WNDRVR
SUPPORT NETWORK

WIND RIVER DIAB COMPILER ERROR MESSAGES REFERENCE, 5.9.8.1

Corporate Headquarters

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.
Toll free (U.S.A.): +1-800-545-WIND
Telephone: +1-510-748-4100
Facsimile: +1-510-749-2010

For additional contact information, see the Wind River website:

www.windriver.com

For information on how to contact Customer Support, see:

www.windriver.com/support

Wind River Diab Compiler Error Messages Reference, 5.9.8.1

20 March 2023

1. ABOUT ERROR MESSAGES

1. Introduction

In analyzing messages, remember that a message can be generated for code that is apparently correct. Such a message may be the result of earlier errors. If a message persists after all other errors have been cleared, please report the circumstances to Customer Support.

2. Compiler Message Format

Information on compiler message format.

Compiler messages have the form:

```
"file", line # : severity-level (compiler :error # ) : message
```

Messages have one of four severity level values as follows. Note that the message descriptions in the following sections contain a severity code, rather than the severity level. Actual messages generated by the tools display the severity level.

Level	Code	Compilation Continues	Object File Produced	Notes
Information	**i**	Yes	Yes	Usually provides detailed information for an earlier message.
Warning	**w**	Yes	Yes	
Error	**e**	Yes	No	
Fatal	**f**	No	No	

The severity level of a message can be changed with the -e command-line option. See the user's guide for more information about -e and command-line options in general.

In each message, *"compiler"* identifies the compiler reporting the error: **dcc**, **ctoa**, or **etoa**.

Example:

```
"err1.c", line 2: error (dcc:1525): identifier i not declared
```

2. ERRORS IN ASM MACROS AND ASM STRINGS

1. Errors in asm Macros and asm Strings

Errors in assembly code embedded in C or C++ using **asm** macros or **asm** string statements are caught by the assembler, not by the compiler.

If the -S option is not used, the compiler will generate a temporary assembly file that is discarded after assembly. To preserve the assembly file for use in diagnosing errors reported in **asm** macros or **asm** strings, do one of the following:

- Use the -Xkeep-assembly-file and -Xpass-source command-line options to generate an annotated assembly file along with the object file.
- Use the -S option to stop after compilation, along with the -Xpass-source option, and then assemble the file explicitly using **das**.

Errors and warnings issued by llopt

Consider the following example:

```
$ cat test.c
void Test(void)
{
    asm(" addis r11,r0,(__SP_INIT-0x10)@ha ");
    asm(" addi r1,r11,(__SP_INIT-0x10)@l ");
}
$
```

The compilation step generates llopt warnings:

```
$ dcc -tPPCE200Z7VEN:simple -S test.c
/folk//diabuild/wrc_iter10/5.9.8.0/LINUX386/bin/llopt - "/tmp/dtmpCAAa0028605":50, Error, Stack or
 frame pointer used, but offset required for register allocation has not been adjusted. Instructio
n: addi r1,r11,(__SP_INIT-16)@l
/folk//diabuild/wrc_iter10/5.9.8.0/LINUX386/bin/llopt - "/tmp/dtmpCAAa0028605":50, Info, Additiona
l stack space is required when: 1) registers are spilled, and 2) when preserved registers are chos
en and must be saved on the stack.
$
```

The file should be compiled without coloring by using the -Xcoloring=0 option.

Using the -Xcoloring=0 option impacts optimization. As little code as possible should therefore be compiled without coloring.

3. COMPILER ERROR MESSAGES

1. Compiler Error Messages

There are two divisions of compiler error messages, **ctoa** and **etoa**.

Compiler error messages are divided up as follows:

- Those generated by **ctoa** (the default C frontend)—described in Compiler Message Format on page 1
- Those generated by **etoa** (the newer, EDG-based compiler for C, C99, and C++)—described in Compiler Message Format on page 1

Note: **etoa** is also the default frontend for VxWorks RTP projects.

When a message is shared by compilers, the same error message number is used for all instances.

2. Messages Generated by ctoa

The messages in this section are generated by **ctoa**, the default C frontend.

Messages generated by **etoa** (invoked by default for C++ and for C with **-Xc-new**) are listed beginning in Compiler Message Format on page 1.

Numbered messages are issued by the compiler subprogram. Unnumbered messages are issued by the driver and are listed first.

(driver) **can't find program** *program_name*

program_name will be the name of some component of the compiler or other tool. (f)

Possible causes:

- The compiler is not installed properly.
- One of the compiler files has been deleted, hidden, or protected.
- The **dtools.conf** or other configuration file is incorrect.

(driver) **can't fork**

The system cannot start a new process. (f)

(driver) **missing comma in -Y option**

The **-Y***c*,*dir* option must include a comma. (f)

(driver) **illegal output name** *file*

Specific output filenames given with the **-o** option are invalid to avoid common typing mistakes. (f)

```
dplus a.c -o b.c # b.c is an illegal output file name
```

(driver) **invalid option** *unknown*

The driver was started with an unrecognizable -W or -Y option. Note: -X options that are not recognized generate an "unknown option" message, and unrecognized but otherwise valid non -X options are passed to the linker. (f)

(driver) program *tool-name* terminated

The given executable has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1000: (general compiler error)

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1001: **illegal argument type**

The operand cannot be used with the operator. (e)

```
if ( i > pointer ) . . .
```

1003: function takes no arguments

Function was defined without arguments, but was called with arguments. (e)

```
int fun (){} main(){ fun(1); }
```

1004: wrong number of arguments

Number of arguments given does not match prototype or function definition, (w) in C modules if **-Xpcc** or **-Xk-and-r** or **-Xmismatch-warning**, (e) otherwise.

```
int fn(int, int); ... fn(1,2,3);
```

1006: *string* in *string*

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1007: ambiguous conversion -- cannot cast operand

The compiler cannot find an unambiguous way to convert an item from one type to another. (e)

1010: *Operator*, *type-designator*, **argument must be of pointer or integral type**

An operator that requires an integral or pointer type was applied to a different type.

```
float f; f = ~f;
```

1012: *operator*, *type-designator*, **argument must be of pointer or arithmetic type**

The operator requires a pointer or arithmetic type operand. (e)

```
struct S { int i; }s; struct S *p; *p ->
                            i =3;     //
```

1013: left argument must be of integral type

The left operand must be an integral type. (e)

```
pointer | 3;
```

1015: *type-designator* , *operator* , *type-designator* , **left argument must be of arithmetic type**

The operand to the left of the operator must be of arithmetic type. (e)

```
pointer * 2; pointer / 2;
```

1017: right argument must be of integral type

The right operand must be an integral type. (e)

```
 7 | pointer;
```

1019: *type-designator* , *operator* , *type-designator* ,**right argument must be of arithmetic type**

The operand on the right of the operator must be of arithmetic type. (e)

```
2 * pointer; 2 / pointer;
```

1025: division by zero

The compiler has detected a source expression that would result in a division by 0 during target execution. (w)

```
 int z = 0; fn(10/z);
```

1028: *type-designator* [*type-designator*] **requires a pointer and an int**

A subscripted expression requires a pointer and an integer. (e)

```
main(){ int x; x[3]=4; }
```

1030: can't take address of main

Special rules for the function main( ) are violated. (e)

```
int *p; p = main;
```

1031: can't take address of a cast expression

The address operator requires an **lvalue** for its operand. (e)

```
int i, *p; float f; p = &(int)f;
```

1032: (anachronism) address of bound member function

The correct way to refer to the address of a member function is to use the "**::**" operator. The C method, using the dot "**.**" operator, causes the compiler to generate the "anachronism" warning. (w)

```
class C { public: fun(); } c; main(){ class C *
                            p; p= &c.fun; // Old way to reference a function }
```

1033: can't take address of expression

Cannot use "**&**" or other means to find the address of the expression. (e)

```
int *pointer; &pointer++;
```

1034: can't take address of bit-field expression

The address of bit-fields is not available. (e)

```
int *p; struct { int i:3; }s; p =
                            &s.i;
```

1041: returning from function with address of local variable

A **return** statement should not return the address of a local variable. That stack area will not be valid after the return. (w)

```
int i; return &i;
```

1042: ?"*type-designator* ":" *type-designator* , **bad argument type**(**s**)

Incompatible types have been used with the conditional operator. (e)

```
int i, *pointer, *p; p = (2>1) ? i :
                            pointer;
```

1043: trying to decrement object of type bool

A a **boolean** cannot be decremented. (e)

```
bool b; b--;
```

1044: assignment to constant expression

A constant cannot be assigned a value after the constant is defined. (e)

```
const int i=5; i=7;
```

1045: assignment to non-lvalue of type *type-designator*

The operand being assigned is not an **lvalue** type. (e)

```
const c = 5; c = 7;
```

1046: assignment from *type-designator* to *type-designator*

An attempt has been made to assign a type to an incompatible type. (e)

```
int i, j; i = &j;
```

1047: trying to assign "**ptr to const**" **to** "**ptr**"

A pointer to a **const** cannot be assigned to an ordinary pointer. (e)

```
 const int *pc; int pi; ... pi = pc;
```

1050: bad left argument to operator *operator* **not a pointer**

The operator requires a pointer for its left operand. (e)

```
int int1, j; int1 -> j=3;
```

1051: not a class/struct/union expression before ...

The left hand side of a "**.**" or "**.***" or "**->**" or "**->***" operator must be of type **class** or pointer to **class**. (e)

```
                            5->a = 128; // 5 is not a pointer to a class
```

1055: illegal function call

The function call is not valid. (e)

```
int i; i();
```

1056: illegal function definition

A function definition is invalid. (e)

```
fun(iint i);
```

1057: main( ) may not be called from within a program

Calling **main( )** is not permitted. (e)

```
fun(){ main(); }
```

1059: (compiler error)

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1060: assignment operator "=" found where "==" expected

Encountered a conditional where the left hand side is assigned a constant value: (w)

```
if (i = 0) ... /* should possibly be i == 0) */
```

1061: illegal cast from *type-designator* to *type-designator*

An attempt is made to perform a cast to an invalid type, i.e., a structure or array type. (e)

```
struct a = (struct abc)x;
```

1063: ambiguous conversion from *type-designator* to *type-designator*

The compiler cannot find an unambiguous way to convert an item from one type to another. (e)

1074: illegal cast

An attempt is made to perform a cast to an invalid type, i.e., a structure or array type. (e)

1075: friend declaration outside class/struct/union declaration

The keyword **friend** is used in a invalid context (e)

```
friend class foo { ... };
```

1076: static only allowed inside { } in a linkage specification

Attempt to declare a static object in a one-line linkage specification. (e)

```
extern "C" static int i; // static + extern at same time?
```

1077: typedefs cannot have external linkage

Linkage specification ignored for **typedef**, cannot have "**C**" or "**C++**" linkage. (w)

```
extern "C" typedef int foo;
```

1079: identifier *name* **previously declared** *linkage*

The identifier was already declared with another linkage specification. (e)

```
int foo; extern "C" int foo;
```

1080: inconsistent storage class specification for *name*

The identifier was already declared, with another storage class. (e)

```
bar() { int foo; // foo is auto by default
                            static int foo; // now static }
```

1081: illegal storage class

External variables cannot be **automatic**. Parameters cannot be **automatic**, **static**, **external**, or **typedef**. (e)

```
int fn(i) static int i; { ... }
```

1082: illegal storage class

A variable has been declared, but cannot legally be assigned to storage. (e)

```
 register int r;            // Outside of any
                             function
```

1083: only functions can be inline

The **inline** keyword was applied to a non-function, for example, a variable. (e)

1084: only non-static member functions can be virtual

For example, operators **new** and **delete** cannot be **virtual**.

```
 virtual void *operator new( size_t
                            size){...}
```

1086: redeclaration of *identifier*

It is invalid to redeclare a variable on the same block level. (e)

```
 int a; double a;
```

1087: redeclaration of function

A function was already declared. May be caused by mis-typing the names of similar functions. (e)

1088: illegal declaration

Common causes and examples: (e)

A scalar variable can only be initialized to a single value of its type.	`int i = 1, 2;`
Functions cannot return arrays or functions.	`char fn()[10];`
Variables cannot be of type **void**. (Usually caused by a missing asterisk, e.g. **void *p;** is correct.)	`void a;`
Only one **void** is allowed as function argument.	`int fn(void, void);`
An array cannot contain functions.	

1089: illegal initializer

An initializer is not of the proper form for the object being initialized. Often caused by a type mismatch or a missing member in a structure constant. (e)

1090: static/external initializers must be constant expressions

Static initializations can only contain constant expressions. (e)

```
static int i = j+3;
```

1091: string too long

A string initializer is larger than the array it is initializing. (e)

```
char str[3] = "abcd";
```

1092: too many initializers

The number of initializers supplied exceeds the number of members in a structure or array. (e)

```
int ar[3] = { 1,2,3,4 };
```

1094: illegal type for identifier *identifier*

This can indicate a **template** was instantiated with the wrong arguments. (e)

```
template<class T> class C{}; C<int,
                            int> WrongArgs;
```

1096: typedef may not have the same name as its class

Only constructors and destructors for a class may have the same name as the class. (e)

1097: function-declaration in wrong context

A function may not be declared inside a **struct** or **union** declaration. (e)

```
struct { int f(); };
```

1098: only non-static member functions can be *string*

Only non-static member functions can be **const** or **volatile**.

```
class A { static foo() const; };
```

1099: all dimensions must be specified for non-static arrays

For an array in a class all dimensions must be specified, even if the array is not **static**. (e)

1100: member is incomplete

The structure member has an incomplete type, i.e., an empty array or undefined structure. (e)

```
struct { int ar[]; };
```

1101: anonymous union member may not have the same name as its class

Only constructors and destructors for a class may have the same name as the class. (e)

1102: anonymous unions can't have member functions

1103: anonymous unions can't have protected or private members

1104: name of anonymous union member *name* **already defined**

An identifier with the same name as an anonymous **union** member was already declared in the scope. (e)

```
int i; static union {
                            int i; // i already declared }
```

1105: anonymous unions in file scope must be static

A special rule for an anonymous **unions** is violated. (e)

1106: friends can't be virtual

A **friend** is not a member of the class; it cannot be **virtual**. (e)

1107: conversion functions must be members of a class

It is not valid to define a conversion function that is not a class member. A conversion function cannot take arguments. A conversion function cannot convert to the type of the class if it is a member of, or a reference to it. (e)

1108: member function declared as friend in its own class

Invalid declaration. (e)

```
class A { foo(int); friend A::foo(int);
                            }
```

1110: identifier *identifier* **is not a member of class** *class-name*

The identifier to the right of **::** is not in the class on the left side. (e)

1111: identifier *identifier* **not member of struct/union**

The expression on the right side of a "**.**" or "**->**" operator is not a member of the left side's **struct** or **union** type. (e)

1112: member declaration without identifier

A **struct** or **union** declaration contains an incomplete member having a type but no identifier. (w)

```
struct foo { int; ...};
                            struct { struct bar { ... }; ... };
```

1113: identifier *name* **used both as member and in access declaration**

A use of the *name* would be ambiguous. (e)

```
class A { public: int foo; };
                            struct B : private A { int foo; A::foo; };
```

1114: array is incompletely specified

An array cannot be declared with an incomplete type. (e)

```
 int a[]; // No array size
```

1115: type ... is incomplete

Attempt to access a member in an incomplete type. (e)

1117: identifier *identifier* **not an argument**

An identifier that is not in the parameter list was encountered in the declaration list of an old-style function. (e)

```
 f(a) int b; { ... }
```

1120: constant expression expected

The expression used in an enumerator list is not a constant. (e)

```
enum a { b = f(), c };
```

1121: integer constant expression expected

The size of an array must be computable at compile time. (e)

```
int ar[fn()];
```

1122: switch expression should be integral not a pointer or long long - cast inserted

The expression in a **switch** statement must be an integer that fits into a **long** (e.g., for 32-bit targets it must fit into 32 bits). On 32-bit targets **long long** values will be truncated; pointer values are allowed by ctoa but will be cast to integers. (w)

See also Error 4031 in Messages Generated by etoa on page 52.

1123: illegal type of switch-expr

A **switch** expression is of a non-integral type. (e)

1124: duplicate default labels

A **switch** has should not have more than one default label.

1125: int constant expected

A bit-field width must be an integer constant. (e)

1126: case expression should be integral constant

Case expressions must be integral constants. (e)

```
int i,j; switch (i) {   case j: i = 8;
                              }
```

1127: duplicate case constants

A **case** constant should not occur more than once in a **switch** statement. (e)

```
case 1: ... case 1:
```

1127: duplicate case constants

Duplicate **case** constants were detected. (e)

```
main(){ int year,j; switch (year) {   case 2000:
                              j = 8;   case 2000: j = 9; } }
```

1128: function must return a value

Found a **return** statement with no value in a function. (e)

```
int foo() { return; // Must return a value.
                              }
```

1129: constructor and destructor may return no value

A constructor or destructor must not return a value. (e)

1130: parameter decl. not compatible with prototype

There is a mismatch between a prototype and the corresponding function declaration in either number of parameters or parameter types. (e)

```
int fn(int, int);
                            int fn(int a, float b) { ... }
```

1131: multiple initializations

A variable was initialized more than once. (e)

```
static int a = 4; static int a = 5;
```

1133: extern objects can only be initialized in file scope

An **extern** object cannot be initialized inside a function. (e)

```
main(){ extern int i=7; }
```

1133: extern objects can only be initialized in file scope

Attempt to initialize an **extern** object in a function. (e)

```
foo() { extern int one = 1; }
```

1134: can't initialize arguments

It is not valid to attempt to initialize function parameters. (e)

```
 f(i) int i = 5; { ... }
```

1135: can't init typedefs

A **typedef** declaration cannot have an initializer. (e)

```
 typedef unsigned int uint = 5;
```

1136: initialization of automatic aggregates is an ANSI extension

When the compiler is run in PCC compatibility mode on a C module (-Xpcc), it will report initialization of automatic aggregate types. (w)

```
 f() { int ar[3] = {1,2,3}; ... }
```

1140: too many parameters for operator ...

Overloaded operator declared with too many parameters. (e)

1141: too few parameters for operator ...

Overloaded operator declared with too few parameters. (e)

1142: second argument to postfix operator "++" or "--" must be of type int

The argument is of the wrong type. (e)

```
struct A {
                            operator++(double); // Arg type must be int };
```

1143: operator->( ) must return class or reference to class

1144: operator ... can only be overloaded for classes

The operators "**,**" and "**=**" and the unary "**&**" can only be overloaded for classes. (e)

1145: operator . . . must be a non-static member function

The operators **( )**, **[]**, and **->** must be non-static member functions. These operators can only be defined for classes. (e)

1146: non-member operator function must take at least one argument of class or enum type or reference to class or enum type

A non-member operator function must take at least one argument, which is of a **class** or **enum** type or a reference to a **class** or **enum** type. (e)

```
Date operator+(int i, j){...}
```

1147: constructors can't be declared *string*

Constructors cannot be declared **static** or **virtual**.

1148: constructors can't have a return type

A constructor declaration is invalid. (e)

1149: constructor is illformed, must have other parameters

A constructor declaration is invalid. (e)

1151: can't have a destructor in a nameless class/struct/union

A nameless class cannot have a destructor since the destructor takes its name from the class. (e)

```
class { ~foo(); };
```

1152: destructors must have same name as the class/struct/union

The destructor declaration is invalid. (e)

1153: destructors may have no return type

For example:

```
const ~k(){}
```

1154: destructors can't be declared *string*

Destructors cannot be declared **static**.

