply defined (by <object1> and <object2>).`
See **L6200E.**

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 specify the single, unique entry point with the linker option `-entry`, e.g. `-entry 0x0` is typical for an embedded system.

L6306W: `'<attr1>' section <objname>(<secname>) should not use the address of '<attr2>' function <sym>.`
     See **L6238E.**

L6307W: `<objname>(<secname>) contains 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 link-line, but the library does not contain any members.

L6310W: `Unable to find ARM libraries.`
     This is most often caused by a missing or invalid value of the environment variable RVCT20LIB or by
     incorrect arguments to -libpath.

L6311W: `Undefined symbol <symbol> (referred from <objname>).`
     See **L6218E.**

L6312W: `Empty Load/Execution region description for region <region>`

L6313W: `Using <oldname> as an section selector is obsolete. Please use <newname> instead.`

L6314W: `No section matches pattern <module>(<section>).`
     Example:
     `No section matches pattern foo.*o(ZI).`

     This can occur for two possible reasons:
     1) 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.
     2) 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.

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 <secno>`
     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
```
gives:

`init.o(foo) contains branch to a non-code symbol bar.`

If the destination has no name, e.g:

```
    BL 0x200 ; Branch with link to 0x200 bytes ahead of PC
```
you will see, e.g:

`bootsys.o(BOOTSYS_IVT) contains branch to a non-code symbol <Anonymous Symbol>.`

L6319W: Ignoring <cmd> command. Cannot find section <objname>(<secname>).
L6320W: Ignoring <cmd> command. Cannot find argument '<argname>'.
L6321W: Ignoring <cmd>. Cannot be used without <prereq_cmd>.

L6322W: <n_cycles> cyclic references found while sorting <sec> sections.

L6323W: Multiple variants of <sym> exist. Using the <type> variant to resolve relocation #NN in <objname>(<secname>)

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, which uses the +offset form of base address, inherits the PI/RELOC/OVERLAY attributes from the previous region in the description, or the parent load region if it is the first execution region in the load region.

L6326W: Ignoring ZEROPAD attribute for non-root execution region <ername>.

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 RVCT 2.0 Linker and Utilities guide, section 3.3.3, "Unused section elimination"

L6330W: Undefined symbol <symbol> (referred from <objname>). Unused section has been removed.

L6331W: No eligible global symbol matches pattern <pat>.

L6332W: Undefined symbol <sym1> (referred from <obj1>). Resolved to symbol <sym2>.

L6333W: Undefined symbol <symbol> (referred from <objname>). To be resolved during dynamic linking.
> This warning is produced when a symbol is undefined but the user has marked the symbol to be placed in the Dynamic symbol table. The message is only informational in content and may be ignored. This warning is suppressed by default.

L6334W: Illegal alignment constraint (<align>) for <objname>(<secname>) ignored. Using 4 byte alignment.
> An illegal alignment was specified for an AOF object (see **L6287E**).

L6335W: ARM interworking code in <objname>(<secname>) may contain invalid tailcalls to ARM non-interworking code.

The documentation on interworking can be found in Chapter 3 of the Developer Guide. This makes good background reading, but unless you are coding assembler routines that will be interworked, all you really need to do is use `-apcs /interwork`.

L6337W: `Common code sections <o1>(<s1>) and <o2>(<s2>) have incompatible floating-point linkage`

L6338W: `Load/Execution region <regionname> at <offset> aligned to next <align> byte boundary.`

L6339W: `Ignoring RELOC attribute for execution region <regname>.`
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 addresssing.

L6340W: `options first and last are ignored for link type of -partial`
The `-first` and `-last` options are meaningless when creating a partially-linked object

L6341E: `Address of <objname>(<secname>) (<base0xX>) does not match the required address <reqd_base0xX>`

L6559I: `Searching for ARM libraries in directory <dirname>`

L6565I: `Not eliminating unused sections as image is unsuitable for such optimization.`
Unused section elimination cannot be performed on this image.

L6567I: `Not enough information to produce a SYMDEFs file.`
The `-symdefs` option could not create a symdefs file because, e.g, linking failed to complete.

L6568I: `Not enough information to list image symbols.`
The `-symbols` option could not complete because, e.g, linking failed to complete.

L6569I: `Not enough information to list the image map.`
The `-map` option could not complete because, e.g, linking failed to complete.

L6570I: `Not enough information to list the image sizes and/or totals.`
The `-info sizes` or `totals` option could not complete because, e.g, linking failed to complete.

L6602W: `Unmatched literal pool end symbol <symname> ignored in file <filename>.`
L6603W: `Literal pool begin symbol <symname> found within Literal pool in file <filename>.`
L6604E: `Literal pool end symbol <symname1> is in different area from literal pool begin symbol <symname2> in file <filename>`
These three relate to AOF to ELF object conversion. The AOF object may be invalid. Try recompiling the source file with SDT to create a new AOF object, or preferably, recompile the source file with RVCT to create a new ELF object.