1155: destructors may take no arguments

The destructor declaration is invalid. (e)

1156: conversion functions may take no arguments

It is not valid to define a conversion function that is not a class member. A conversion function cannot take arguments. A conversion function cannot convert to the type of the class if it is a member of, or a reference to it. (e)

1157: conversion to original class or reference to it

It is not valid to define a conversion function that is not a class member. A conversion function cannot take arguments. A conversion function cannot convert to the type of the class if it is a member of, or a reference to it. (e)

1159: no type found for *identifier***, can be omitted for member functions only**

The identifier has not been declared. (e)

1160: class already has operator delete with *number of* **argument(s)**

The **delete** operator cannot be overloaded. (e)

1161: member operator functions can't be static

Operator functions in a class cannot be declared **static**. (e)

1162: member of abstract class

A class member cannot be of abstract type. (e)

1163: unions can't have virtual member functions

Union cannot have **virtual** functions as members. (e)

1164: member function of local class must be defined in class definition

Because functions cannot be defined in other functions, any function in a local class must be defined in the class body. (e)

1165: redeclaration of member *identifier*

A member occurs more than once in **a struct**, **union**, or **class**. (e)

```
struct { int m1; int m1; };
```

1166: member *name* **already declared**

Attempt to re-declare a member. (e)

```
class A { int a; int a; // Already declared
                            };
```

1167: static data member may not have the same name as its class

Only constructors and destructors for a class may have the same name as the class. (e)

1168: a local class can't have static data members

Only non-**static** members can be used in a local class. (e)

1169: unions can't have static data members

Union cannot have **static** data members. (e)

1170: illegal union member

An object of a class with a constructor, a destructor, or a user defined assignment operator cannot be a member of a union. (e)

1171: illegal storage class for class member

A class member cannot be **auto**, **register**, or **extern**. (e)

1172: parameter has no identifier

When declaring a function, a name as well as a type, must be supplied for each parameter. (e)

```
int fn(int a, int) { ... }
```

1173: compiler out of sync: probably missing ";" or "}

For example:

`int i int j;`	missing **';**" after i
`dribble f;`	should be **double**

1174: ellipsis not allowed as argument to overloaded operator

Cannot declare an overloaded operator with "**...**" as arguments. (e)

1175: ellipsis not allowed in pascal functions

Functions declared with the **pascal** keyword are not allowed to have a variable number of arguments as indicated by an ending ellipsis "...". (e)

1176: argument *n* **to string must be of type size_t**

For example, operator **delete's** second argument must be of type size_t

```
void operator delete(void *type, int x){
                            free(type); }
```

1177: *string* **must return void ***

For example the operator **new** must return a **void** pointer.

```
 int *operator new(size_t size){...}
```

1179: *string* **takes one or two arguments**

For example, operator **delete** takes one or two arguments (e).

```
 void operator delete(void *type, size_t size,
                            int x){...}
```

1180: operator delete must have a first argument of type void *

The first argument of **delete** must be of type **void***.

```
void operator delete(int x){ free(x);
                            }
```

1181: *string* **must return void**

For example, operator **delete** must return **void**.

```
 int operator delete(void
                            *type){...}
```

1182: class *class-name* **has no constructor**

It is invalid to initialize an object that does not have a constructor by using the constructor initialization syntax. (e)

```
struct A { int b, c; }; A a(1,2);
```

1183: temporary inserted for non-const reference

The compiler made a temporary copy of a variable used in an assignment to a C++ reference. (w)

```
void getCount(unsigned int& count) { count =
                            5; return; } ... signed int x = 100; getCount(x);
```

In this example, the compiler makes a temporary copy of **x** and passes the copy (cast to **unsigned int**) to **getCount**. Hence it is the copy of **x**, and not **x** itself, that is modified by **getCount**; after the function executes, the value of **x** is still 100, not 5.

1184: temporary inserted for reference return

For Example:

```
vint& constant1() {  return 1;
                            }
```

1186: const member *identifier* **must have initializer**

A constant member of a class must be initialized. (e)

```
class line{ const int length; ...
                            };
```

1188: jump past initializer

An object cannot be accessed before it has been constructed.

```
class C { public: int i; C(int ii) : i(ii) {}
                            }; void AllAlarmsOnOff (int function) { switch ( function ) {  cas
e 1:
                             C c(0);  break;  default:  c.i = 12;  // invalid access  break; }
                            }
```

1190: this cannot be used outside a class member function body

1192: mismatching parenthesis, bracket or ? : around expression

Mostly likely, a parenthesis or bracket was left out of an expression, or the "**?**" and "**:**" in a conditional expression where interchanged. (e)

```
                            int i = (5 + 4]; // ] should have been a )
```

1193: missing operand for operator

An operand is missing. (e)

```
 i & ;
```

1194: (compiler error)

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please notify Customer Support. (f)

1195: missing operand somewhere before

An operand was left out of an expression. (e)

1196: missing expression inside parenthesis

An expression was expected between the parentheses. (e)

```
 i =() ;
```

1197: missing operand for operator ... inside parenthesis

An operand was left out of an expression. (e)

1198: too many operands inside parenthesis

An operator between the operands is missing. (e)

1199: missing expression inside brackets

An expression was expected between the brackets. (e)

```
int x[5];
                              int i = x[]; // x must be subscripted
```

1200: missing operand for operator ... inside brackets

1201: too many operands inside brackets

1202: missing operator before *string*

An operator is needed before *string* .

```
 i = (2>1) 3: 4; // Conditional operator
                              needs '?'
```

1205: operator ? without matching :

Operator "**?**" must be followed by a "**:**" . (e)

```
 int i = 4 ? 5; // Missing : part
```

1207: syntax error near *token*

The parser has found an unexpected token. (e)

```
 if (a == 1 ( /* missing ')' */
```

1208: expression expected

Could not find an expression where it was expected. (e)

```
if () { // The condition is missing. ...
                              }
```

1209: illegal expression

There was something wrong with the expression. Another error has probably already been reported. (e)

1210 to **1216: (compiler error)**

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please notify Customer Support. (f)

For users searching online: 1211, 1212, 1213, 1214, 1215, 1216.

1219: (internal error)

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1221: don't know size of object

The **sizeof** operator is used on an incompletely specified array or undefined structure, or an array of objects of unknown size is declared. (e)

```
 extern int ar[]; sz = sizeof(ar);
```

1224: type must have default constructor

The class must have a default constructor. (e)

1227: EOF in comment

The source file ended in a comment. (w) if **-Xpcc**, (e) otherwise.

1228: too many characters in character constant

A character constant has more than four characters. The limit is four on 32 bit machines. (e)

```
int i1 = 'abcd';  /* ok */
                            int i2 = 'abcde'; /* not ok */
```

1229: EOF in character constant

The source file ended at an unexpected place during parsing. (f)

1230: newline in character constant

For example:

```
 vchar TAB = '\t;
```

1231: empty character constant

There are no characters in a character constant. If an empty string is desired, use string quotes "". (e)

```
                            int i3 = '';    /* This is two single quotes characters. */
```

1232: too many characters in wide character constant

1234: newline in wide character constant

A newline is in a wide character constant.

Example: in the following, the wide character constant is intended to be **L'ab'**, but is broken across two lines.

```
int i = L'a b';
```

1235: empty wide character constant

Empty wide character constants are not allowed:

```
 int i = L'';
```

1236: EOF in string constant

The source file ended at an unexpected place during parsing. (f)

1237: newline in string constant

The end of a line was found while parsing a string constant. Usually caused by a missing double quote character at the end of the constant. (e)

```
                            char * message = "Not everything that counts can be counted.
```

1238: illegal hex constant

Reported whenever an "**x**" or "**X**" is found in a numeric constant and is not prefixed with a single zero. (e)

```
i = 1xab;
```

1239: too long constant

A numeric constant is longer than 256 characters. (e)

1240: floating point value (...) out of range

A floating point constant exceeds the range of the representation format. (e)

```
double d = 1e10000;
```

1241: floating point overflow

Floating point overflow occurred during constant evaluation. (e)

```
float f=4E200;
```

1242: bad octal constant

A numeric constant with a leading zero is an octal constant and can only contain digits 0 through 7. (w)

```
i = 078;    // '8' is invalid in an octal
                            constant
```

1243: constant out of range

Constant overflows its type. (e)

```
                            int i = 4294967299; // Constant bigger than ULONG_MAX
```

1243: constant out of range [*operator* **]**

A constant is out of the range of the context in which it is used. If the operator is present, it shows the operator near the use of the invalid constant. (w)

```
int j = 0xfffffffffff;
```

1244: constant out of range (*string***)**

An invalid constant was used. (w)

```
const int x=0xfffffffff; if ((char)c==257)
                            ...
```

1245: illegal character: 0*n* **(octal)**

The source file contains a character with octal code *n* that is not defined in the C language. This can only occur outside of a string constant, character constant, or comment. (e)

```
name$from$PLM = 1;
```

1246: no value associated with token

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1247: syntax error after *string***, expecting** *string*

The expression is missing a semicolon or some token. (e)

```
int i
```

1248, 1249: label *identifier* **already exists**

A label can only refer to a single place in a function. (e)

1250: label *identifier* **not defined**

The label used in a **goto** statement is not defined. (e)

1251: label *identifier* **not used**

The label is never used. One possible cause is the misspelling of a label. This message appears if the **-Xlint** option is used. (w)

```
main(){ agian:                // typo? goto
                             again; }
```

1252: typedef specifier may not be used in a function definition

Bad use of the **typedef** specifier. (e)

```
typedef int foo() { }
```

1253: virtual specifier may only be used inside a class declaration

Function cannot be declared **virtual** outside class body. (e)

```
struct A { foo(); };
                             virtual A::foo() {} // Not virtual in the class declaration
```

1254: redefinition of function

The function is already defined. (e)

```
int foo() {} int foo() {}
```

1255: unions may not have base classes

Union cannot have base classes. (e)

1256: unions can't be base classes

Union cannot be used as base classes. (e)

1257: inconsistent exception specifications

Two function declarations specify different exceptions. (e)

```
int foo() throw (double);
                             int foo() throw (int);
```

1258: exception handling disabled

Exception handling has been turned off. Use **-Xexception=1** to enable it. (e)

1259: rtti disabled

RTTI (run-time type information) can be enabled or disabled through the **-Xrtti-...** option. See the User's Guide for more information on this option.

1260: non-unique struct/union reference

In PCC mode (**-Xpcc**) the compiler attempts to locate a member of another **struct** if given an invalid reference. If no unique member can be found, this error is issued. (e)

```
struct a { int i; int m; };
                             struct b { int m; int n; }; int i; ... i->m = 1;
```

1261: insufficient access rights to *member-name* in *base-class-name* **base class of** *derived-class-name*

Attempt to access a member in a **private** or **protected** base class. (e)

1264: main can't be overloaded

Special rules for the function **main( )** are violated. (e)

1265: can't distinguish *function_name1* from *function_name2*

Two overloaded functions cannot be distinguished from each other; they effectively have the same number and types of arguments in the same order. (e)

```
int foo(int); int foo(int &);
```

1266: function *function-name* **already has "C" linkage**

Only one of a set of overloaded functions can have "**C**" linkage. (e)

```
extern "C" foo(int);
                            extern "C" foo(double);
```

1268: only virtual functions can be pure

Pure specifier found after non-**virtual** function. (e)

```
class foo { bar() = 0 // Must be virtual };
```

1269: *identifier* **is not a struct/class/union member**

The identifier is not a member of a structure, class, or union. (e)

```
int i; i.j = 3; // j is not a member of a
                            structure.
```

1272: member *name* **used outside non-static member function**

Attempt to reference a class member directly in a **static** member function or an inlined **friend** function. That is invalid in a function where keyword **this** cannot be used. (e)

1275: *error string*

This error number can indicate a number of different kinds of errors. In some cases, this message gives additional information about an error message displayed above this one. For example, if a function call is ambiguous, this error prints the names of candidate functions.

1276: can't use ... in default argument expression

Class members can only be used in default arguments if they are **static**. Function arguments cannot be used in default arguments. Local variables cannot be used unless they are declared **extern**. (e)

```
int foo(int  a, int b = a) { ... }
```

1278: can't restrict access to *identifier*

An access declaration cannot restrict access to a member that is accessible in the base class, nor can an access declaration enable access to a member that is not accessible in the base class. (e)

1279: can't enable access to *identifier*

1281: no function matches call to *string*

The compiler did not find a match for a class method, or a **template** function. This can also indicate that a class does not have a default constructor. (e)

```
class line{ public: line(){} }; line
                            l(5,6);
```

Second example:

```
template< class T> T max(T a, T b) {
                            return(a>b) ? a : b; main(){ int i; char c; max(i,c); }
```

1282: can't resolve function call, possible candidates:

An overloaded function was called, but the function arguments did not match any prototype. (e)

```
fun(int i){} fun(char c){} main(){ float f;
                            fun(f); }
```

1285: ambiguous reference to *identifier***, could be** *candidate1 candidate2 ...*

The identifier could not be resolved unambiguously. The error message is followed by a list of possible candidates. (e)

```
struct A { int a; }; struct B { int a; };
                            struct C : public A, public B {}; foo() { C c;
                            c.a = 1; // Which a, A::a or B::a? }
```

1288: return type not compatible with ...

A virtual function has a return type that is incompatible with the return type of the **virtual** function in the base class. (w)

1292: too many arguments for function style cast to *string*

Function style casts to a basic type or a union type can only take a single argument. (e)

```
 int i = int(3.4, 5.6);
```

1293: non-type in new expression

A **new** expression requires a type.

```
class list {}; . . . class list * cp; cp = new
                            lis;    // Spelled wrong
```

1294: type in new expression is abstract

The type in a **new** expression must not be abstract.

1295: first dimension must be an integral expression

The first dimension of an array type in a **new** expression must be an integral expression. (e)

```
double d; int *p = new int[d];
```

1296: can't create void objects

The type in a **new** expression was void.

```
 void *p = new void;
```

1297: type in new expression is incompletely specified

1298: object of abstract class

Attempt to declare an object of an abstract class. (e)

1298: can't construct object of abstract type

The type in a **new** expression is of abstract class. (e)

```
struct A { virtual foo() = 0; }; A *p =
                              new A;
```

1299: can't construct objects of array type

Array elements in an array allocated with **new** cannot be given initial values. (e)

```
struct A {};
                              A *p = new A[5](1,2,3,4,5);
```

1304: already volatile

A variable was declared **volatile** more than once. (w)

```
 int * volatile volatile foo;
```

1305 to 1336: (compiler error)

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

For users searching online: 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, and 1336.

1337: EOF in inline function body

The end of the source file was found while parsing an inline function. (f)

1338: arguments do not match template

The actual **template** argument types must match the declaration exactly. (e)

```
template<int size> class foo { // ... };
                              foo<7, 7> qux;
```

1339: arguments do not match template *template name*

The arguments do not match the **template**.

```
template<class T> class C{}; C<int,
                              int> WrongArgs;
```

1340: can't recover from earlier errors

Certain earlier errors have made it impossible for the parser to continue. (f)

1341: compiler out of sync: mismatching parens in inline function

The compiler is unable to parse an inline function. Check the function to see if the parentheses are nested correctly. (f)

1344: syntax error - unexpected end of file

The parser has found an unexpected token. (e)

1347: identifier *name* **used as template** *name*

The identifier cannot be used as a **class**, **struct**, or **union** tag since it is already a **template** name. (e)

```
template<class T> class foo { ... };
                            struct foo { }
```

1354: "0" expected in pure specifier

A value other than 0 was found in a pure specifier. (e)

```
class foo {
                            virtual bar() = 5; // Should have been 0 }
```

1355: all dimensions but the first must be positive constant integral expressions

The first dimension of an array may be empty in some contexts. In a multi-dimensional array, no other dimensions may be empty (and none may be negative). (e)

```
 int array[-4];
```

1360: base class expected

Base class not found after "**:**" or "**,**" in a class definition. (e)

```
 class A : {}; // The base class is
                            missing
```

1361: can't initialize ... with a list

An object of a class which has constructors, bases, or non-public members cannot be initialized as an aggregate.

```
struct foo { private: int i public: int j, k;
                            }; foo bar = { 1, 2, 3 }; // i is private
```

1362: can't nest function definitions

Functions cannot be defined inside other functions.

```
void foo() { void bar() { } // No nesting
                            }
```

1367: class *class-name* **used twice as direct base class**

Cannot use the same class as a base class more than once. (e)

```
class A {}; class B : A, A {};
```

1368: class name expected after ~

Encountered "**~**" in a class, apparently to declare a destructor, but it was not followed by the class name. (e)

```
class foo { ~; };
```

1370: class/struct/union cannot be declared *specifier*

A function specifier is applied to a definition of a **class**, **struct**, or **union**. (e)

```
inline class foo {    /* inline is invalid for a class  */ ...
                            };
```

1371: conflicting declaration specifiers: *specifier1 specifier2*

Illegal mixing of **auto**, **static**, **register**, **extern**, **typedef** and/or **friend**. (e)

```
 extern static int foo;
```

1372: conflicting type declarations

More than one type specified in a declaration. (e)

```
int double foo;
```

1373: enumerator may not have same name as its class

Only constructors and destructors for a class may have the same name as the class. (e)

1376: function *function name* **is not a member of class** *class name*

A function was not declared, it was misspelled, or the parameters were not used consistently. (e)

```
class line{ lint(int l); // Misspelled };
                            line::line(int l){}
```

1378: function *function name* **is not found**

A function call referred to a function that was not found. (e)

```
static int fun(); main(){ fun(); }
```

1379, 1380: identifier ... declared as struct or class. Use struct or class specifier identifier ... declared as union. Use union specifier

There was a type mismatch between the declaration and the use of an identifier. (e)

```
union u { ... };
                            struct u foo; // u was a union, cannot also be struct
```

1381: identifier *name* **not a nested class nor a base class**

Something that is not a class was used as a base class. (e)

1383: identifier *identifier* **is not a type**

What appeared to be a declaration began with an identifier that is not the name of a type.

```
INT I;
```

1384: identifier *name* **not a direct member**

Attempt to initialize a variable that is not a direct member of the class. (e)

```
struct B { int b; }; struct C : public B { int
                            c; C(int i) : c(i), b(i) {} // Can't initialize b here }
```

1385: identifier *identifier* **not a static member of class** *class name*

Invalid declaration. (e)

```
struct A { int i; }; int A::i;
```

1386: identifier *identifier* **not declared in** *string*

An identifier is used but not declared. Check the *identifier* for spelling errors. (e)

1388: identifier *identifier* **not declared**

An identifier was used without being declared. (e)

1391: identifier *name* **is not a class**

An identifier that is not a class was used before "**::**".

1394: illegal *expression*

A **break** statement is only allowed inside a **for**, **while**, **do** or **switch** statement. (e)

A **continue** is only allowed inside a **for**, **while** or **do** statement. (e)

A **default** or **case** label is only allowed inside a **switch** statement. (e)

1395: illegal function specifier for argument

A parameter cannot be declared **inline** or **virtual**.

```
void foo(inline int);
```

1397: illegal storage class for class/struct/union

A storage class other than **extern** is specified for a definition of a **class**, **struct**, or **union**. (e)

```
auto class foo { ... };
```

1403: main can't be declared string

Special rules for the function **main( )** are violated. (e)

1404: mem initializers only allowed for constructors

Members can only be initialized with the member initializer syntax in constructors. (e)

```
class A { int i;
                            int foo() : i(4711) {} // Not a constructor }
```

1405: missing argument declaration

Argument declaration omitted. (e)

```
class bar { foo(, int); };
```

1410: no default arguments for overloaded operators

Overloaded operators cannot have default arguments. (e)

1411: no redefinition of default arguments

An argument can be given a default value only once in a set of overloaded functions. (e)

```
void foo(int = 17);
                            void foo(int = 4711);
```

1412: no return type may be specified for conversion functions

The return type of conversion function is implicit. (e)

```
class foo {
                            double operator int(); // Cannot specify type }
```

1414: non-extern object *name* **of type** *type-name* **must be initialized**

A **const** object must be initialized unless it is **extern**.

1415: non-extern reference *name* **must be initialized**

References and **const** objects, which are not declared **extern**, must be initialized. So must objects of classes that have constructors but no default constructors. (e)

```
const struct S &structure;
```

1417: only functions can have pascal calling conventions

For example:

```
int pascal i;
```

1418: only static constant member of integral type may have initializer

A member that is a static integral type can be initialized; others cannot. (e)

```
struct { const int *p =0x3333; }s;
```

1419: operator ... cannot be overloaded

It is invalid to overload any of the operators "**.**" or "**.***" or "**? :**" .

1420: parenthesized expression-list expected after type *typename*

1423: redeclaration of symbol ...

A symbol in an enumerated type clashes with an earlier declaration. (e)

1427: static function declared in a function

There is no use declaring a **static** function inside another function. (e)

```
void foo() { static void bar();
                            bar(); // Call to bar, but where can it be defined? }
```

1428: static member ... can't be initialized

A **static** class member cannot be initialized in a member initializer. (e)

```
class A { static int si; A(int ii) : si(ii) {}
                            };
```

1429: string literal expected in asm definition

String missing in an **asm** statement.

```
                            asm();    // the parentheses should contain an instruction
```

1430: subsequent argument without default argument

Only the trailing parameters may have default arguments. (e)

```
void foo(int = 4711, double);
```

1431: syntax error - catch handler expected after try

The parser has found an unexpected token. (e)

1432: syntax error - catch without matching try

The parser has found an unexpected token. (e)

1433: syntax error - class key seen after type. Missing ;?

The parser has found an unexpected token. (e)

1434: syntax error - class name expected after ::

The parser has found an unexpected token. (e)

1435: syntax error - colon expected after access specifier

The parser has found an unexpected token. (e)

1436: syntax error - declarator expected after ...

The parser has found an unexpected token. (e)

1437: syntax error - declarator expected after type

The parser has found an unexpected token. (e)

1438: syntax error - declarator or semicolon expected after class definition

The parser has found an unexpected token. (e)

1439: syntax error - else without matching if

The parser has found an unexpected token. (e)

1441: syntax error - identifier expected after ...

The parser has found an unexpected token. (e)

1442: syntax error - initializer expected after =

The parser has found an unexpected token. (e)

1444: syntax error - keyword operator must be followed by an operator or a type specifier

The parser has found an unexpected token. (e)

1446: syntax error - type tag expected after keyword enum

The parser has found an unexpected token. (e)

1454: type defined in return type (forgotten ";"?)

It is illegal to define a type in the function return type. (e)

```
struct foo {} bar() { }
```

1455: type definition in bad context

A type was defined where it was not allowed. (e)

1456: type definition in condition

Types cannot be defined in conditions. (e)

```
if (struct foo { int i } bar) { // ...
                              }
```

1457: type definition not allowed in argument list

Types cannot be defined in argument lists. (e)

```
                              int foo( struct bar int a; } barptr);
```

1460: type expected after new

A **new** expression requires a type. (e)

```
p = new;
```

1461: type expected for ...

No type found in declaration of a variable. (e)

1462: type expected in template parameter

This could indicate a misspelling of a **template** parameter. (e)

```
template<classT> ...;
```

1463: type expected in arg-declaration-clause

An argument type is missing in a function declaration. (e)

```
class bar { foo(imt); };
```

1464: type expected in cast

Found something that was not a type in a cast expression. (e)

1465: type expected

Found an expression that was not a type where a type was expected. (e)

1466: type in new expression can't be *string*

A type in a **new** expression cannot be **pascal** or **asm**.

1467: type in new expression may not contain class/struct/enum declarations

Cannot declare types in a **new** expression. Nor can the types used in a **new** expression be **const**, **volatile**, **pascal**, or **asm**. The type used must be completely specified and cannot have pure virtual functions. (e)

```
void *p = new enum foo { bar };
```

1469: unknown language string in linkage specifier: ...

Only "**C**" and "**C++**" allowed in linkage specifiers. (e)

```
extern "F77 { // Don’t know anything about F77 linkage}
```

1477: already const

A variable was declared **const** more than once. (w)

```
int * const const foo;
```

1479: comma at end of enumerator list ignored

A superfluous comma at the end of a list of enumerators was ignored. (w)

```
enum foo { bar, };
```

1480: enumerators can’t have external linkage

extern cannot be specified for **enum** declarations. (e)

```
extern enum foo { bar };
```

1481: function *function-name* **not declared**

If the -Xforce-declarations option is used, the compiler will generate this error message when a function is used before it has been declared. (w)

1484: missing declarator in typedef

No declarator was given in a **typedef** statement. (e)

```
typedef class foo { // ... };
```

1485: old style function definition

A function was defined using the older K & R C syntax. This is invalid in C++. (w)

```
int foo(a, b) int a, b { ... }
```

1486: initializer that is a brace-enclosed list may contain only constant expressions

A variable was initialized using a brace-enclosed list containing an expression (such as a variable) that cannot be evaluated during compilation.

```
int i = 12; ... int x[] = { 1, 2, 3 , i
                            };
```

This is allowed in C++ but not in C.

1488: redeclaration of parameter *identifier*

One of a function's parameters is shadowed by a declaration within the function, (w) if **-Xpcc** or **-Xk-and-r**, (e) otherwise.

```
 f1(int a) { int a; ... }
```

1489: redundant semicolon ignored

Found an extra semicolon among the members of a function. (w)

```
class A { int a; ; };
```

1492: virtual specified both before and after access specifier

Syntax error. (w)

```
class A {};
                            class B : virtual public virtual A {};
```

1493: redeclaration of ...

A function has been redeclared to something else. (e)

```
int i(int); double i(int); double i( int i)
                            {...}
```

1494: non-extern object identifier of type *type-designator* **must be initialized**

This message may indicate that a **const** member of a class/structure/union was not initialized. (e)

```
class C { const int ci; } c;
```

1495: non-extern const object *name* **must be initialized**

A **const** object must be initialized unless it is **extern**.

```
 const char c;
```

1497: too many declaration levels

An internal stack overflowed. This is unlikely to happen in the absence of other errors. (f)

1498: internal table-overflow

Internal stack overflowed. May occur with extremely complex, deeply nested code. To work-around, simplify or modularize the code. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1499: dcc

Error message caused by either an empty file, a file containing only comments, or a file that otherwise does not include code that can be compiled into an executable.

```
#if 1 /* * */ #else int main (void) { printf
                            ("Code not included"); return 0; } #endif
```

1500: function *<function_name>* **has no prototype**

The function *function_name* was used without a preceding prototype declaration. In C,

```
void f();
```

is a declaration but not a prototype declaration—it declares *f* to be a function but says nothing about the number or type of arguments it takes. This warning is returned when an attempt has been made to use *f* without making a prototype declaration of it first.

This warning is returned only when the command line option -Xforce-prototypes is used. (w)

1501: function-pointer has no prototype

A function pointer was used but was declared to have a type that lacks a prototype. In C,

```
void (*f)();
```

declares *f* to be a function pointer but says nothing about the number or type of arguments it takes. This warning is returned when an attempt has been made to use *f* without making a prototype declaration of it first.

This warning is returned only when the command line option -Xforce-prototypes is used. (w)

1504: arglist in declaration

An old style function declaration is found in the wrong context. (w)

```
f1() { int f2(a,b,c); ... }
```

1507: end of memory

Ran out of virtual memory during compilation. The compiler first attempts to skip some optimizations in order to use less memory, however this error can occur for large functions on machines with limited memory. Note: initialized arrays require the compiler to hold all initial data and can contribute to this error. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1509: expression involving packed member too complicated

This indicates that the processor does not support “compound assignment” for volatile members of packed structures.

```
struct1.a |=3; // May have to use struct1.a =
                            struct1.a|3
```

1511: can't access short or int bit-fields in packed structures unless the architecture supports atomic unaligned accesses (-Xmin-align=1)

Packed structures cannot contain bit-fields unless the architecture support atomic unaligned access. To see if the architecture supports atomic unaligned access, compile a file with the **-S** option and then examine the **.s** assembly file. Look for the **-X93** option in the header. If **X93**=1, the architecture supports atomic unaligned access. (e)

```
#pragma pack(1) struct S { int j:3;
                           };
```

1513: byte swapped structures can't contain bit-field

Bit-fields are not allowed in byte-swapped structures. (e)

```
#pragma pack (,,1)  // Byte swap struct s { int
                           j:3; }
```

1515: profile information out of date

The file given with the -Xfeedback option is out of date or has an old format. Re-compile with the -Xblock-count option and create a new profiling file. (e)

1516: parameter *parameter name* **is never used**

A parameter to a function is not used. This message appears if the -Xlint option is used. (w)

```
 fun(int i){};
```

1517: function *function name* **is never used**

A **static** function was declared but not used. This message appears if the -Xlint option is used. In the example, the file consists of one line. (w)

```
 static fun();
```

For C89, inline functions return a different warning; see #tmt1501705396361__dt1788 on page 49.

1518: variable *identifier* **is never used**

A variable is never used. This message appears if the -Xlint option is used. (w)

```
fun(){ int i; }
```

1519: expression not used

The compiler has detected all or part of an expression which will never be used. (w)

```
a+b;            /* statement with no side effects */
                           a=(10,b+c);     /* 10 is not used */
                           *s++;           /* the '*' is not needed: s++; /
```

Note: the compiler will not issue this warning for an expression consisting solely of a reference to a volatile variable.

1520: large structure is used as argument

The size of a structure passed as an argument to a function equals or exceeds the size specified by -Xstruct-arg-warning. (This message is returned only when the command-line option -Xstruct-arg-warning is used.) (w)

1521: missing return expression

A function is defined with a return type, but does not return a value. This message appears if the **-Xlint** option is used. (w)

```
float fun(){ return; }
```

1522: statement not reached

A statement can never be executed. This message appears if the **-Xlint** option is used. (w)

```
main(){ int never; return 0; never=6;
                            }
```

1523: can't recognize storage mode *unknown*

The storage mode specified in an **asm** macro is unknown. See the User's Guide for more information on embedding assembly code. (e)

1524: too many enhanced asm parameters

There can be a maximum of 20 parameters and labels used in an **asm** macro. See the User's Guide for more information on embedding assembly code..

1525: identifier *identifier* **not declared**

An identifier was not declared. (e)

```
fun(){ return i;   }
```

1526: asm macro line too long

A very long line was given in an **asm** macro. See the *User's Guide* for more information on embedding assembly code. (e)

1527: non-portable mix of old and new function declarations

A function declaration was made in accordance to an older C standard. In K & R C, **char**s and **short**s are promoted to **int**, and **float**s are promoted to **double** just before a call is made to a function. However, in ANSI C, the arguments match the prototype at the call site. (w)

1528: can't initialize variable of type *type_designator*

Some types do not allow initialization. (e)

```
void a = 1;
```

1534: only first array size may be omitted

The size of the first dimension of an array can be omitted; all others must be specified. (e)

```
int x[3][];
```

1535: illegal width of bit-field

A bit-field width is greater than the underlying type used for the bit-field. (e)

Example for a target with 32 bit integers:

```
struct { int i:33; }
```

1536: bit-field must be int or unsigned

The compiler detected an unsupported bit-field type. (e)

```
struct { float a:4; };
```

1541: redeclaration of struct/union/enum ...

A **struct**, **union**, or **enum** tag name was used more than once: (e)

```
                              struct t1 { ... }; struct t1 { ... };
```

1542: redeclaration of member *variable name*

A member has been declared more than once. (e)

```
struct{ int i; int i; };
```

1543: negative subscript not allowed

The size of an array cannot be negative. (e)

```
 int ar[-10];
```

1544: zero subscript not allowed

An array of zero size cannot be declared when compiling for strict ANSI C (**-X7=2**, or **--Xdialect-strict-ansi**). (w)

```
 int x[0];
```

1546: dangerous to take address of member of packed or swapped structure

Using the address of a packed or byte-swapped structure is not recommended. (w)

```
#pragma pack (2,2,1) . . . ptr =
                              &(struct1.i);
```

1547: can't take address of object

Trying to take the address of a function, constant, or register variable that is not stored in memory. (e)

```
 register int r; fn(&r);
```

1548: can't do sizeof on bit-field

The **sizeof** function does not work on bit-fields. (e)

```
struct { int j:3; } struct1; i =
                              sizeof(struct1.j);
```

1549: illegal value

Only certain expressions can be on the left hand side of an assignment. (e)

```
a+b = 1; (a ? b : c) =
                              2;    /* not valid in C modules*/
```

1550: can't push *identifier*

It is invalid to use an expression of type function or **void** as an argument. (e)

```
                              void *pv; int (*pf)(); fn(*pv,*pf);
```

1551: argument [*identifier***] type does not match prototype**

The type of an argument to a function is not compatible with its type as given in the function's prototype. (w) if **-Xpcc** or **-Xk-and-r** or **-Xmismatch-warning,** (e) otherwise.

```
                              int f(char *), i; ... i = f(&i);
```

1552: initializer type "*type* " **incompatible with object type** "*type* "

The type of an initializer is not compatible with the type of the variable, (w) if **-Xpcc** or **-Xmismatch-warning**, (e) otherwise.

```
char c; int *ip = &c;
```

1553: too many errors, good bye

The compiler has found so many errors that it does not seem worthwhile to continue. (f)

1554: illegal type(s): *type-signatures*

The operators of an expression do not have the correct or compatible types, (w) if **-Xpcc** or **-Xk-and-r** or **-Xmismatch-warning**, (e) otherwise. This message may also indicate an attempt has been made to find the sum of two pointers.

```
int *pi, **ppi; ... if (pi == ppi) ...
                            #illegal types: ptr-to-int '==' ptr-to-ptr-to-int int *p, *q; p =
p +
                            q;   // Attempt to add pointers
                            #illegal types: ptr-to-int '+' ptr-to-ptr-to-int
```

1555: not a struct/union reference

The left hand side of a "**->**" or "**.**" expression is not of **struct** or **union** type. If **-Xpcc** is specified the offset of the given member name in another **struct** or **union** is used. (w) if **-Xpcc**, **-Xk-and-r**, or **-Xmismatch-warning**, (e) otherwise.

1556: volatile packed member cannot be accessed atomically

For the selected processor, a packed member cannot be accessed atomically if it is **volatile**. (w)

```
#pragma pack(1, 1) struct { volatile int v;
                            } s; s.v =3;              /* generates error 1556 */
```

1560: unknown pragma

The **pragma** is not recognized. (w)

```
#pragma tist
```

1561: unknown option -X*unknown*

The compiler was started with an **-X** option that is not recognized. (w)

1562: bad #pragma use_section: section *section name* **not defined**

A **#pragma use_section** command has not been correctly given. (w)

```
#pragma section DATA3  // Correct #pragma
                            use_section x  // Omitted section class name DATA3
```

1563: bad #pragma [*name* **]**

If issued without the *name*, the compiler did not recognize the pragma. If issued with a *name*, there is a problem with either the operands to the **pragma** or the context in which it appears. (w)

1564: bad #pragma pack

The **#pragma pack** statement is not correct. (w)

```
#pragma pack(1,2,3,4) // Takes up to three
                            arguments
```

1565: illegal constant in #pragma pack

An invalid constant has been used in a **pack pragma**. (w)

```
#pragma pack(7) // Must use powers of 2 for
                           alignment
```

1566 to 1572: obsolete messages

Messages numbered 1566 to 1572 should not appear because they refer to obsolete features.

1573: user's error string

Error number 1573 displays a message from any of the following directives:

- **#info**
- **#warning**
- **#error**
- **#pragma info**
- **#pragma warning**
- **#pragma error**

This message can also appear along with other compiler messages, where it supplies additional information about the error.

1574: can't open *file* **for input**

The given *file* cannot be opened. (f)

1575: can't open *file* **for output**

The given *file* cannot be opened. (f)

1577: can't open profiling file *file*

The file given with the **-Xfeedback=***file* option cannot be opened. (w)

1578: profile file is of wrong version (*file***)**

The file given with the **-Xfeedback** option is out of date or has an old format. Re-compile with the **-Xblock-count** option and create a new profiling file. (e)

1579: profile file *file* **is corrupted**

The file given with the **-Xfeedback** option is corrupted. Re-compile with the **-Xblock-count** option and create a new profiling file. (e)

1580: can't find current module in profile file ...

No data about the current source file is available in the profiling file. (w)

Possible causes:

- No function in the current file was actually executed during profiling.
- The profiling file belongs to another executable program.

1583: overflow in constant expression

The constant expression overflows the underlying data type. For example, the expression:

```
return 0x40000000 + 0x40000000;
```

will flag this warning. Here, the type is a signed **int** and this operation will result in an overflow. However, if the constant is updated with a size specifier, for example:

```
return 0x40000000U + 0x40000000U;
```

No warning is issued.

1584: illegal declaration-attribute

A declaration contains an invalid combination of declaration specifiers. (w)

```
 unsigned double foo;
```

1585: global register *register name* **is already used**

The global register has already been reserved. (w)

```
#pragma global_register counter = r14 #pragma
                           global_register kounter = r14
```

1586: cannot use scratch registers for global register variables

Scratch registers cannot be used for global register variables. (w)

```
 #pragma global_register counter=scratch-register-name
```

1587: global register *register-name* **is invalid**

Found an unrecognized register name in a **global_register pragma**. (w)

1588: no .cd file specified!

The target description (**.cd**) file was not specified.

The compiler reads a *target description file* during initialization (see the User's Guide). Normally, when the **dcc** command is given, the **.cd** file is automatically specified. To find out the **.cd** filename for your selected target configuration, run **dcc** with the **-#** option to display all of the commands generated, and look at the **-M** option for the **ctoa** program. (f)

Likely causes:

- The compiler is not installed properly.
- One of the compiler files has been deleted, hidden, or protected.
- The **dtools.conf** or other configuration file is incorrect.

1589: can't open .cd file!

See error 1588 for a description of the **.cd** file and likely causes.

1590: .cd file is of wrong type!

See error 1588 for a description of the **.cd** file and likely causes.

1591: .cd file is of wrong version!

See error 1588 for a description of the **.cd** file and likely causes.

1592: cd file *file* **too small?!**

See error 1588 for a description of the **.cd** file and likely causes.

1593: rite error

Write to output file failed. (f)

1594: internal consistency check fails

Please contact customer support.

1595: illegal arg to *function name*

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1596: test version of compiler: File is too big!

This error is generated when certain limits in an evaluation copy of the compiler are exceeded. (f)

1597: test version of compiler: Can't continue!

This error is generated when certain limits in an evaluation copy of the compiler are exceeded. (f)

1598: no matching asm pattern exists

While scanning an **asm** macro, no storage-mode-line matching the given parameters was found. See the User's Guide for more information on embedding assembly code.

1599: expression too complex. Try to simplify

Can occur if an expression is too complex to compile. Should not happen on most modern processors. Can occur on a processor with few registers and no built-in stack support. (f)

1600: no table entry found!

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1601: address taken in initializer (PIC)

Position-independent code. A static initializer containing the address of a variable or string has been found when generating position-independent code. Such address values cannot be position-independent. (w) or (e) depending on whether **-Xstatic-addr-warning** or **-Xstatic-addr-error** is used.

1602: variable ... is incomplete

A variable is defined with a type that is incomplete. (e)

```
struct a; struct a b;
```

1603: logic error in *internal-identification*

The compiler has detected an internal error. May result from other errors reported earlier. If the problem does not appear to be a consequence of some earlier error, please report it to Customer Support. (f)

1604: useless assignment to variable *identifier* **. Assigned value not used**

The variable assignment has no effect, since the assigned value is not used. This message appears if the **-Xlint** option is used. (w)

```
fun(){ int i=1; }
```

1605: not enough memory for reaching analysis

Certain optimizations, called “reaching analysis”, will be skipped if the host machine cannot provide enough memory to execute them. The compiler continues, but produces less than optimal code. (w)

1606: conditional expression or part of it is always true/false

A conditional test is made, but the results will always be the same. This message appears if the **-Xlint** option is used. (w)

```
int main(){ int i = 3; if (i < 6) return 4;
                              }
```

1607: variable *name* **is used before set**

During optimization, the compiler discovers a variable that is used before it is set. (w)

```
func() { int a; if (a == 0) ... }
```

1608: variable *identifier* **might be used before set**

A variable may have been used before it was given a value. (w)

```
fun(){ int i,j; i = j;     // j is used before
                              set }
```

1609: illegal option -**D***invalid_name*

The preprocessor was invoked with the **-D** option and an invalid name. Names must start with a letter or underscore. (w)

1611: argument list not terminated

The end of the source file was found in a macro argument list. (w) if **-Xpcc**, (e) otherwise.

1612: EOF inside #if

The source file ended before a terminating **#endif** was found to match an earlier **#if** or **#ifdef**. If not caused by a missing **#endif**, then it is frequently caused by an unclosed comment or unclosed string. (w) if **-Xpcc**, (e) otherwise.