L6627U: `Bad error message list <list> for command <command>`

L6629E: `Unmatched parentheses expecting ) but found <character> at position <col> on line <line>`

L6630E: `Invalid token start expected number or ( but found <character> at position <col> on line <line>`

L6631E: `Division by zero on line <line>`

L6632W: `Subtraction underflow on line <line>`

L6633E: Could not open intermediate scatter file to send to pre-processor

L6634E: Pre-processor command too long, maximum length of <max_size>

L6635E: Could not open intermediate scatter file produced by pre-processor

L6636E: Scatter file pre-processor step failed

L6637W: Could not find any explicitly loaded input objects, need at least one input object or library(object).

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>

L6641E: Cannot find base of consolidated output section for input sections <secname> as sections are not contiguous

L6649E: EMPTY region <regname> must have a maximum size.

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.
L6654E: Local symbol <symname> from <secname> in <objname> is referred to from a non group section <nongrpname>

If the one of the three Errors above is reported this may indicate a compiler fault; please contact your supplier.

L6652E: Cannot reverse Byte Order of Data Sections, input objects are <inputendian> requested data byte order is <dataendian>.

L6653W: Dynamic relocations are not partitioned into code and data targets.
Warning for RVCT2.0, full v6 byte invariant addressing is not available in this version

L6664W: Relocation #<rel_number> in <objnams>(<secname>) is wrt a symbol(#<idx> before last Map Symbol #<last>).

L6670W: -nodebug overrides -bestdebug, all debug sections will be removed.

5. ELF Format Converter (fromelf) Errors and Warnings

Q0105E: Base and/or size too big for this format, max = 0x<maxval>.

Q0106E: Out of Memory.

Q0107E: Failed writing output file.

Q0108E: Could not create output file '<filename>'.

Q0111E: Unrecognised option '<opt>'.

Q0112E: Missing output format before '<arg>'.

Q0113W: Ignoring unrecognised text information category '<cat>'.

Q0114W: Ignoring multiple input file '<filename>'.

Q0115W: Deprecated command syntax will not be supported in future versions. Use -output to specify the output file.
> This warning is intended to highlight that the old SDT 2.5x form of the `fromelf` command:
>> `fromelf -bin image.elf image.bin`
> has now been changed to:
>> `fromelf image.elf -bin -o image.bin`

Q0116E: No text information category specified.

Q0117E: Unrecognised file format '<arg>'.

Q0118E: Missing argument for option '<arg>'.

Q0119E: No output file specified.
Q0120E: No input file specified.
Q0122E: Could not open file '<filename>'.
Q0123E: Failed to read file. Invalid seek offset possible.

Q0127E: Cannot translate an ELF Relocatable file (object) into <format> format.
> Only executable files can be translated in this way.

Q0128E: File i/o failure.

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. In RVCT, 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**.

Q0139E: ELF Image has insufficient information to effect this translation.

Some fromelf operations require the ELF image to contain debug information. Rebuild your image with '-g'.

Q0140E: `ELF image requires an entry point to effect this translation.`
Some fromelf operations require the ELF image to have an entry point. Rebuild your image with '-entry'. This error can also occur with 3rd-party tools that do not set an ARM-specific flag (e_flags) in the ELF header. This flag is used by ARM tools to distinguish between an ELF image with no entry point, and an ELF image with an entry address of 0.

Q0141E: `Invalid debug offset found. Seek failure.`

Q0142E: `ELF Image does not have a ROOT Region.`
The image entry point must correspond to a valid instruction in the root-region in the image. Images that have been successfully created with the ARM linker will always have this.

Q0143E: `Failed to write High level debug information.`
Q0144E: `Failed to write Low level debug information.`
Q0145E: `Failed to write image string table.`
A file could not be written to - check that you have write access permissions.

Q0147E: `Failed to create Directory <dir>.`
Q0148E: `Failed to change to directory <dir>.`
Q0149E: `Failed to change to directory <dir>.`
The directory could not be accessed - check your access permissions.

Q0158W: `Cannot use filename as argument '<file>'.`

Q0159W: `Multiple output formats specified. Ignoring <format>.`

Q0160E: `Invalid ELF section offset found '<offset>'.`
See **Q0131E**.

Q0161E: `Section contents do not lie fully within the file '<offset>'.`

Q0162E: `Invalid ELF program segment offset found '<offset>'.`
See **Q0131E**.

Q0163E: `Program segment contents do not lie fully within the file. '<idx>'.`

Q0164E: `Invalid e_shstrndx value (<shstrndx>) found in ELF header (total sections <e_shnum>).`

Q0165E: `Symbol Table Section has not got type of SHT_SYMTAB.`
The ELF section '.symtab', which contains the symbol table, must have type SHT_SYMTAB. If a given ELF file does not have this, this may be due to the ELF file being corrupt. Try re-linking it.

Q0166E: `Relocation Section has not got type of SHT_REL nor SHT_RELA.`

Q0167E: `Error occurred in section <idx>.`
Q0168E: `Error occurred in section <idx>.`