1617: syntax error in #if

The expression in an **#if** directive is incorrect, (w) if **-Xpcc**, (e) otherwise.

```
#if a *
```

1618: too complex #if expression

The expression in an **#if** directive overflowed an internal stack. This is unlikely to happen in the absence of other errors, (w) if **-Xpcc**, (e) otherwise.

1619: include nesting too deep

The preprocessor cannot nest header files deeper than 100 levels, (w) if **-Xpcc**, (e) otherwise.

1621: can't find header file *unknown*

The preprocessor cannot find a file named in an **#include** directive. (w) if **-Xpcc**, (e) otherwise.

1622: found #elif, #else, or endif without #if

Found an **#elif**, **#else**, or **#endif** directive without a matching **#if** or **#ifdef**. (w) if **-Xpcc**, (e) otherwise.

1623: bad **include syntax**

The **#include** directive is not followed by **<** or **"** or the filename is too long. (w) if **-Xpcc**, (e) otherwise.

1624, 1625: illegal macro name illegal macro definition

Macro names and arguments must start with a letter or underscore, (w) if **-Xpcc**, (e) otherwise.

1626: illegal redefinition of *macro_name*

__LINE__, __FILE__, __DATE__, __TIME__, **defined**, and **__STDC__** cannot be redefined, (w) if **-Xpcc**, (e) otherwise.

1627: macro *macro name* **redefined**

The macro was previously defined. (w)

```
#define PI 3.14 #define PI 3.1416
```

1629: undefined control

Undefined or unsupported directive found after **#**, (w) if **-Xpcc, (e)** otherwise.

```
 #pragmo
```

1630: illegal assert name

An **#assert** name must be an identifier and must be preceded by a "**#**" character, (w) if **-Xpcc**, (e) otherwise.

1631: macro *identifier* **: argument mismatch**

Either too few or too many arguments supplied when using a macro, (w) if **-Xpcc**, (e) otherwise.

```
#define M(a,b)  (a+b)
                            i = M(1,2,3);
```

1632: recursive macro *macro name*

A recursive macro has been detected. The error occurs when the macro substitution occurs, line 4 in this case: (e)

```
#define max(A,B) A>B ? A : max(A,B) main(){
                            int i=1,j=2,k; k = max(i,j); // Reports error for this line.
                            }
```

1633: parse error

The compiler was not able to parse the expression. (e)

```
x = multiply(y, ); // Comma, but no second
                            argument main(}              // Typed } instead of )
```

1635: license error: *error message*

An error occurred when checking the license for the software tools. The error message describes the problem (no server for this feature, etc.). Please refer to your Getting Started manual or contact Customer Support. (f)

1638: illegal error level *error level* **in option** *option name*

The **-e**x*n* option was used with an invalid error level. The **-e** option is used for increasing the severity of error messages for a particular error. (w)

```
 dcc -e99 test.c // 99 is invalid error
                            level
```

1640: illegal error message number *message number*

The **-e**x*n* option was used with an invalid error message number. The **-e** option is used for increasing the severity of error messages for a particular error. (w)

```
 dcc -ew10000 test.c // There is no message
                            number 10000
```

1641: cannot reduce severity of error message *number* **below** *error level*

For example:

```
% dcc -ew1614 test.c warning (dcc:1641): Cannot
                            reduce severity of message 1641 below "error"
```

1643: narrowing or signed-to-unsigned type conversion found: *type* to *type*

A type conversion from signed to unsigned, or a narrowing type conversion has been found. This message appears if the **-Xlint** option is used. (w)

```
main(){ int i; char c; c = i; }
```

1647: non-*string* **method invocation** *expression* **on** *string* **object** *expression*

This error indicates a mismatch between an invocation and the declaration of a method.

For example, non-**const** method invocation in **const** object. Methods of **const** objects must be **const**.

```
class C { int i; public: f() { i = 12; } C() {}
                            }; const C c; main() { c.f(); } "x.cpp", line 11: error (1647):
                            non-const method invocation f() on const object c
```

1657: initializer *method name* **initializes neither a direct base nor a member**

Only classes that are direct bases or **virtual** bases can be used in a member initializer. (e)

```
struct A { A(int); };
                            struct B : public A { B(int); }; struct C : public B {
                            C(int i) : A(i) {} // Can't initialize A here };
```

1663: inline of *function* **does not occur in routine function - try increasing value of -Xinline**

This warning is generated whenever the **inline** keyword is specified but the compiler does not inline the function. Increasing the value of **-Xinline** or **-Xparse-size** can help, but there are other reasons for not inlining a function.

1665: long long bit-fields are not supported

long long cannot be used with bit-fields. (w)

```
struct { long long story:3; }
```

1671: non-portable behavior: operands of *type* **are promoted to unsigned type only in non-ANSI mode**

When a non-ANSI compilation mode is used, for example, **-Xpcc**, this warning appears when the compiler selects an unsigned integral type for an expression which would have been signed under ANSI mode. This message appears if the **-Xlint** option is used. Use **-Xlint**=0x200 to suppress this message. (w)

1672: scope of tag *tag* **is only this declaration/definition**

The tag referred to in a parameter list does not have a prior definition. (w)

```
/* struct bar does not have a definition before
                            this point */ foo(struct bar a);
```

1674: template argument *argument* **should be pointer/reference to object with external linkage**

Arguments for template functions need to be pointers or references to objects with external linkage. (e)

```
template <class T, int& Size> class
                            Base { .... }; class A { ... } ; static int local_linkage_int;
                            Base<A, local_linkage_int> ob;
```

1675: sizeof expression assumed to contain type-id type-id (use "*typename*" **)**

When a **type-id** is used in a **sizeof** expression, the compiler assumes that this is intended; otherwise a typename should be used instead. (w)

```
template <class T, int& Size> class
                            Base { ... void incr() { Size = Size + sizeof(A); } ... };
```

1676: class *class* **is abstract because it doesn't override pure virtual** *function*

A class that has un-overridden pure virtual functions is an “abstract class” and cannot be instantiated. (i)

1679: no definition found for inline function *function*

The template member function referred to has no definition. (w)

1680: delete called on incomplete type *type*

The **delete** operator is called on a pointer to a type whose full declaration has been deferred. (w)

1682: "(unsigned) long long" type is not supported by the ANSI standard

The ANSI standard does not support the **long long** type. (w; future error)

```
 long long x;
```

1683: non-int bit-fields are not supported by the ANSI standard

The ANSI standard allows bit-fields of integer type only. (w; future error)

```
struct foo { char x:2; };
```

1684: vector must occur before any other type specifiers

This message is specific to the AltiVec target.

The **vector** keyword should occur before any other type specifier. (e)

```
 int vector b;
```

1685: pixel can only be specified as part of a vector type

This message is specific to the AltiVec target.

The **pixel** keyword cannot be used for declarations outside a vector type. (e)

```
vector b; pixel c;
```

1686: long keyword for vector types is deprecated

This message is specific to the AltiVec target.

The **long** keyword for vector types has been deprecated. (w)

```
 vector long b;
```

1687: vector pixel should have no other type specifiers

This message is specific to the AltiVec target.

No other types are allowed in the declaration of a vector pixel. (e)

```
vector unsigned short pixel a; vector unsigned
                            int pixel b;
```

1688: vector float cannot have bool specifier

This message is specific to the AltiVec target.

A vector float cannot have a **bool** specifier. (e)

```
vector float bool c;
```

1689: *vector name* **is an invalid vector type**

This message is specific to the AltiVec target.

An invalid vector type is defined. (e)

```
struct X {     int a; }; vector struct X
                            checking;
```

1691: vec_step is only valid for vector types

This message is specific to the AltiVec target.

vec_step can be called only with vector types. (e)

```
int j = vec_step(char const);
```

1692: vector initialization may only contain constants

This message is specific to the AltiVec target.

A vector initialization should contain constants only. (e)

```
int c; vector int A = (vector int)(1, c, 3,
                            4);
```

1693: vector initialization - not enough initializer constants

This message is specific to the AltiVec target.

Too few initializer constants are specified for a vector. (e)

```
vector int A = (vector int)(1, 2,
                            3);
```

1694: vector initialization - too many initializer constants

This message is specific to the AltiVec target.

Too many initializer constants are specified for a vector. (e)

```
vector int A = (vector int)(1, 2, 3, 4, 5,
                            6);
```

1695: vector arguments can only be passed to vector intrinsics or functions with prototypes

This message is specific to the AltiVec target.

Vector arguments can be sent only to intrinsic functions or functions that have already been defined or declared. (e)

```
vector int A; /* foo has no definition or
                            prototype till this point    and is not an intrinsic function */
                            foo(A);
```

1696: intrinsic *function name* **must have** *n* **argument(s)**

The number of arguments passed to an intrinsic function is incorrect. (e)

```
int a, b; ... a = __ff1(a, b);
```

1697: invalid types on arguments to intrinsic *function name*

An argument of an invalid type is passed to an intrinsic function. (e)

```
char *ptr; int a; ... a =
                            __ff1(ptr);
```

1698: the number i argument to intrinsic function name must be an unsigned constant less than n

This message is specific to the AltiVec target.

The argument passed to the intrinsic function should be less than the specified value. (e)

```
int a; vector unsigned long B, C; ... C =
                            vec_splat(B, a);
```

1699: the number *i* **argument to intrinsic** *function name* **must be a signed constant between** *n* **and** *m*

This message is specific to the AltiVec target.

The argument passed to the intrinsic function should be within the specified range of values. (e)

1700: implicit intrinsic *function name* **must have** *n* **argument(s) - when the intrinsic is enabled, optional user prototype must match**

When an enabled intrinsic function is redefined, the number of arguments must be the same. (e)

```
unsigned int __ff1(unsigned int x, unsigned int
                            y) { ... }
```

1701: invalid types on prototype to intrinsic *function name* **- when the intrinsic is enabled, optional user prototype must match**

When an enabled intrinsic function is redefined, the prototypes must match. (e)

```
unsigned int __ff1(int a) { ... }
```

1702: prototype return type of intrinsic *function name* **should be** *type* **- when the intrinsic is enabled, optional user prototype must match**

When an enabled intrinsic function is redefined, the return type must match. (e)

```
void __ff1(unsigned int a) { ... }
```

1703: function name matches intrinsic *function name* **- rename function or disable the intrinsic with -Xintrinsic-mask**

A function with the same name as an intrinsic function has been defined. The function should be renamed or intrinsic functions should be disabled. (w)

```
unsigned int __ff1(unsigned int x) { ...
                            }
```

1704: structure or union cannot contain a member with an incomplete type

Structures or unions should not contain fields of incomplete type. (w; future error)

```
struct x { void a; };
```

1707: invalid pointer cast/assignment from/to __X mem/__Y mem

The pointer assignment is invalid because it is between locations in two different memory banks. (e)

1708: cannot take address of an intrinsic function

An intrinsic function, which represents a specific CPU instruction, has no location in memory.

1709: unsupported GNU Extension : inline assembler code

The compiler does not translate extended GNU inline assembler syntax (such as register usage specification). (e)

1710: macro *macroname* : **vararg argument count does not match. expected** *n* **or more but given** *m*

Too few arguments are passed to a **vararg** macro. (w)

```
#define TEST_INFO_1(fmt, val, ...) printf(fmt,
                            val, __VA_ARGS__) ... TEST_INFO_1("val1 = %d, val2 = %d",
                            12);
```

1711: undefined identifier *identifier* **used in constant expression**

An undefined macro name occurs in a **#if** preprocessor directive. To disable this warning, use **-Xmacro-undefined-warn**. (w)

```
#if (FooDef1 == FooDef2) # ...
                            #endif
```

1712: only vector literals may be used in vector initializations

Vectors can be initialized only with vector constants. (e)

```
 vector int a[2] = {1, 2};
```

1713: invalid assert name *name*

1714: invalid macro name *name*

1715: no input file given

1716: memory unavailable

1717: unterminated comment

1718: unterminated character or string constant

1719: duplicate parameter name *param* **in macro** *macro*

1720: implicit include file *"file"* **not found**

1721: missing ">" in '#include <*filename***> syntax"**

1722: junk after "#include <*filename***>"**

1723: junk after "#include "filename**"**

1724: "#include" expects <*filename***> or "filename"**

1725: #if nesting too deep

1726: #include file nesting too deep. possible recursion

1727: unmatched *condition* . **block starts on line** *n*

1728: unmatched *condition*

1729: unbalanced *condition*

1730: undefined control after *expr*

1731: EOF inside #... conditional

1732, 1733: illformed macro parameter list in macro *macro*

1734: invalid macro name *name*

1735: invalid argument to macro

1736: illformed macro invocation

1737: invalid assert name *name*

1738: "##" at start of macro definition

1739: "#" precedes non macro argument name or empty argument

1740: macro *macro* : **argument count does not match. expected** *n* **but given** *m*

1741: redefinition of macro "*macro***". previously defined here**

1742: predefined macro *macro* **redefined**

1743: empty token-sequence in "#assert"

1744: no closing ")" in "#assert"

1745: garbage at the end of "#assert"

1746: invalid number in #line

1747: only a string is allowed after #line <*num***>**

1748: string expected after #error

1749: string expected after #ident

1750: # directive not understood

1751: "defined" without an identifier

1752: no closing ")" in "defined"

1753: bad digit in number

1754: bad number in #if...

1755: floating point number not allowed in #if...

1756: wide character constant value undefined

1757: undefined escape sequence in character constant

1758: empty character constant

1759: multi-character character constant

1760: octal character constant does not fit in a byte

1761: hex character constant does not fit in a byte

1762: character constant taken as unsigned

1763: garbage at the end of *condition* **argument**

1764: illegal identifier *identifier* in *condition*

1767: can't find include file *file* **in the include path**

1768: invalid "vector bool" constant, valid values 0, 1 or -1

1769: the called object is not a function

1770: array is too large

There is a physical limitation on the amount of space that can be allocated for an array. (e)

1771: reserved identifiers "__FUNCTION__" and "__PRETTY_FUNCTION__" may only be used inside a function

The special identifiers **__FUNCTION__** and **__PRETTY_FUNCTION__**, which return the name of the current function, can be used only within a function. (e)

1772: possible redundant expression

The compiler has encountered a valid but redundant operation, such as **x**&**x**. This message appears if the **-Xlint** option is used. (w)

1773: quoted section name cannot be empty, set to: *default name*

Quoted section names cannot be empty (**""** or **" "**). For example,

```
.section " ",4,rx
```

will be changed to:

```
.section "default_section_name",4,rx
```

where the default section name is determined by context. (w)

1774: asm macro must be completed with "}" in the very first position

An **asm** macro must conclude with a right brace ("**}**") in the first column of a new line. The example below shows a valid **asm** macro. (e)

```
asm void setsr (unsigned short value) {
                          %mem   value;    move.w  value,d0    move.w  d0,sr }
```

1775: deprecated use of constructor/destructor ignored, use attribute keyword

The compiler encountered an initialization or finalization function declared with the obsolete prefix **_STI__***nn* _ or **_STD__***nn* _. Use the **__attribute__** keyword to identify initialization and finalization functions, or specify **-Xinit-section=2** to use old-style initialization and finalization sections. (f)

1776: constructor/destructor priority out of range (*number***)**

The specified priority is out of range. The default range is 0-65535; but if **-Xinit-section=2** is enabled, the range is 0-99. (e)

1777: default constructor/destructor priority out of range, setting to lowest

The priority for default constructors and destructors has been set with **-Xinit-section-default-pri** to a value that is out of range. The default range is 0-65535; but if **-Xinit-section=2** is enabled, the range is 0-99. (w)

1778: option -Xc++-old is deprecated and dtoa will be removed in a future release

-**Xc++-old**, which invokes an obsolete version of the C++ compiler, will not be supported indefinitely. Legacy projects should be ported to the latest C++ compiler. See the User's Guide for more information. (w)

1779: CODE section without execute access mode: *section-name*

A **CODE** section has been created with a specified access mode that does not include execute permission. For example:

```
#pragma section CODE ".SOME_CODE_SECTION" RW
                            far-code
```

In this example, **RW** (read-write) is not a valid access mode, since a **CODE** section must allow execution. **X** (execute) should be added to the access mode. (e)

1780: non-int bitfields not allowed in packed structures

Bit-fields of type **char** or **short** are nonstandard. Depending on the compilation target, such bit-fields can result in faulty code when they occur in packed structures. Code such as this, therefore, should be avoided:

```
struct { ... unsigned short foo:11; ... }
                            __attribute__((packed)) struct1
```

int bit-fields are allowed in packed structures; therefore, in this example, the **unsigned short** should be replaced with an **int**. (e)

1781: absolute only supported with const int

__attribute__((“absolute”)) can only be used with variables of type **const int**. (e)

1782: *item* **is deprecated -** *message*

This warning is issued if a function, variable, or type has been marked by the user as deprecated (with **__attribute__ (deprecated)**), but is nonetheless referenced. *message* is an optional user-authored string. (w)

1783: Inlining function *function* **from file** *file*

This is an informational message that reports when the function *function* in the file *file* is being inlined during cross-module optimization. (i)

1784: Not inlining *function* **from exclude list**

This is an informational message only. This message appears if you have compiled with **-Xcmo-exclude-inline** and *function* is one of the functions to be excluded from cross-module optimization. (i)

1785:

(The error message with this number is no longer generated.)

1786: -Xpic not supported in specified target environment

An attempt was made to use the **-Xpic** option, but **-Xpic** does not work for the chosen targeted environment. **-Xpic** is intended for the VxWorks RTP application environment, and so should be used with the targeted environment **rtp** only. (f)

1787: Incorrectly promoting bitfield to unsigned int for backwards compatibility (for standards-compliant bitfield promotion please use -Xstrict-bitfield-promotions=1)

When a bit-field occurs in an expression where an **int** is expected, the compiler promotes the bit-field to a larger integral type. According to the ANSI standard, such a promotion should be value-preserving, but not necessarily sign-preserving. Specifically, an unsigned bit-field (of size < 32 bits) should be promoted to a signed **int** because the value fits into a signed **int**.

This warning indicates that the sign is being preserved (which was the case for pre-ANSI compilers). Use **-Xstrict-bitfield-promotions=1** to ensure that bit-fields are promoted according to standard.

1788: inline function *inline_function* **is never used**

A **static** function was declared as **inline** but was not called. This message appears if the **-Xlint** option is used.

This warning appears for C89 code only. Note that for C89, the function must be declared using **#pragma inline** and compiled using optimization (**-XO**). For more information, see also #tmt1501705396361__dt1517 on page 32, and the sections on inline pragmas in the User's Guide. (w)

1789: #pragma use_section *section section_type* **conflicts with earlier use_section for same variable**

A variable can only be assigned to one section. In this case **#pragma use_section** has been used more than once with the same variable. In such cases, the final **use_section** directive applies. (w)

1790: Incorrect struct/union/enum specifier in '*type1 struct_name***' (expected '***type2***')**

The wrong type has been applied to a **struct**, **union**, or **enum**. (e)

For example:

```
union U {       int x;     int y; }; struct U
                           u; /* ERROR */
```

1791: '/*' within comment. nested comment is not allowed

A "begin-comment" token (/*) follows another begin-comment token before an end-comment token (*/) is reached. Nesting of comments is not allowed; the second begin-comment token is ignored. (w)

1792: trying to assign 'ptr to volatile' to 'ptr'

An attempt has been made to assign a "pointer to **volatile**" to a pointer. (e)

It is illegal in C to assign a pointer to a variable that has a less restrictive type (with the exception that a **void *** pointer can be assigned a variable of any other pointer type). Qualifiers like **const** and **volatile** restrict the semantics of the objects they apply to. Thus it is permissible to assign a pointer to a "pointer to **volatile**," but not the other way around:

```
 volatile int * pVolatile; int *p; int main()
                                 volatile int * qVolatile = p; /* OK */       int * q = pVolat
ile;
                           /* ERROR */ }
```

If you really need to remove a **const** or **volatile** qualifier (and understand the implications of doing so), use an explicit cast.

1793: conflicting types for section section**:**

An attempt has been made to mix types of information in a single object-file section; for example, constant data (such as a string constant) into a section reserved for code or variables.

In this example, the compiler assumes from the first statement that the section **.mydata** is intended to be of the **DATA** section class, whereas the second statement assumes that **.mydata** will be a **CONST** section class:

```
__attribute__((section(".mydata"))) int var =
                           1; __attribute__((section(".mydata"))) const int const_var =
                           2;
```

1794: expensive optimizations disabled for function '*function***': size (***function_size***) > size specified with -Xopt-limit (-***Xopt_limit_size***)**

Example:

```
"application.cpp", line 491609: warning (etoa:1794): expensive optimizations disabled for func
tion '_foo': size (357930) > size specified with -Xopt-limit (10000)
```

1795: Semicolon found in assembly statement but neither -Xsemi-is-newline nor -Xsemi-is-comment has been specified

1796: __thread not supported in the specified target environment

1797: __thread variable used in initializer

1798: Unexpected flag for intrinsic function prototype. *diagnostic_message*

1799: function '*function***' is marked 'inline' but has no body (and so cannot be inlined)**

1800: [-Xwhole-program-optim: *diagnostic_message***]**

Examples:

```
info (dcc:1800): [-Xwhole-program-optim:Recompiling module a.c (from information stored at off
set 224 in a.o)]
info (dcc:1800): [-Xwhole-program-optim:Recompiling function foo]
```

1801: -Xwhole-program-optim: *fatal_error_message*

Examples:

```
fatal error (dcc:1801): -Xwhole-program-optim: in ModuleReader: cannot open file fileName for
reading
fatal error (dcc:1801): -Xwhole-program-optim: in ModuleWriter: cannot open file fileName for
writing
```

1802: -Xcmo-use and -Xcmo-gen are now deprecated. Please use -Xwhole-program-optim instead

1803: Addressing mode incompatible with -Xpic

1804: Recursion detected in function having __attribute__((always_inline))

1805: Recursion detected for function marked __attribute__((flatten))

1811: the alignment of auto variable *x* **exceeds frame alignment size** *n*

A variable has been declared with a size that exceeds alignment boundaries. For example, if the default alignment boundary is 16 bytes, the following code will cause this error to be issued:

```
 extern void goo(int*); int main(void) {  int
                          var __attribute__((aligned(32)));   /* exceeds alignment */
                           goo(&var);   printf("var 0x%x\n", var); }
```

1812: call to asm macro (*ASM_FUNC***) that is defined in a different compilation unit**

Using assembler macros across compilation units usually causes the linker to produce an error message, because assembler macros are handled local to their own compilation units. However, the **-Xwhole-program-optim** option provides cross-module optimization during link time. This error indicates that one compilation unit calls an assembler macro from a different compilation unit. Disabling the error (with **-ei1812**) or reducing the error to a warning level (**-ew1812**) will pull in the assembler macro like an externally defined function. (An assembler macro definition in one file may be used in many files.)

1813:Vectored *{fast}* **interrupts are not supported on this target**

This error message may apply to fast interrupts or to interrupts in general.

1814: Interrupt vector must be non-negative

1815: function *function* **not inlined as** *reason*

Examples:

```
info (dcc:1815): function test not inlined as callee has ASM macro
info (dcc:1815): function fibonacci not inlined as callee has asm(string)
```

1816: Unbalanced '(' or '[' or '{' in asm macro

1817: taking address of a cast expression is not legal in ANSI C, even if the cast is not size-changing

1819: Prototype mismatch while trying to inline call to '*function***' - please fix prototype, otherwise no further inlining will be performed for this file, due to inlining plan failure!!**

1820: optimizations will only be done at link-time, due to -Xwhole-program-optim. To optimize at both compile time and link time, use -Xwhole-program-optim=3.

1821: routine is both "fastcall" and "nofastcall" ("nofastcall" assumed)

1822: argument *arg_#* **in call to intrinsic** *function* **must be a compile-time constant**

Example:

```
line 30: error (etoa:1822): argument 2 in call to intrinsic _foo must be a compile-time consta
nt
```

1823: Invalid LTO group name (-Xlto-group='*lto_group_name***'). The name can only include alphanumeric characters and '_'.**

Example:

```
line 12: error (dcc:1823): Invalid LTO group name (-Xlto-group='Group-A'). The name can only i
nclude alphanumeric characters and '_'.
```

1824: explicit cast from '*type1***' to '***type2***' discards volatile qualifier**

You are trying to cast a type with a volatile qualifier to one that doesn't have one. Example:

```
volatile int x;

int* f(void)
{
   return (int*)&x;
}

$ dcc -tPPCEH:windiss -c vol.c
"vol.c", line 5: warning (dcc:1824): explicit cast from 'ptr-to-volatile int' to 'ptr-to-int'
discards volatile qualifier
```

1825: incomplete type is not allowed

1827: array element size (*array_size***) incompatible with requested element alignment (***base_size***) - elements will only be** *element_aligned***-byte aligned.**

Example:

```
array element size (6) incompatible with requested element alignment (32) - elements will only
 be 2-byte aligned
```

1828: SSP - in function '*function*', variable '*varname*' is still vulnerable after stack reordering because of its struct layout

1829: SSP - alloca() used in function '*function*'

1830: SSP enabled on '*function*' because *reason*

Examples:

```
info (dcc:1830): SSP enabled on 'foo' because has __attribute__((stack_protect))
info (dcc:1830): SSP enabled on 'bar' because -Xstack-protection-all
```

1831: Stack Smashing Protection is not supported on this target

2274: enumerated type mixed with another type

A variable of enumerated type is assigned a value of another type, such that the resulting value might be outside of the enumeration of the enumerated type.

```
//assume two different enumerated types e_Enum1 and e_Enum2
e_Enum1 enum1;
e_Enum2 enum2;
int noenum;

enum1=enum2; //2274: value of enum2 might not fall into enumeration of e_Enum1
enum2=enum1; //2274: value of enum1 might not fall into enumeration of e_Enum2
enum2=noenum; //2274: value of noenum might not fall into enumeration of e_Enum2
noenum=enum1; //OK
```

3. Messages Generated by etoa

The messages in this section are generated by **etoa**, which is the default frontend for C++, and which may be invoked for C with the **-Xc-new** option.

Messages generated by the default C compiler, **ctoa**, are listed in Compiler Message Format on page 1.

No further documentation is currently available for these messages. If a message if unclear, contact Customer Support.

The severity of some C++ diagnostics (information, warning, error, or fatal) varies according to the circumstances under which the message is generated.

- Ignore/Information (i) - The compiler is making a note about a non-serious issue with the source.
- Internal (in) - The compiler has detected an internal error that may have resulted from other errors reported earlier. If this appears to be unrelated to a previous error, please contact Customer Support.
- Fatal/Severe (f) - The compiler has detected a serious error in the source. The compilation attempt is terminated.
- Error (e) - The compiler has detected a serious error in the source. More errors may be reported after the first as a consequence of an attempt to recover from the error condition.
- Warning (w) - The compiler has detected a potential problem with the source and while not critical to continue code generation, this may result in undesired behavior.
- Deprecated (d) - This message is not generated, but has not been removed from in order to preserve the error numbers.

2273: suspicious extension of a 32-bit value when assigned to a 64-bit integral type (potential portability problem).

This error number is generated for previous versions.

4000: unknown error (i)

4001: last line of file ends without a newline (i)

4002: last line of file ends with a backslash (i)

4003: #include file "*xxxx***" includes itself** (f)

4004: out of memory (f)

4005: could not open source file "*xxxx***"** (d)

4006: comment unclosed at end of file (e)

4007: unrecognized token (i)(w)

4008: missing closing quote (i)(w)

4009: nested comment is not allowed (w)

4010: "#" not expected here (i)

4011: unrecognized preprocessing directive (w)

4012: parsing restarts here after previous syntax error (e)(w)

4013: expected a file name (e)(w)

4014: extra text after expected end of preprocessing directive (w)

4015: "*xxxx***" is not a file containing source text** (d)

4016: "*xxxx***" is not a valid source file name** (d)

4017: expected a "]" (i)

4018: expected a ")" (i)

4019: extra text after expected end of number (e)(w)

4020: identifier "*xxxx***" is undefined** (e)

4021: type qualifiers are meaningless in this declaration (w)

4022: invalid hexadecimal number (e)(w)

4023: integer constant is too large (e)(w)

4024: invalid octal digit (e)(w)

4025: quoted string should contain at least one character (i)

4026: too many characters in character constant (i)

4027: character value is out of range (w)

4028: expression must have a constant value (e)

4029: expected an expression (e)

4030: floating constant is out of range (w)

4031: expression must have integral type (e)(w)

Note that a **switch** statement must be an integer that fits into a **long** (e.g., for 32-bit targets it must fit into 32 bits). On 32-bit targets **long long** values will be truncated, while pointer values are not allowed by etoa and will produce this error.

See also Error 1122 in Messages Generated by ctoa on page 3.

4032: expression must have arithmetic type (i)

4033: expected a line number (e)

4034: invalid line number (e)

4035: #error directive: *xxxx* (f)

4036: the #if for this directive is missing (e)

4037: the #endif for this directive is missing (e)

4038: directive is not allowed -- an #else has already appeared (w)

4039: division by zero (e)(w)

4040: expected an identifier (e)(w)

4041: expression must have arithmetic or pointer type (e)(w)

4042: operand types are incompatible (*type* **and** *type***)** (e)(w)

4043: expression must have integral or pointer type (d)

4044: expression must have pointer type (i)

4045: #undef may not be used on this predefined name (w)

4046: this predefined name may not be redefined (i)

4047: incompatible redefinition of macro entity (i)

4048: cast between pointer-to-object and pointer-to-function (d)

4049: duplicate macro parameter name (e)

4050: "##" may not be first in a macro definition (e)

4051: "##" may not be last in a macro definition (e)

4052: expected a macro parameter name (e)

4053: expected a ":" (e)

4054: too few arguments in macro invocation (w)

4055: too many arguments in macro invocation (w)

4056: operand of sizeof may not be a function (e)(w)

4057: this operator is not allowed in a constant expression (e)

4058: this operator is not allowed in a preprocessing expression (e)

4059: function call is not allowed in a constant expression (i)

4060: this operator is not allowed in an integral constant expression (e)

4061: integer operation result is out of range (e)

4062: shift count is negative (i)

4063: shift count is too large (i)

4064: declaration does not declare anything (w)

4065: expected a ";" (e)

4066: enumeration value is out of "int" range (e)(w)

4067: expected a "}" (e)

4068: integer conversion resulted in a change of sign (i)

4069: integer conversion resulted in truncation (w)

4070: incomplete type is not allowed (e)

4071: operand of sizeof may not be a bit field (i)

4072: operand of "&" may not be a constant (d)

4073: operand of "&" in an initializer must be static (d)

4074: invalid operand of "&" (d)

4075: operand of "*" must be a pointer (w)

4076: argument to macro is empty (w)

4077: this declaration has no storage class or type specifier (e)

4078: a parameter declaration may not have an initializer (e)

4079: expected a type specifier (e)

4080: a storage class may not be specified here (e)

4081: more than one storage class may not be specified (e)

4082: storage class is not first (i)

4083: type qualifier specified more than once (e)(w)

4084: invalid combination of type specifiers (e)

4085: invalid storage class for a parameter (w)

4086: invalid storage class for a function (e)

4087: a type specifier may not be used here (e)

4088: array of functions is not allowed (e)

4089: array of void is not allowed (e)

4090: function returning function is not allowed (e)

4091: function returning array is not allowed (e)

4092: identifier-list parameters may only be used in a function definition (e)

4093: function type may not come from a typedef (e)

4094: the size of an array must be greater than zero (e)

4095: array is too large (e)

4096: a translation unit must contain at least one declaration (i)

4097: a function may not return a value of this type (e)

4098: an array may not have elements of this type (e)

4099: a declaration here must declare a parameter (e)(w)

4100: duplicate parameter name (e)

4101: "*xxxx***" has already been declared in the current scope** (e)

4102: forward declaration of enum type is nonstandard (w)

4103: class is too large (i)

4104: struct or union is too large (i)

4105: invalid size for bit field (e)

4106: invalid type for a bit field (e)

4107: zero-length bit field must be unnamed (e)(w)

4108: signed bit field of length 1 (w)

4109: expression must have (pointer-to-) function type

4110: expected either a definition or a tag name

4111: statement is unreachable

4112: expected "while"

4113: this use of a default argument is nonstandard

4114: *entity-kind "entity"* **was referenced but not defined**

4115: a continue statement may only be used within a loop

4116: a break statement may only be used within a loop or switch

4117: non-void *entity-kind "entity"* **should return a value**

4118: a void function may not return a value

4119: cast to type *"type"* **is not allowed**

4120: return value type does not match the function type

4121: a case label may only be used within a switch

4122: a default label may only be used within a switch

4123: case label value has already appeared in this switch

4124: default label has already appeared in this switch

4125: expected a "("

4126: expression must be an lvalue

4127: expected a statement

4128: loop is not reachable from preceding code

4129: a block-scope function may only have extern storage class

4130: expected a "{"

4131: expression must have pointer-to-class type

4132: expression must have pointer-to-struct-or-union type

4133: expected a member name

4134: expected a field name

4135: *entity-kind "entity"* **has no member "***xxxx***"**

4136: *entity-kind "entity"* **has no field "***xxxx***"**

4137: expression must be a modifiable lvalue

4138: taking the address of a register variable is not allowed

4139: taking the address of a bit field is not allowed

4140: too many arguments in function call

4141: unnamed prototyped parameters not allowed when body is present

4142: expression must have pointer-to-object type

4143: program too large or complicated to compile

4144: a value of type *"type"* **cannot be used to initialize an entity of type** *"type"*

4145: *entity-kind "entity"* **may not be initialized**

4146: too many initializer values

4147: declaration is incompatible with *entity-kind "entity"* (declared at line *xxxx*)

4148: *entity-kind "entity"* **has already been initialized**

4149: a global-scope declaration may not have this storage class

4150: a type name may not be redeclared as a parameter

4151: a typedef name may not be redeclared as a parameter

4152: conversion of nonzero integer to pointer

4153: expression must have class type

4154: expression must have struct or union type

4155: old-fashioned assignment operator

4156: old-fashioned initializer

4157: expression must be an integral constant expression

4158: expression must be an lvalue or a function designator

4159: declaration is incompatible with previous *"entity"* (declared at line *xxxx*)

4160: name conflicts with previously used external name "*xxxx***"**

4161: unrecognized #pragma

4162: expression must have arithmetic, pointer, or void type

4163: could not open temporary file "*xxxx***"**

4164: name of directory for temporary files is too long ("*xxxx***")**

4165: too few arguments in function call

4166: invalid floating constant

4167: argument of type *"type"* **is incompatible with parameter of type** *"type"*

4168: a function type is not allowed here

4169: expected a declaration

4170: pointer points outside of underlying object

4171: invalid type conversion

4172: external/internal linkage conflict with previous declaration

4173: floating-point value does not fit in required integral type

4174: expression has no effect

4175: subscript out of range

4176: constant string subscript out of range

4177: *entity-kind "entity"* **was declared but never referenced**

4178: "&" applied to an array has no effect

4179: right operand of "%" is zero

4180: argument is incompatible with formal parameter

4181: argument is incompatible with corresponding format string conversion

4182: could not open source file "*xxxx*" (no directories in search list)

4183: type of cast must be integral

4184: type of cast must be arithmetic or pointer

4185: dynamic initialization in unreachable code

4186: comparison of unsigned integer with zero is always true/false

4187: use of "=" where "==" may have been intended

4188: enumerated type mixed with another type

4189: error while writing *xxxx* **file**

4190: invalid intermediate language file

4191: type qualifier is meaningless on cast type

4192: unrecognized character escape sequence

4193: zero used for undefined preprocessing identifier

4194: expected an asm string

4195: an asm function must be prototyped

4196: an asm function may not have an ellipsis

4197: asm may only be used to declare a function

4198: an asm function may not have a storage class

4199: asm return value size does not match function return type

4200: asm parameter size does not match function parameter size

4201: expected a "%"

4202: invalid combination of asm control specifiers

4203: extra text after expected end of asm control line

4204: expected an asm control specifier

4205: this asm name is already defined

4206: invalid register name

4207: an asm parameter may not have void type

4208: expected an asm type specification

4209: invalid asm type specification

4210: invalid asm type width

4211: invalid asm constant

4212: an asm temporary may not have this type

4213: this parameter may not be referenced because it has no type

4214: the return value may not be referenced because its type is void

4215: invalid register specifier

4216: an expansion leaf must have at least one expansion line

4217: the return value may not be referenced because it has no type

4218: the return value may not have this asm type

4219: error while deleting file "*xxxx***"**

4220: integral value does not fit in required floating-point type

4221: floating-point value does not fit in required floating-point type

4222: floating-point operation result is out of range

4223: function declared implicitly

4224: the format string requires additional arguments

4225: the format string ends before this argument

4226: invalid format string conversion

4227: macro recursion

4228: trailing comma is nonstandard

4229: bit field cannot contain all values of the enumerated type

4230: nonstandard type for a bit field

4231: declaration is not visible outside of function

4232: old-fashioned typedef of "void" ignored

4233: left operand is not a struct or union containing this field

4234: pointer does not point to struct or union containing this field

4235: variable "*xxxx***" was declared with a never-completed type**

4236: controlling expression is constant

4237: selector expression is constant

4238: invalid specifier on a parameter

4239: invalid specifier outside a class declaration

4240: duplicate specifier in declaration

4241: a union is not allowed to have a base class

4242: multiple access control specifiers are not allowed

4243: class or struct definition is missing

4244: qualified name is not a member of class *"type"* **or its base classes**

4245: a nonstatic member reference must be relative to a specific object

4246: a nonstatic data member may not be defined outside its class

4247: *entity-kind "entity"* **has already been defined**

4248: pointer to reference is not allowed

4249: reference to reference is not allowed

4250: reference to void is not allowed

4251: array of reference is not allowed

4252: reference *entity-kind "entity"* **requires an initializer**

4253: expected a ","

4254: type name is not allowed

4255: type definition is not allowed

4256: invalid redeclaration of type name *"entity"* (declared at line *xxxx*)

4257: const *entity-kind "entity"* **requires an initializer**

4258: "this" may only be used inside a nonstatic member function

4259: constant value is not known

4260: explicit type is missing ("int" assumed)

4261: access control not specified ("*xxxx***" by default)**

4262: not a class or struct name

4263: duplicate base class name

4264: invalid base class

4265: *entity-kind "entity"* **is inaccessible**

4266: *"entity"* **is ambiguous**

4267: old-style parameter list (anachronism)

4268: declaration may not appear after executable statement in block

4269: implicit conversion to inaccessible base class *"type"* **is not allowed**

4270: name is not a member of a base class of "*xxxx***"**

4271: access adjustment in a "private" section is not allowed

4272: increasing an inherited member's access is not allowed

4273: restricting an inherited member's access is not allowed

4274: improperly terminated macro invocation

4276: name followed by "::" must be a class or namespace name

4277: invalid friend declaration

4278: a constructor or destructor may not return a value

4279: invalid destructor declaration

4280: declaration of a member with the same name as its class

4281: global-scope qualifier (leading "::") is not allowed

4282: the global scope has no "*xxxx***"**

4283: qualified name is not allowed

4284: NULL reference is not allowed

4285: initialization with "{...}" is not allowed for object of type *"type"*

4286: base class *"type"* **is ambiguous**

4287: derived class *"type"* **contains more than one instance of class** *"type"*

4288: cannot convert pointer to base class *"type"* **to pointer to derived class** *"type"* **-- base class is virtual**

4289: no instance of constructor *"entity"* **matches the argument list**

4290: copy constructor for class *"type"* **is ambiguous**

4291: no default constructor exists for class *"type"*

4292: "*xxxx***" is not a nonstatic data member or base class of class** *"type"*

4293: indirect nonvirtual base class is not allowed

4294: invalid union member -- class *"type"* **has a disallowed member function**

4295: cannot overload functions -- parameter types are too similar

4296: invalid use of non-lvalue array

4297: expected an operator

4298: inherited member is not allowed

4299: cannot determine which instance of *entity-kind "entity"* **is intended**

4300: a pointer to a bound function may only be used to call the function

4301: typedef name has already been declared (with same type)