Q0170E: `Section pointer is null`

Q0171E: `Invalid st_name index into string table <idx>.`
See **Q0131E**.

Q0172E: `Invalid index into symbol table <idx>.`

See `Q0131E`.

Q0173E: `Failed to close temporary file`
Q0174E: `Failed to delete temporary file`
Q0175E: `Failed to rename temporary file`

Q0178U: `Internal error: bad section header pointer in section with index <idx>.`

Q0179W: `Multiple bank types specified. Ignoring <banks>.`

Q0180W: `Symbol Table entry size is 0.`

Q0181W: `Relocation entry size is 0.`

Q0182E: `Failed to open temporary file`

Q0183W: `<format> format is obsolete and will not be supported in future versions of the toolkit.`

Q0184E: `Section <name> (<number>) has File Offset <offset> which is not <required_align> byte aligned`

Q0185E: `Unable to make unique temporary file from <file>`

Q0186E: `This option requires dwarf2 debugging information to be present`

Q0187E: `Cannot produce addresses for Relocatable Elf file`
"`fromelf -a`", which prints data addresses, can only be used on executable image files, not object files.

Q0188E: `Program segment <number> must be <required_align> aligned in file`

Q0189U: `Internal error: bad segment header pointer in section with index <idx>.`

Q0190E: `String Table Section <idx> has not got type of SHT_STRTAB.`
The ELF section '`.strtab`', which contains the string table, must have type `SHT_STRTAB`. If a given ELF file does not have this, this may be due to the ELF file being corrupt. Try re-linking it.

Q0191E: `Option <old> has changed name and is now deprecated, please use <new> instead.`

Q0192E: `Warning IHF format is deprecated and will be removed in a future release`
See also `Q0105E`

Q0193E: `Could not save output file <filename>, removal of temporary file failed`
Is given if a failure is detected when deleting a file

Q0194E: `Could not save output file <filename>, renaming of temporary file failed`
Is given if a failure is detected when renaming a file

Q0195E: `Cannot open <filename>, existing directory with same name`

Q0419E: `No SYMTAB_SHNDX section exists for section <sec_idx>`

Q0420E: `Out of range symbol idx <sym_idx>`

Q0421E: `No associated SHT_SYMTAB_SHNDX section for SHT_SYMTAB section <symtab_sec>`

Q0422E: Bad error message list <list> for command <command>

Q0424E: More than one relocation section for <secname>

6. ARM Librarian (armar) Errors and Warnings

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
L6833E: File '<filename>' does not exist
L6834E: Cannot open file '<filename>' : <reason>
L6835E: Reading file '<filename>' : <reason>

L6836E: '<filename>' already exists, so will not be extracted

L6837E: Unrecognized option '<option>'

L6838E: No archive specified

L6839E: One of [<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>

L6844E: Missing argument to '<option>'

L6845E: Cannot delete '<member>' as '<member>' is a variant of it
L6846E: Cannot insert variant '<member>' as there is no symbol-compatible base member
L6847E: Cannot insert '<filename>' as it has incompatible build attributes
L6848E: Cannot replace '<member>' as new version and old version are not symbol compatible, and it has a variant member ('<variant_member>') dependant upon it

L6849E: Unrecognized long option '<option>'

L6850E: Archive contains non ELF Object <name>

L6851E: Bad error message list <list> for command <command>

L6870W: Via file '<filename>' is empty

L6871W: Build attributes of archive members inconsistent with archive name

L6874W: Minor variants of archive member '<member>' include no base variant

It is possible to have minor variants of the same function within a library, by compiling each variant with different build options in separate (individually named) object files. If these objects are combined in a library, at link-time the linker will select the most appropriate version of the function according to the callers build attributes. Examples of minor variants are versions compiled for different architectures,

ROPI/non-ROPI etc. Major variants must be placed in separate libraries, examples are versions compiled for different instruction sets (ARM/Thumb), endianness etc.

A base variant is a library member that contains all the attributes in common to all the variants. armar is warning as it is usually a mistake to define a set of variants without a base variant, as the linker may not be able to find a default acceptable member in the library.
For the case of:

```
    Warning: L6874W: Minor variants of archive member 'abc.o' include no
base variant
```

'abc.o' (probably unintentionally) contains a function which is also defined in another archived object, which was built with different options. You can view the symbol table of an archive using 'armar -zs' - variant symbols will be appended with their build attributes.
For example, if an archive contained an architecture v3 function 'func' and an architecture v4 variant, the symbols table might show:

```
func                             from v3_func.o  at offset   120
func$$BuildAttributes$$ARM_ISAv4  from v4_func.o  at offset  1104
```

Assuming that you intended to have different variants of the function, you would need to add an object containing a base variant in order to fix the warning. Alternatively, you could safely ignore the warning, but at link-time there is a risk that the linker may not be able to find a suitable default member.

L6875W: Archive <name> is not an ELF Object Library