4302: *entity-kind "entity"* **has already been defined**

4304: no instance of *entity-kind "entity"* **matches the argument list**

4305: type definition is not allowed in function return type declaration

4306: default argument not at end of parameter list

4307: redefinition of default argument

4308: more than one instance of *entity-kind "entity"* **matches the argument list:**

4309: more than one instance of constructor *"entity"* **matches the argument list:**

4310: default argument of type *"type"* **is incompatible with parameter of type** *"type"*

4311: cannot overload functions distinguished by return type alone

4312: no suitable user-defined conversion from *"type"* **to** *"type"* **exists**

4313: type qualifier is not allowed on this function

4314: only nonstatic member functions may be virtual

4315: the object has cv-qualifiers that are not compatible with the member function

4316: program too large to compile (too many virtual functions)

4317: return type is not identical to nor covariant with return type *"type"* **of overridden virtual function** *entity-kind "entity"*

4318: override of virtual *entity-kind "entity"* **is ambiguous**

4319: pure specifier ("= 0") allowed only on virtual functions

4320: badly-formed pure specifier (only "= 0" is allowed)

4321: data member initializer is not allowed

4322: object of abstract class type *"type"* **is not allowed:**

4323: function returning abstract class *"type"* **is not allowed:**

4324: duplicate friend declaration

4325: inline specifier allowed on function declarations only

4326: "inline" is not allowed

4327: invalid storage class for an inline function

4328: invalid storage class for a class member

4329: local class member *entity-kind "entity"* **requires a definition**

4330: *entity-kind "entity"* **is inaccessible**

4331: direct path to base class *"type"* **gives less access than indirect path**

4332: class *"type"* **has no copy constructor to copy a const object**

4333: defining an implicitly declared member function is not allowed

4334: class *"type"* **has no suitable copy constructor**

4335: linkage specification is not allowed

4336: unknown external linkage specification

4337: linkage specification is incompatible with previous *"entity"* (declared at line *xxxx*)

4338: more than one instance of overloaded function *"entity"* **has "C" linkage**

4339: class *"type"* **has more than one default constructor**

4340: value copied to temporary, reference to temporary used

4341: "operatorxxxx**" must be a member function**

4342: operator may not be a static member function

4343: no arguments allowed on user-defined conversion

4344: too many parameters for this operator function

4345: too few parameters for this operator function

4346: nonmember operator requires a parameter with class type

4347: default argument is not allowed

4348: more than one user-defined conversion from *"type"* **to** *"type"* **applies:**

4349: no operator "*xxxx***" matches these operands**

4350: more than one operator "*xxxx***" matches these operands:**

4351: first parameter of allocation function must be of type "size_t"

4352: allocation function requires "void *" return type

4353: deallocation function requires "void" return type

4354: first parameter of deallocation function must be of type "void *"

4355: second parameter of deallocation function must be of type "size_t"

4356: type must be an object type

4357: base class *"type"* **has already been initialized**

4358: base class name required -- *"type"* **assumed (anachronism)**

4359: *entity-kind "entity"* **has already been initialized**

4360: name of member or base class is missing

4361: assignment to "this" (anachronism)

4362: "overload" keyword used (anachronism)

4363: invalid anonymous union -- nonpublic member is not allowed

4364: invalid anonymous union -- member function is not allowed

4365: anonymous union at global or namespace scope must be declared static

4366: *entity-kind "entity"* **provides no initializer for:**

4367: implicitly generated constructor for class *"type"* **cannot initialize:**

4368: *entity-kind "entity"* **defines no constructor to initialize the following:**

4369: *entity-kind "entity"* **has an uninitialized const or reference member**

4370: *entity-kind "entity"* **has an uninitialized const field**

4371: class *"type"* **has no assignment operator to copy a const object**

4372: class *"type"* **has no suitable assignment operator**

4373: ambiguous assignment operator for class *"type"*

4374: const or volatile qualifier is not allowed

4375: declaration requires a typedef name

4376: unknown error

4377: "virtual" is not allowed

4378: "static" is not allowed

4379: cast of bound function to normal function pointer (anachronism)

4380: expression must have pointer-to-member type

4381: extra ";" ignored

4382: nonstandard member constant declaration (standard form is a static const integral member)

4383: a pointer to const may not be deleted

4384: no instance of overloaded *"entity"* **matches the argument list**

4385: operator delete() may not be overloaded

4386: no instance of *entity-kind "entity"* **matches the required type**

4387: delete array size expression used (anachronism)

4388: "operator->" for class *"type1"* **returns invalid type** *"type2"*

4389: a cast to abstract class *"type"* **is not allowed:**

4390: function "main" may not be called or have its address taken

4391: a new-initializer may not be specified for an array

4392: member function *"entity"* **may not be redeclared outside its class**

4393: pointer to incomplete class type is not allowed

4394: reference to local variable of enclosing function is not allowed

4395: single-argument function used for postfix "*xxxx***" (anachronism)**

4397: implicitly generated assignment operator cannot copy:

4398: cast to array type is nonstandard (treated as cast to *"type"***)**

4399: *entity-kind "entity"* **has an operator new***xxxx***() but no default operator delete***xxxx***()**

4400: *entity-kind "entity"* **has a default operator delete***xxxx***() but no operator new***xxxx***()**

4401: destructor for base class *"type"* **is not virtual**

4403: *entity-kind "entity"* **has already been declared**

4404: function "main" may not be declared inline

4405: member function with the same name as its class must be a constructor

4406: using nested *entity-kind "entity"* **(anachronism)**

4407: a destructor may not have parameters

4408: copy constructor for class *"type"* **may not have a parameter of type** *"type"*

4409: *entity-kind "entity"* **returns incomplete type** *"type"*

4410: protected *entity-kind "entity"* **is not accessible through a** *"type"* **pointer or object**

4411: a parameter is not allowed

4412: an "asm" declaration is not allowed here

4413: no suitable conversion function from *"type"* **to** *"type"* **exists**

4414: delete of pointer to incomplete class

4415: no suitable constructor exists to convert from *"type"* **to** *"type"*

4416: more than one constructor applies to convert from *"type"* **to** *"type"***:**

4417: more than one conversion function from *"type"* **to** *"type"* **applies:**

4418: more than one conversion function from *"type"* **to a built-in type applies:**

4424: a constructor or destructor may not have its address taken

4425: dollar sign ("$") used in identifier

4426: temporary used for initial value of reference to non-const (anachronism)

4427: qualified name is not allowed in member declaration

4428: enumerated type mixed with another type (anachronism)

4429: the size of an array in "new" must be non-negative

4430: returning reference to local temporary

4431: const qualifier dropped in initializing reference to non-const

4432: "enum" declaration is not allowed

4433: qualifiers dropped in binding reference of type *"type"* **to initializer of type** *"type"*

4434: a reference of type *"type"* **(not const-qualified) cannot be initialized with a value of type** *"type"*

4435: a pointer to function may not be deleted

4436: conversion function must be a nonstatic member function

4437: template declaration is not allowed here

4438: expected a "<"

4439: expected a ">"

4440: template parameter declaration is missing

4441: argument list for *entity-kind "entity* **is missing**

4442: too few arguments for *entity-kind "entity"*

4443: too many arguments for *entity-kind "entity"*

4444: template parameter for a function template must be a type

4445: *entity-kind "entity"* **is not used in declaring the parameter types of** *entity-kind "entity"*

4446: two nested types have the same name: *"entity"* **and** *"entity"* (declared at line *xxxx*) **(cfront compatibility)**

4447: global *"entity"* **was declared after nested** *"entity"* (declared at line *xxxx*) **(cfront compatibility)**

4449: more than one instance of *entity-kind "entity"* **matches the required type**

4450: the type "long long" is nonstandard

4451: omission of "*xxxx***" is nonstandard**

4452: return type may not be specified on a conversion function

4453: detected during:

4456: excessive recursion at instantiation of *entity-kind "entity"*

4457: "*xxxx***" is not a function or static data member**

4458: argument of type *"type"* **is incompatible with template parameter of type** *"type"*

4459: initialization requiring a temporary or conversion is not allowed

4460: declaration of "*xxxx*" hides function parameter

4461: initial value of reference to non-const must be an lvalue

4463: "template" is not allowed

4464: *"type"* **is not a class template**

4465: static data member may not be an anonymous union

4466: "main" is not a valid name for a function template

4467: invalid reference to *entity-kind "entity"* **(union/nonunion mismatch)**

4468: a template argument may not reference a local type

4469: tag kind of *xxxx* **is incompatible with declaration of** *entity-kind "entity"* (declared at line *xxxx*)

4470: the global scope has no tag named "*xxxx*"

4471: *entity-kind "entity"* **has no tag member named "*xxxx*"**

4472: member function typedef (allowed for cfront compatibility)

4473: *entity-kind "entity"* **may be used only in pointer-to-member declaration**

4474: unknown error

4475: a template argument may not reference a non-external entity

4476: name followed by "::~" must be a class name or a type name

4477: destructor name does not match name of class *"type"*

4478: type used as destructor name does not match type *"type"*

4479: *entity-kind "entity"* **redeclared "inline" after being called**

4480: destructor name does not match left operand of "->" or "."

4481: invalid storage class for a template declaration

4482: *entity-kind "entity"* **is an inaccessible type (allowed for cfront compatibility)**

4483: a return type is not allowed

4484: invalid explicit instantiation declaration

4485: *entity-kind "entity"* **is not an entity that can be instantiated**

4486: compiler generated *entity-kind "entity"* **cannot be explicitly instantiated**

4487: inline *entity-kind "entity"* **cannot be explicitly instantiated**

4489: *entity-kind "entity"* **cannot be instantiated -- no template definition was supplied**

4490: *entity-kind "entity"* **cannot be instantiated -- it has been explicitly specialized**

4491: class *"type"* **has no constructor**

4492: % must be used in a parameter without a default value in *entity-kind "entity"*

4493: no instance of *entity-kind "entity"* **matches the specified type**

4494: declaring a void parameter list with a typedef is nonstandard

4495: global *entity-kind "entity"* **used instead of** *entity-kind "entity"* **(cfront compatibility)**

4496: template parameter "*xxxx***" may not be redeclared in this scope**

4497: declaration of "*xxxx***" hides template parameter**

4498: template argument list must match the parameter list

4499: conversion function to convert from *"type1"* **to** *"type2"* **is not allowed**

4500: extra parameter of postfix "operator*xxxx***" must be of type "int"**

4501: an operator name must be declared as a function

4502: operator name is not allowed

4503: *entity-kind "entity"* **cannot be specialized in the current scope**

4504: nonstandard form for taking the address of a member function

4505: too few template parameters -- does not match previous declaration

4506: too many template parameters -- does not match previous declaration

4507: function template for operator delete(void *) is not allowed

4508: class template and template parameter may not have the same name

4510: a template argument may not reference an unnamed type

4511: enumerated type is not allowed

4512: type qualifier on a reference type is not allowed

4513: a value of type *"type"* **cannot be assigned to an entity of type** *"type"*

4514: comparison of unsigned integer with a negative constant is always true/false"

4515: cannot convert to incomplete class *"type"*

4516: const object requires an initializer

4517: object has an uninitialized const or reference member

4518: nonstandard preprocessing directive

4519: *entity-kind "entity"* **may not have a template argument list**

4520: initialization with "{...}" expected for aggregate object

4521: pointer-to-member selection class types are incompatible (*"type"* **and** *"type"***)**

4522: pointless friend declaration

4523: "." used in place of "::" to form a qualified name

4524: non-const function called for const object (anachronism)

4525: a dependent statement may not be a declaration

4526: a parameter may not have void type

4529: this operator is not allowed in a template argument expression

4530: try block requires at least one handler

4531: handler requires an exception declaration

4532: handler is masked by default handler

4533: handler is potentially masked by previous handler for type *"type"*

4534: use of a local type to specify an exception

4535: redundant type in exception specification

4536: exception specification is incompatible with that of previous *entity-kind "entity"* (declared at line *xxxx*)**:**

4537: previously specified: no exceptions will be thrown

4538: previously omitted: *"type"*

4539: previously specified but omitted here: *"type"*

4540: support for exception handling is disabled

4541: omission of exception specification is incompatible with previous *entity-kind "entity"* (declared at line *xxxx*)

4542: could not create instantiation request file "*xxxx***"**

4543: non-arithmetic operation not allowed in nontype template argument

4544: use of a local type to declare a nonlocal variable

4545: use of a local type to declare a function

4546: transfer of control bypasses initialization of:

4547: *entity-kind "entity"*

4548: transfer of control into an exception handler

4549: *entity-kind "entity"* **is used before its value is set**

4550: *entity-kind "entity"* **was set but never used**

4551: *entity-kind "entity"* **cannot be defined in the current scope**

4552: exception specification is not allowed

4553: external/internal linkage conflict for *entity-kind "entity"* (declared at line *xxxx*)

4554: *entity-kind "entity"* **will not be called for implicit or explicit conversions**

4555: tag kind of *xxx* **is incompatible with template parameter of type** *"type"*

4556: function template for operator new(size_t) is not allowed

4557: invalid access declaration -- inherited name "*xxxx***" is ambiguous**

4558: pointer to member of type *"type"* **is not allowed**

4559: ellipsis is not allowed in operator function parameter list

4560: *"entity***" is reserved for future use as a keyword**

4561: invalid macro definition:

4562: invalid macro undefinition:

4563: invalid preprocessor output file

4564: cannot open preprocessor output file

4565: IL file name must be specified if input is

4566: invalid IL output file

4567: cannot open IL output file

4568: invalid C output file

4569: cannot open C output file

4570: error in debug option argument

4571: invalid option:

4572: back end requires name of IL file

4573: could not open IL file

4574: invalid number:

4575: incorrect host CPU id

4576: invalid instantiation mode:

4577: missing include file directory name

4578: invalid error limit:

4579: invalid raw-listing output file

4580: cannot open raw-listing output file

4581: invalid cross-reference output file

4582: cannot open cross-reference output file

4583: invalid error output file

4584: cannot open error output file

4585: virtual function tables can only be suppressed when compiling C++

4586: anachronism option can be used only when compiling C++

4587: instantiation mode option can be used only when compiling C++

4588: automatic instantiation mode can be used only when compiling C++

4589: implicit template inclusion mode can be used only when compiling C++

4590: exception handling option can be used only when compiling C++

4591: strict ANSI mode is incompatible with K&R mode

4592: strict ANSI mode is incompatible with cfront mode

4593: missing source file name

4594: output files may not be specified when compiling several input files

4595: too many arguments on command line

4596: an output file was specified, but none is needed

4597: IL display requires name of IL file

4598: a template parameter may not have void type

4599: excessive recursive instantiation of *entity-kind "entity"* **due to instantiate-all mode**

4600: strict ANSI mode is incompatible with allowing anachronisms

4601: a throw expression may not have void type

4602: local instantiation mode is incompatible with automatic instantiation

4603: parameter of abstract class type *"type"* **is not allowed:**

4604: array of abstract class *"type"* **is not allowed:**

4605: floating-point template parameter is nonstandard

4606: this pragma must immediately precede a declaration

4607: this pragma must immediately precede a statement

4608: this pragma must immediately precede a declaration or statement

4609: this kind of pragma may not be used here

4611: overloaded virtual function *"entity"* **is only partially overridden in** *entity-kind "entity"*

An overloaded virtual function in the base class does not have a function with the corresponding signature in the derived class. The function in the base class is hidden in the derived class; this may result in unexpected behavior. For example:

```
    struct Base
{
        virtual void f(int) { /* base class definition*/ }
        virtual void f(double) { /* base class definition */ }
}
```

```
struct Derived : Base
{       void f(int) { /* derived class definition */ };
}
int main()
{
        Derived d;
        d.f(1.0); /* calls Derived::f(int) with the argument
                truncated from a double to an int, *not* Base::f(double)! */
}
```

Calling **f( )** on an instance of **Derived** with an argument of type double will nevertheless end up calling **Derived::f(int)** because that is the only overload that is visible.

To eliminate this warning either define functions in the derived class for all overloaded functions in the base class or explicitly introduce functions from the base class with the **using** keyword. For example:

```
    struct Base
{
        virtual void f(int) { /* base class definition */ }
        virtual void f(double) { /* base class definition */ }
}
struct Derived : Base {
        using Base::f;
        void f(int) { /* derived class definition */ };
}
```

4612: specific definition of inline template function must precede its first use

4613: invalid error tag:

4614: invalid error number:

4615: parameter type involves pointer to array of unknown bound

4616: parameter type involves reference to array of unknown bound

4617: pointer-to-member-function cast to pointer to function

4618: struct or union declares no named members

4619: nonstandard unnamed field

4620: nonstandard unnamed member

4621: a function type cannot be used as a template argument

4622: invalid precompiled header output file

4623: cannot open precompiled header output file

4624: "*xxxx*" is not a type name

4625: cannot open precompiled header input file

4626: precompiled header file "*xxxx*" is either invalid or not generated by this version of the compiler

4627: precompiled header file "*xxxx*" was not generated in this directory

4628: header files used to generate precompiled header file "*xxxx*" have changed

4629: the command line options do not match those used when precompiled header file "*xxxx*" was created

4630: the initial sequence of preprocessing directives is not compatible with those of precompiled header file "*xxxx*"

4631: unable to obtain mapped memory

4632: "*xxxx*": using precompiled header file "*xxxx*"

4633: "*xxxx*" creating precompiled header file "*xxxx*"

4634: memory usage conflict with precompiled header file "*xxxx*"

4635: invalid PCH memory size

4636: PCH options must appear first in the command line

4637: insufficient memory for PCH memory allocation

4638: precompiled header files may not be used when compiling several input files

4639: insufficient preallocated memory for generation of precompiled header file ("*xxxx*" bytes required)

4640: very large entity in program prevents generation of precompiled header file

4641: "*xxxx*" is not a valid directory

4642: cannot build temporary file name

4643: "restrict" is not allowed

4644: a pointer or reference to function type may not be qualified by "restrict"

4645: "*xxxx*" is an unrecognized __declspec attribute

4646: a calling convention modifier may not be specified here

4647: conflicting calling convention modifiers

4648: strict ANSI mode is incompatible with Microsoft mode

4649: cfront mode is incompatible with Microsoft mode

4650: calling convention specified here is ignored

4651: a calling convention may not be followed by a nested declarator

4652: calling convention is ignored for this type

4653: calling conventions may only be applied to function types

4654: declaration modifiers are incompatible with previous declaration

4655: the modifier "*xxxx*" is not allowed on this declaration

4656: transfer of control into a try block

4657: inline specification is incompatible with previous "*entity*" (declared at line *xxxx*)

4658: closing brace of template definition not found

4659: wchar_t keyword option can be used only when compiling C++

4660: invalid packing alignment value

4661: expected an integer constant

4662: call of pure virtual function

4663: invalid source file identifier string

4664: a class template cannot be defined in a friend declaration

4665: "asm" is not allowed

4666: "asm" must be used with a function definition

4667: "asm" function is nonstandard

4668: ellipsis with no explicit parameters is nonstandard

4669: "&..." is nonstandard

4670: invalid use of "&..."

4671: alternative token option can be used only when compiling C++

4672: temporary used for initial value of reference to const volatile (anachronism)

4673: a reference of type *"type"* **cannot be initialized with a value of type** *"type"*

4674: initial value of reference to const volatile must be an lvalue

4675: SVR4 C compatibility option can be used only when compiling ANSI C

4676: using out-of-scope declaration of *entity-kind "entity"* (declared at line *xxxx*)

4677: strict ANSI mode is incompatible with SVR4 C mode

4678: call of *entity-kind "entity"* declared at line *xxxx*) **cannot be inlined**

4679: *entity-kind "entity"* **cannot be inlined**

4680: invalid PCH directory:

4681: expected __except or __finally

4682: a __leave statement may only be used within a __try

4688: "*xxxx***" not found on pack alignment stack**

4689: empty pack alignment stack

4690: RTTI option can be used only when compiling C++

4691: *entity-kind "entity***", required for copy that was eliminated, is inaccessible**

4692: *entity-kind "entity***", required for copy that was eliminated, is not callable because reference parameter cannot be bound to rvalue**

4693: <typeinfo> must be included before typeid is used

4694: ”*xxxx*” cannot cast away const or other type qualifiers

4695: the type in a dynamic_cast must be a pointer or reference to a complete class type, or void *

4696: the operand of a pointer dynamic_cast must be a pointer to a complete class type

4697: the operand of a reference dynamic_cast must be an lvalue of a complete class type

4698: the operand of a runtime dynamic_cast must have a polymorphic class type

4699: bool option can be used only when compiling C++

4700: invalid storage class for condition declaration

4701: an array type is not allowed here

4702: expected an "="

4703: expected a declarator in condition declaration

4704: ”*xxxx*”, declared in condition, may not be redeclared in this scope

4705: default template arguments are not allowed for function templates

4706: expected a "," or ">"

4707: expected a template parameter list

4708: incrementing a bool value is deprecated

4709: bool type is not allowed

4710: offset of base class *"entity*” **within class** *"entity*” **is too large**

4711: expression must have bool type (or be convertible to bool)

4712: array new and delete option can be used only when compiling C++

4713: *entity-kind "entity*” **is not a variable name**

4714: __based modifier is not allowed here

4715: __based does not precede a pointer operator, __based ignored

4716: variable in __based modifier must have pointer type

4717: the type in a const_cast must be a pointer, reference, or pointer to member to an object type

4718: a const_cast can only adjust type qualifiers; it cannot change the underlying type

4719: mutable is not allowed

4720: redeclaration of *entity-kind "entity*” **is not allowed to alter its access**

4721: nonstandard format string conversion

4722: use of alternative token "<:" appears to be unintended

4723: use of alternative token "%:" appears to be unintended

4724: namespace definition is not allowed

4725: name must be a namespace name

4726: namespace alias definition is not allowed

4727: namespace-qualified name is required

4728: a namespace name is not allowed

4729: invalid combination of DLL attributes

4730: *entity-kind "entity"* **is not a class template**

4731: array with incomplete element type is nonstandard

4732: allocation operator may not be declared in a namespace

4733: deallocation operator may not be declared in a namespace

4734: *entity-kind "entity"* **conflicts with using-declaration of** *entity-kind "entity"* (declared at line *xxxx*)

4735: using-declaration of *entity-kind "entity"* **conflicts with** *entity-kind "entity"*

4736: namespaces option can be used only when compiling C++

4737: using-declaration ignored -- it refers to the current namespace

4738: a class-qualified name is required

4739: argument types are: (*xxxx***)**

4740: operand types are: *xxxx*

4742: *entity-kind "entity"* **has no actual member "***xxxx***"**

4743: global-scope qualifier (leading "::") on friend declaration is nonstandard

4744: incompatible memory attributes specified

4745: memory attribute ignored

4746: memory attribute may not be followed by a nested declarator

4747: memory attribute specified more than once

4748: calling convention specified more than once

4749: a type qualifier is not allowed

4750: *entity-kind "entity"* (declared at line *xxxx*) **was used before its template was declared**

4751: static and nonstatic member functions with same parameter types cannot be overloaded

4752: no prior declaration of *entity-kind "entity"*

4753: a template-id is not allowed

4754: a class-qualified name is not allowed

4755: *entity-kind "entity"* **may not be redeclared in the current scope**

4756: qualified name is not allowed in namespace member declaration

4757: *entity-kind "entity"* **is not a type name**

4758: explicit instantiation is not allowed in the current scope

4759: *entity-kind "entity"* **cannot be explicitly instantiated in the current scope**

4760: *entity-kind "entity"* **explicitly instantiated more than once**

4761: typename may only be used within a template

4762: special_subscript_cost option can be used only when compiling C++

4763: typename option can be used only when compiling C++

4764: implicit typename option can be used only when compiling C++

4765: nonstandard character at start of object-like macro definition

4766: exception specification for virtual *entity-kind "entity"* **is incompatible with that of overridden** *entity-kind "entity"*

4767: conversion from pointer to smaller integer

4768: exception specification for implicitly declared virtual *entity-kind "entity"* **is incompatible with that of overridden** *entity-kind "entity"*

4769: *"entity"***, implicitly called from** *entity-kind "entity"***, is ambiguous**

4770: option "explicit" can be used only when compiling C++

4771: "explicit" is not allowed

4772: declaration conflicts with "*xxxx***" (reserved class name)**

4773: only "()" is allowed as initializer for array *entity-kind "entity"*

4774: "virtual" is not allowed in a function template declaration

4775: invalid anonymous union -- class member template is not allowed

4776: template nesting depth does not match the previous declaration of *entity-kind "entity"*

4777: this declaration cannot have multiple "template <...>" clauses

4778: option to control the for-init scope can be used only when compiling C++

4779: "*xxxx***", declared in for-loop initialization, may not be redeclared in this scope**

4780: 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*)

4781: option to control warnings on for-init differences can be used only when compiling C++

4782: definition of virtual *entity-kind "entity"* **is required here**

4783: empty comment interpreted as token-pasting operator "##"

4784: a storage class is not allowed in a friend declaration

4785: template parameter list for *"entity"* **is not allowed in this declaration**

4786: *entity-kind "entity"* **is not a valid member class or function template**

4787: not a valid member class or function template declaration

4788: a template declaration containing a template parameter list may not be followed by an explicit specialization declaration

4789: explicit specialization of *entity-kind "entity"* **must precede the first use of** *entity-kind "entity"*

4790: explicit specialization is not allowed in the current scope

4791: partial specialization of *entity-kind "entity"* **is not allowed**

4792: *entity-kind "entity"* **is not an entity that can be explicitly specialized**

4793: explicit specialization of *entity-kind "entity"* **must precede its first use**

4794: template parameter *"type"* **may not be used in an elaborated type specifier**

4795: specializing *entity-kind "entity"* **requires "template<>" syntax**

4796: "template<>" syntax is required when declaring a member function template instance as a friend

4797: nonstandard "asm" declaration is not supported inside a template

4798: option "old_specializations" can be used only when compiling C++

4799: specializing *entity-kind "entity"* **without "template<>" syntax is nonstandard**

4800: this declaration may not have extern "C" linkage

4801: *"xxxx"* **is not a class or function template name in the current scope**

4802: specifying a default argument when redeclaring an unreferenced function template is nonstandard

4803: specifying a default argument when redeclaring an already referenced function template is not allowed

4804: cannot convert pointer to member of base class *"type"* **to pointer to member of derived class** *"type"* **-- base class is virtual**

4805: exception specification is incompatible with that of *entity-kind "entity"* (declared at line *xxxx*)**:'**

4806: omission of exception specification is incompatible with *entity-kind "entity"* (declared at line *xxxx*)

4807: unexpected end of default argument expression

4808: default-initialization of reference is not allowed

4809: uninitialized *entity-kind "entity"* **has a const member**

4810: uninitialized base class *"type"* **has a const member**

4811: const *entity-kind "entity"* **requires an initializer -- class** *"type"* **has no explicitly declared default constructor**

4812: const object requires an initializer -- class *"type"* **has no explicitly declared default constructor**

4813: option "implicit_extern_c_type_conversion" can be used only when compiling C++

4814: strict ANSI mode is incompatible with long preserving rules

4815: type qualifier on return type is meaningless

4816: in a function definition a type qualifier on a "void" return type is not allowed

4817: static data member declaration is not allowed in this class

4818: template instantiation resulted in an invalid function declaration

4819: "..." is not allowed

4820: option "extern_inline" can be used only when compiling C++

4821: extern inline *entity-kind "entity"* **was referenced but not defined**

4822: invalid destructor name for type *"type"*

4824: destructor reference is ambiguous -- both *entity-kind "entity"* **and** *entity-kind "entity"* **could be used**

4825: virtual inline *entity-kind "entity"* **was never defined**

4826: *entity-kind "entity"* **was never referenced**

4827: only one member of a union may be specified in a constructor initializer list

4828: support for "new[]" and "delete[]" is disabled

4829: "double" used for "long double" in generated C code

4830: *entity-kind "entity"* **has no corresponding operator delete"***xxxx***" (to be called if an exception is thrown during initialization of an allocated object)**

4831: support for placement delete is disabled

4832: no appropriate operator delete is visible

4833: pointer or reference to incomplete type is not allowed

4834: invalid partial specialization -- *entity-kind "entity"* **is already fully specialized**

4835: incompatible exception specifications

4836: returning reference to local variable

4837: omission of explicit type is nonstandard ("int" assumed)

4838: more than one partial specialization matches the template argument list of *entity-kind "entity"*

4840: a template argument list is not allowed in a declaration of a primary template

4841: partial specializations may not have default template arguments

4842: *entity-kind "entity"* **is not used in template argument list of** *entity-kind "entity"*

4844: the template argument list of the partial specialization includes a nontype argument whose type depends on a template parameter

4845: this partial specialization would have been used to instantiate *entity-kind "entity"*

4846: this partial specialization would have been made the instantiation of *entity-kind "entity"* **ambiguous**

4847: expression must have integral or enum type

4848: expression must have arithmetic or enum type

4849: expression must have arithmetic, enum, or pointer type

4850: type of cast must be integral or enum

4851: type of cast must be arithmetic, enum, or pointer

4852: expression must be a pointer to a complete object type

4853: a partial specialization of a member class template must be declared in the class of which it is a member

4854: a partial specialization nontype argument must be the name of a nontype parameter or a constant

4855: return type is not identical to return type *"type"* **of overridden virtual function** *entity-kind "entity"*

4856: option "guiding_decls" can be used only when compiling C++

4857: a partial specialization of a class template must be declared in the namespace of which it is a member

4858: *entity-kind "entity"* **is a pure virtual function**

4859: pure virtual *entity-kind "entity"* **has no overrider**

4860: __declspec attributes ignored

4861: invalid character in input line

4862: function returns incomplete type *"type"*

4863: effect of this "#pragma pack" directive is local to *entity-kind "entity"*

4864: *"xxxx"* **is not a template**

4865: a friend declaration may not declare a partial specialization

4866: exception specification ignored

4867: declaration of "size_t" does not match the expected type *"type"*

4868: space required between adjacent ">" delimiters of nested template argument lists (">>" is the right shift operator)

4869: could not set locale *"xxxx"* **to allow processing of multibyte characters**

4870: invalid multibyte character sequence

4871: template instantiation resulted in unexpected function type of *"type"* **1 (the meaning of a name may have changed since the template declaration -- the type of the template is** *"type"* **)**

4872: ambiguous guiding declaration -- more than one function template *"entity"* **matches type** *"type"*

4873: non-integral operation not allowed in nontype template argument

4874: option "embedded_c++" can be used only when compiling C++

4875: Embedded C++ does not support templates

4876: Embedded C++ does not support exception handling

4877: Embedded C++ does not support namespaces

4878: Embedded C++ does not support run-time type information

4879: Embedded C++ does not support the new cast syntax

4880: Embedded C++ does not support using-declarations

4881: Embedded C++ does not support "mutable"

4882: Embedded C++ does not support multiple or virtual inheritance

4883: invalid Microsoft version number:

4884: pointer-to-member representation "*xxxx***" has already been set for** *entity-kind "entity***"**

4885: *"type"* **cannot be used to designate constructor for** *"type"*

4886: invalid suffix on integral constant

4887: operand of __uuidof must have a class or enum type for which __declspec(uuid("...")) has been specified

4888: invalid GUID string in __declspec(uuid("..."))

4889: option "vla" can be used only when compiling C

4890: variable length array with unspecified bound is not allowed

4891: an explicit template argument list is not allowed on this declaration

4892: an entity with linkage cannot have a type involving a variable length array

4893: a variable length array cannot have static storage duration

4894: *entity-kind "entity***" is not a template**

4896: expected a template argument

4897: explicit function template argument lists are not supported yet in expression contexts

4898: nonmember operator requires a parameter with class or enum type

4899: option "enum_overloading" can be used only when compiling C++

4900: using-declaration of *entity-kind "entity***" is not allowed**

4901: qualifier of destructor name *"type"* **does not match type** *"type"*

4902: type qualifier ignored

4903: option "nonstd_qualifier_deduction" can be used only when compiling C++

4904: a function declared "dllimport" may not be defined

4905: incorrect property specification; correct form is __declspec(property(get=name1,put=name2))

4906: property has already been specified

4907: __declspec(property) is not allowed on this declaration

4908: member is declared with __declspec(property), but no "get" function was specified

4909: the __declspec(property) "get" function "*xxxx***" is missing**

4910: member is declared with __declspec(property), but no "put" function was specified

4911: the __declspec(property) "put" function "*xxxx***" is missing**

4912: ambiguous class member reference -- *entity-kind "entity***" used in preference to** *entity-kind "entity***"** (declared at line *xxxx*)

4913: missing or invalid segment name in __declspec(allocate("..."))

4914: __declspec(allocate) is not allowed on this declaration

4915: a segment name has already been specified

4916: cannot convert pointer to member of derived class *"type"* **to pointer to member of base class** *"type"* **-- base class is virtual**

4917: invalid directory for instantiation files:

4918: option "one_instantiation_per_object" can be used only when compiling C++

4919: invalid output file: "*xxxx***"**

4920: cannot open output file: "*xxxx***"**

4921: an instantiation information file name may not be specified when compiling several input files

4922: option "one_instantiation_per_object" may not be used when compiling several input files

4923: more than one command line option matches the abbreviation "--*xxxx***":**

4924: --*xxxx*

4925: a type qualifier cannot be applied to a function type

4926: cannot open definition list file: "*xxxx***"**

4927: late/early tiebreaker option can be used only when compiling C++

4928: incorrect use of va_start

4929: incorrect use of va_arg

4930: incorrect use of va_end

4931: pending instantiations option can be used only when compiling C++

4932: invalid directory for #import files:

4933: an import directory can be specified only in Microsoft mode

4934: a member with reference type is not allowed in a union

4935: "typedef" may not be specified here

4936: redeclaration of *entity-kind "entity"* **alters its access**

4937: a class or namespace qualified name is required

4938: return type "int" omitted in declaration of function "main"

4939: pointer-to-member representation "*xxxx***" is too restrictive for** *entity-kind "entity"*

4940: missing return statement at end of non-void *entity-kind "entity"*

4941: duplicate using-declaration of *"entity"* **ignored**

4942: enum bit-fields are always unsigned, but enum *"type"* **includes negative enumerator**

4943: option "class_name_injection" can be used only when compiling C++

4944: option "arg_dep_lookup" can be used only when compiling C++

4945: option "friend_injection" can be used only when compiling C++

4946: name following "template" must be a member template

4947: name following "template" must have a template argument list

4948: nonstandard local-class friend declaration -- no prior declaration in the enclosing scope

4949: specifying a default argument on this declaration is nonstandard

4950: option "nonstd_using_decl" can be used only when compiling C++

4951: return type of function "main" must be "int"

4952: a nontype template parameter may not have class type

4953: a default template argument cannot be specified on the declaration of a member of a class template

4954: a return statement is not allowed in a handler of a function try block of a constructor

4955: ordinary and extended designators cannot be combined in an initializer designation

4956: the second subscript must not be smaller than the first

4957: option "designators" can be used only when compiling C

4958: option "extended_designators" can be used only when compiling C

4959: declared size for bit field is larger than the size of the bit field type; truncated to "*xxxx***" bits**

4960: type used as constructor name does not match type *"type"*

4961: use of a type with no linkage to declare a variable with linkage

4962: use of a type with no linkage to declare a function

4963: return type may not be specified on a constructor

4964: return type may not be specified on a destructor

4965: incorrectly formed universal character name

4966: universal character name specifies an invalid character

4967: a universal character name cannot designate a character in the basic character set

4968: this universal character is not allowed in an identifier

4969: the identifier __VA_ARGS__ can only appear in the replacement lists of variadic macros

4970: the qualifier on this friend declaration is ignored

4971: array range designators cannot be applied to dynamic initializers

4972: property name cannot appear here

4973: "inline" used as a function qualifier is ignored

4974: option "compound_literals" can be used only when compiling C

4975: a variable-length array type is not allowed

4976: a compound literal is not allowed in an integral constant expression

4977: a compound literal of type *"type"* **is not allowed**

4978: a template friend declaration cannot be declared in a local class

4979: ambiguous "?" operation: second operand of type *"type"* **can be converted to third operand type***"type"***, and vice versa**

4980: call of an object of a class type without appropriate operator() or conversion functions to pointer-to-function type

4982: there is more than one way an object of type *"type"* **can be called for the argument list:**

4983: typedef name has already been declared (with similar type)

4984: operator new and operator delete cannot be given internal linkage

4985: storage class "mutable" is not allowed for anonymous unions

4986: invalid precompiled header file

4987: abstract class type *"type"* **is not allowed as catch type:**

4988: a qualified function type cannot be used to declare a nonmember function or a static member function

4989: a qualified function type cannot be used to declare a parameter

4990: cannot create a pointer or reference to qualified function type

4991: extra braces are nonstandard

4992: invalid macro definition:

4993: subtraction of pointer types *"type"* **and** *"type"* **is nonstandard**

4994: an empty template parameter list is not allowed in a template template parameter declaration

4995: expected "class"

4996: the "class" keyword must be used when declaring a template template parameter

4997: *entity-kind "entity"* **is hidden by** *"entity"* **-- virtual function override intended?**

4998: a qualified name is not allowed for a friend declaration that is a function definition

4999: *entity-kind "entity"* **is not compatible with** *entity-kind "entity"*

5000: a storage class may not be specified here

5001: class member designated by a using-declaration must be visible in a direct base class

5002: Sun mode is incompatible with Microsoft mode

5003: Sun mode is incompatible with cfront mode

5004: strict ANSI mode is incompatible with Sun mode

5005: Sun mode is only allowed when compiling C++

5006: a template template parameter cannot have the same name as one of its template parameters

5007: recursive instantiation of default argument

5008: a parameter of a template template parameter cannot depend on the type of another template parameter

5009: *entity-kind "entity"* **is not an entity that can be defined**

5010: destructor name must be qualified

5011: friend class name may not be introduced with "typename"

5012: a using-declaration may not name a constructor or destructor

5013: a qualified friend template declaration must refer to a specific previously declared template

5014: invalid specifier in class template declaration

5015: argument is incompatible with formal parameter

5016: option "dep_name" can be used only when compiling C++

5017: loop in sequence of "operator->" functions starting at class *"type"*

5018: *entity-kind "entity"* **has no member class "***xxxx***"**

5019: the global scope has no class named "*xxxx***"**

5020: recursive instantiation of template default argument

5021: access declarations and using-declarations cannot appear in unions

5022: *"entity"* **is not a class member**

5023: nonstandard member constant declaration is not allowed

5024: option "ignore_std" can be used only when compiling C++

5025: option "parse_templates" can be used only when compiling C++

5026: option "dep_name" cannot be used with "no_parse_templates"

5027: language modes specified are incompatible

5028: invalid redeclaration of nested class

5029: type containing an unknown-size array is not allowed

5030: a variable with static storage duration cannot be defined within an inline function

5031: an entity with internal linkage cannot be referenced within an inline function with external linkage

5032: argument type *"type"* **does not match this type-generic function macro**

5033: variable length array *"entity"*

5034: friend declaration cannot add default arguments to previous declaration

5035: *entity-kind "entity"* **cannot be declared in this scope**

5036: the reserved identifier "*xxxx***" may only be used inside a function**

5037: this universal character cannot begin an identifier

5038: expected a string literal

5039: unrecognized STDC pragma

5040: expected "ON", "OFF", or "DEFAULT"

5041: a STDC pragma may only appear between declarations in the global scope or before any statements or declarations in a block scope

5042: incorrect use of va_copy

5043: "*xxxx***" can only be used with floating-point types**

5044: complex type is not allowed

5045: invalid designator kind

5046: floating-point value cannot be represented exactly

5047: complex floating-point operation result is out of range

5048: conversion between real and imaginary yields zero

5049: an initializer cannot be specified for a flexible array member

5050: imaginary *= imaginary sets the left-hand operand to zero

5051: standard requires that *entity-kind "entity"* **be given a type by a subsequent declaration ("int" assumed)**

5052: a definition is required for inline *entity-kind "entity"*

5053: conversion from integer to smaller pointer

5054: a floating-point type must be included in the type specifier for a _Complex or _Imaginary type

5055: #pragma error: *xxxx*

5056: #pragma info: *xxxx*

5057: #pragma warning: *xxxx*

5058: expression must be a vector type or have a constant value

5059: use of obsolete feature: *xxxx*

5060: too few initializer values

5061: vector argument requires a prototype

5062: vector type specifier is not first

5063: operand of vec_step must be a vector type

5064: types cannot be declared in anonymous unions

5065: returning pointer to local variable

5066: returning pointer to local temporary

5067: option "export" can be used only when compiling C++

5068: option "export" cannot be used with "no_dep_name"

5069: option "export" cannot be used with "implicit_include"

5070: declaration of *entity-kind "entity*" **is incompatible with a declaration in another translation unit**

5071: the other declaration is *xxxx*"

5072: detected during compilation of secondary translation unit "*xxxx*"

5073: compilation of secondary translation unit "*xxxx*"

5074: a field declaration cannot have a type involving a variable length array

5075: declaration of *entity-kind "entity*" **had a different meaning during compilation of "***xxxx*"

5076: expected "template"

5077: "export" cannot be used on an explicit instantiation

5078: "export" cannot be used on this declaration

5079: a member of an unnamed namespace cannot be declared "export"

5080: a template cannot be declared "export" after it has been defined

5081: a declaration cannot have a label

5082: support for exported templates is disabled

5083: cannot open exported template file: "*xxxx*"

5084: *entity-kind "entity*" **already defined during compilation of "***xxxx*"

5085: *entity-kind "entity*" **already defined in another translation unit**

5086: a non-static local variable may not be used in a __based specification

5087: the option to list makefile dependencies may not be specified when compiling more than one translation unit

5088: the option to list included files may not be specified when compiling more than one translation unit

5089: the option to generate preprocessed output may not be specified when compiling more than one translation unit

5090: a field with the same name as its class cannot be declared in a class with a user-declared constructor

5091: "implicit_include" cannot be used when compiling more than one translation unit

5092: exported template file "*xxxx***" is corrupted**

5093: *entity-kind "entity***" cannot be instantiated -- it has been explicitly specialized in the translation unit containing the exported definition**

5094: object type is: *xxxx*

5095: the object has cv-qualifiers that are not compatible with the member *entity-kind "entity***"**

5096: no instance of *entity-kind "entity***" matches the argument list and object (the object has cv-qualifiers that prevent a match)**

5097: an attribute specifies a mode incompatible with *"type"*

5098: there is no type with the width specified

5099: invalid alignment value specified by attribute

5100: invalid attribute for *"type"*

5101: invalid attribute for *entity-kind "entity***"**

5102: invalid attribute for parameter

5103: attribute "*xxxx***" does not take arguments**

5104: attribute "*xxxx***" requires arguments**

5105: expected an attribute name

5106: there is no attribute "*xxxx***"**

5107: attributes may not appear here

5108: invalid argument to attribute "*xxxx***"**

5109: the "packed" attribute is ignored in a typedef

5110: in "goto *expr", expr must have type "void *"

5111: "goto *expr" is nonstandard

5112: taking the address of a label is nonstandard

5113: file name specified more than once:

5114: #warning directive: "*xxxx***"**

5115: attribute "*xxxx***" is only allowed in a function definition**

5116: the "transparent_union" attribute only applies to unions, and *"type"* **is not a union**

5117: the "transparent_union" attribute is ignored on incomplete types

5118: *"type"* **cannot be transparent because** *entity-kind "entity"* **does not have the same size as the union**

5119: *"type"* **cannot be transparent because it has a field of type** *"type"* **which is not the same size as the union**

5120: only parameters can be transparent

5121: the "*xxxx***" attribute does not apply to local variables**

5122: attributes are not permitted in a function definition

5123: declarations of local labels should only appear at the start of statement expressions

5124: the second constant in a case range must be larger than the first

5125: an asm name is not permitted in a function definition

5126: an asm name is ignored "*xxxx***"**

5128: modifier letter "*xxxx***" ignored in asm operand**

5129: unknown asm constraint modifier "*xxxx***"'**

5130: unknown asm constraint letter '"*xxxx***"'**

5131: asm operand has no constraint letter

5132: an asm output operand must have one of the '=' or '+' modifiers

5133: an asm input operand may not have the '=' or '+' modifiers

5134: too many operands to asm statement (maximum is 30; '+' modifier adds an implicit operand)

5135: too many colons in asm statement

5136: register "*xxxx***" used more than once**

5137: register "*xxxx***" is both used and clobbered**

5138: register "*xxxx***" clobbered more than once**

5139: register "*xxxx***" has a fixed purpose and may not be used in an asm statement**

5140: register "*xxxx***" has a fixed purpose and may not be clobbered in an asm statement**

5141: an empty clobbers list must be omitted entirely

5142: expected an asm operand

5143: expected a register to clobber

5144: "format" attribute applied to *entity-kind "entity"* **which does not have variable arguments**

5145: first substitution argument is not the first variable argument

5146: format argument index is greater than number of parameters

5147: format argument does not have string type

5148: the "template" keyword used for syntactic disambiguation may only be used within a template

5149: a debug option must be specified on the command-line for the db_opt pragma to be used

5150: more than one preinclude option specified

5151: attribute does not apply to non-function type *"type"*

5152: arithmetic on pointer to void or function type

5153: storage class must be auto or register

5154: *"type"* **would have been promoted to** *"type"* **when passed through the ellipsis parameter; use the latter type instead**

5155: "*xxxx***" is not a base class member**

5156: __super cannot appear after "::"

5157: __super may only be used in a class scope

5158: __super must followed by "::"

5159: [*xxxx* **instantiation contexts not shown]**

5160: mangled name is too long

5161: declaration aliased to unknown entity "*xxxx***"**

5162: declaration does not match its alias *entity-kind "entity***"**

5163: entity declared as alias cannot have definition

5164: variable-length array field type will be treated as zero-length array field type

5165: nonstandard cast on lvalue ignored

5166: unrecognized flag name

5167: void return type cannot be qualified

5168: the auto specifier is ignored here (invalid in standard C/C++)

5169: a reduction in alignment without the "packed" attribute is ignored

5170: a member template corresponding to *"entity***" is declared as a template of a different kind in another translation unit**

5171: excess initializers are ignored

5172: va_start should only appear in a function with an ellipsis parameter

5173: the "short_enums" option is only valid in GNU C mode

5174: invalid export information file "*xxxx***" at line number "***xxxx***"**

5175: statement expressions are only allowed in block scope

5176: from translation unit

5177: an asm name is ignored on a non-register automatic variable

5178: inline function also declared as an alias; definition ignored

5179: cannot initialize __ev64_opaque__ from a brace enclosed list (first cast to a specific ev64 type)

5180: priority out of range

5181: improper object type or scope, attribute ignored

5182: inline *entity-kind "entity"* **was declared but never referenced**

5183: unrecognized UPC pragma

5184: shared block size does not match one previously specified

5185: bracketed expression is assumed to be a block size specification rather than an array dimension

5186: the block size of a shared array must be greater than zero

5187: multiple block sizes not allowed

5188: strict or relaxed requires shared

5189: THREADS not allowed in this context

5190: block size specified exceeds the maximum value of "*xxxx***"**

5191: function returning shared is not allowed

5192: only arrays of a shared type can be dimensioned to a multiple of THREADS

5193: one dimension of an array of a shared type must be a multiple of THREADS when the number of threads is nonconstant

5194: shared type inside a struct or union is not allowed

5195: parameters may not have shared types

5196: a dynamic THREADS dimension requires a definite block size

5197: shared variables must be static or extern

5198: argument of upc_blocksizeof is a pointer to a shared type (not shared type itself)

5199: affinity expression ignored in nested upc_forall

5200: branching into or out of a upc_forall loop is not allowed

5201: affinity expression must have a shared type or point to a shared type

5202: affinity has shared type (not pointer to shared)

5203: shared void* types can only be compared for equality

5204: UPC mode is incompatible with C++ and K&R modes

5205: null (zero) character in input line ignored

5206: null (zero) character in string or character constant

5207: null (zero) character in header name

5208: declaration in for-initializer hides a declaration in the surrounding scope

5209: the hidden declaration is at line "*xxxx***"**

5210: the prototype declaration of *entity-kind "entity***"** (declared at line *xxxx*) **is ignored after this unprototyped redeclaration**

5211: attribute ignored on typedef of class or enum types

5212: *entity-kind "entity***" must have external C linkage**

5213: variable declaration hides declaration in for-initializer

5214: typedef "*xxxx***" may not be used in an elaborated type specifier**

5215: call of zero constant ignored

5216: parameter "*xxxx***" may not be redeclared in a catch clause of function try block**

5217: the initial explicit specialization of *entity-kind "entity***" must be declared in the namespace containing the template**

5218: "cc" clobber ignored

5219: "template" must be followed by an identifier

5220: MYTHREAD not allowed in this context

5221: layout qualifier cannot qualify pointer to shared

5222: layout qualifier cannot qualify an incomplete array

5223: declaration of "*xxxx***" hides handler parameter**

5224: nonstandard cast to array type ignored

5225: this pragma cannot be used in a _Pragma operator (a #pragma directive must be used)

5226: field uses tail padding of a base class

5227: GNU C++ compilers may use bit field padding

5228: use of *entity-kind "entity***" is deprecated:** *xxxx*

5229: an asm name is not allowed on a nonstatic member declaration

5230: unrecognized format function type "*xxxx***" ignored**

5231: base class *"entity"* **uses tail padding of base class** *"entity"*

5232: the "init_priority" attribute can only be used for definitions of static data members and namespace scope variables of class types

5233: requested initialization priority is reserved for internal use

5234: this anonymous union/struct field is hidden by *entity-kind "entity***"** (declared at line *xxxx*)

5235: invalid error number

5236: invalid error tag

5237: expected an error number or error tag

5238: size of class is affected by tail padding

5239: labels can be referenced only in function definitions

5240: transfer of control into a statement expression is not allowed

5241: transfer of control out of a statement expression is not allowed

5242: this statement is not allowed inside of a statement expression

5243: a non-POD class definition is not allowed inside of a statement expression

5244: destructible entities are not allowed inside of a statement expression

5245: a dynamically-initialized local static variable is not allowed inside of a statement expression

5246: a variable-length array is not allowed inside of a statement expression

5247: a statement expression is not allowed inside of a default argument

5248: nonstandard conversion between pointer to function and pointer to data

5249: interface types cannot have virtual base classes

5250: interface types cannot specify "private" or "protected"

5251: interface types can only derive from other interface types

5252: *"type"* **is an interface type**

5253: interface types cannot have typedef members

5254: interface types cannot have user-declared constructors or destructors

5255: interface types cannot have user-declared member operators

5256: interface types cannot be declared in functions

5257: cannot declare interface templates

5258: interface types cannot have data members

5259: interface types cannot contain friend declarations

5260: interface types cannot have nested classes

5261: interface types cannot be nested class types

5262: interface types cannot have member templates

5263: interface types cannot have static member functions

5264: this pragma cannot be used in a __pragma operator (a #pragma directive must be used)

5265: qualifier must be base class of *"type"*

5266: declaration must correspond to a pure virtual member function in the indicated base class

5267: integer overflow in internal computation due to size or complexity of *"type"*

5268: integer overflow in internal computation

5269: __w64 can only be specified on int, long, and pointer types

5270: potentially narrowing conversion when compiled in an environment where int, long, or pointer types are 64 bits wide

5271: current value of pragma pack is "*xxxx***"**

5272: arguments for pragma pack(show) are ignored

5273: invalid alignment specifier value

5274: expected an integer literal

5275: earlier __declspec(align(...)) ignored

5276: expected an argument value for the "*xxxx***" attribute parameter**

5277: invalid argument value for the "*xxxx***" attribute parameter**

5278: expected a boolean value for the "*xxxx***" attribute parameter**

5279: a positional argument cannot follow a named argument in an attribute

5280: attribute "*xxxx***" 1 has no parameter named "***xxxx***"**

5281: expected an argument list for the "*xxxx***" attribute**

5282: expected a "," or "]"

5283: attribute argument "*xxxx***" has already been given a value**

5284: a value cannot be assigned to the "*xxxx***" attribute**

5285: a throw expression may not have pointer-to-incomplete type

5286: alignment-of operator applied to incomplete type

5287: "*xxxx***" may only be used as a standalone attribute**

5288: "*xxxx***" attribute cannot be used here**

5289: unrecognized attribute "*xxxx***"**

5290: attributes are not allowed here

5291: invalid argument value for the "*xxxx***" attribute parameter**

5292: too many attribute arguments

5293: conversion from inaccessible base class "*type***" is not allowed**

5294: option "export" requires distinct template signatures

5295: narrow and wide string literals cannot be concatenated

5296: GNU layout bug not emulated because it places virtual base *"entity"* **outside** *"entity"* **object boundaries**

5297: virtual base *"entity"* **placed outside** *"entity"* **object boundaries**

5298: nonstandard qualified name in namespace member declaration

5299: reduction in alignment ignored

5300: const qualifier ignored

5301: return statement in function marked with "noreturn"

5302: invalid GNU asm qualifiers

5303: non-POD class type passed through ellipsis

5304: a non-POD class type cannot be fetched by va_arg

5305: the 'u' or 'U' suffix must appear before the 'l' or 'L' suffix in a fixed-point literal

5306: option "fixed_point" can be used only when compiling C

5307: integer operand may cause fixed-point overflow

5308: fixed-point constant is out of range

5309: fixed-point value cannot be represented exactly

5310: constant is too large for long long; given unsigned long long type (nonstandard)

5311: layout qualifier cannot qualify pointer to shared void

5312: duplicate THREADS in multidimensional array type

5313: a strong using-directive may only appear in a namespace scope

5314: *entity-kind "entity"* **declares a non-template function -- add <> to refer to a template instance**

5315: operation may cause fixed-point overflow

5316: expression must have integral, enum, or fixed-point type

5317: expression must have integral or fixed-point type

5318: function declared with "noreturn" does return

5319: asm name ignored because it conflicts with a previous declaration

5320: class member typedef may not be redeclared

5321: taking the address of a temporary

5322: attributes are ignored on a class declaration that is not also a definition

5323: fixed-point value implicitly converted to floating-point type

5324: fixed-point types have no classification

5325: a template parameter may not have fixed-point type

5326: hexadecimal floating-point constants are not allowed

5327: option "named_address_spaces" can be used only when compiling C

5328: floating-point value does not fit in required fixed-point type

5329: value cannot be converted to fixed-point value exactly

5330: fixed-point conversion resulted in a change of sign

5331: integer value does not fit in required fixed-point type

5332: fixed-point operation result is out of range

5333: multiple named address spaces

5334: variable with automatic storage duration cannot be stored in a named address space

5335: type cannot be qualified with named address space

5336: function type cannot be qualified with named address space

5337: field type cannot be qualified with named address space

5338: fixed-point value does not fit in required floating-point type

5339: fixed-point value does not fit in required integer type

5340: value does not fit in required fixed-point type

5341: option "named_registers" can be used only when compiling C

5342: a named-register storage class is not allowed here

5343: *entity-kind "entity"* (declared at line *xxxx*) **redeclared with incompatible named-register storage class**

5344: named-register storage class cannot be specified for aliased variable

5345: named-register storage specifier is already in use

5346: option "embedded_c" cannot be combined with options to control individual Embedded C features

5347: invalid EDG_BASE directory:

5348: cannot open predefined macro file: "*xxxx***"**

5349: invalid predefined macro entry at line *xxxx***:** *xxxx*

5350: invalid macro mode name "*xxxx***"**

5351: incompatible redefinition of predefined macro "*xxxx***"**

5352: redeclaration of *entity-kind "entity"* (declared at line *xxxx*) **is missing a named-register storage class**

5353: named register is too small for the type of the variable

5354: arrays cannot be declared with named-register storage class

5355: const_cast to enum type is nonstandard

5356: option "embedded_c" can be used only when compiling C

5357: a named address space qualifier is not allowed here

5358: an empty initializer is invalid for an array with unspecified bound

5359: function returns incomplete class type *"type"*

5360: *entity-kind "entity"* **has already been initialized; the out-of-class initializer will be ignored**

5361: declaration hides *entity-kind "entity"*

5362: a parameter cannot be allocated in a named address space

5363: invalid suffix on fixed-point or floating-point constant

5364: a register variable cannot be allocated in a named address space

5365: expected "SAT" or "DEFAULT"

5366: *entity-kind "entity"* **has no corresponding member operator delete***xxxx* **(to be called if an exception is thrown during initialization of an allocated object)**

5367: a thread-local variable cannot be declared with "dllimport" or "dllexport"

5368: a function return type cannot be qualified with a named address space

5369: an initializer cannot be specified for a flexible array member whose elements have a nontrivial destructor

5370: an initializer cannot be specified for an indirect flexible array member

5371: invalid GNU version number:

5372: variable attributes appearing after a parenthesized initializer are ignored

5373: the result of this cast cannot be used as an lvalue

5374: negation of an unsigned fixed-point value

5375: this operator is not allowed at this point; use parentheses

5376: flexible array member initializer must be constant

5377: register names can only be used for register variables

5378: named-register variables cannot have void type

5379: __declspec modifiers not valid for this declaration

5380: parameters cannot have link scope specifiers

5381: multiple link scope specifiers

5382: link scope specifiers can only appear on functions and variables with external linkage

5383: a redeclaration cannot weaken a link scope

5384: link scope specifier not allowed on this declaration

5385: nonstandard qualified name in global scope declaration

5386: implicit conversion of a 64-bit integral type to a smaller integral type (potential portability problem)

5387: explicit conversion of a 64-bit integral type to a smaller integral type (potential portability problem)

5388: conversion from pointer to same-sized integral type (potential portability problem)

5389: the "sun_linker_scope" option is only valid in Sun mode

5390: friend specifier is not allowed in a class definition; friend specifier is ignored

5391: only static and extern variables can use thread-local storage

5392: multiple thread-local storage specifiers

5393: virtual *entity-kind "entity"* **was not defined (and cannot be defined elsewhere because it is a member of an unnamed namespace)**

5394: carriage return character in source line outside of comment or character/string literal

5395: expression must have fixed-point type

5396: invalid use of access specifier is ignored

5397: pointer converted to bool

5398: pointer-to-member converted to bool

5399: storage specifier ignored

5400: dllexport and dllimport are ignored on class templates

5401: base class dllexport/dllimport specification differs from that of the derived class

5402: redeclaration cannot add dllexport/dllimport to *"entity"* (declared at line *xxxx*)

5403: dllexport/dllimport conflict with *"entity"***; dllexport assumed**

5404: cannot define dllimport entity

5405: dllexport/dllimport requires external linkage

5406: a member of a class declared with dllexport/dllimport cannot itself be declared with such a specifier

5407: field of class type without a DLL interface used in a class with a DLL interface

5408: parenthesized member declaration is nonstandard

5409: white space between backslash and newline in line splice ignored

5410: dllexport/dllimport conflict with *"entity"***; dllimport/dllexport dropped**

5411: invalid member for anonymous member class -- class *"type"* **has a disallowed member function**

5412: nonstandard reinterpret_cast

5413: positional format specifier cannot be zero

5414: a local class cannot reference a variable-length array type from an enclosing function

5415: member *entity-kind "entity"* **already has an explicit dllexport/dllimport specifier**

5416: a variable-length array is not allowed in a function return type

5417: variable-length array type is not allowed in pointer to member of type *"type"*

5418: the result of a statement expression cannot have a type involving a variable-length array

5419: support for trigraphs is disabled

5420: the "xxxx" attribute can only appear on functions and variables with external linkage

5421: strict mode is incompatible with treating namespace std as an alias for the global namespace

5427: invalid symbolic operand name "xxxx"

5428: a symbolic match constraint must refer to one of the first ten operands

5429: use of __if_exists is not supported in this context

5430: __if_exists block not closed in the same scope in which it was opened

5431: thread-local variable cannot be dynamically initialized

5432: conversion drops "__unaligned" qualifier

5433: some enumerator values cannot be represented by the integral type underlying the enum type

5434: default argument is not allowed on a friend class template declaration

5435: multicharacter character literal (potential portability problem)

5436: expected a class, struct, or union type

5437: second operand of offsetof must be a field

5438: second operand of offsetof may not be a bit field

5439: cannot apply offsetof to a member of a virtual base

5440: offsetof applied to non-POD types is nonstandard

5441: default arguments are not allowed on a friend declaration of a member function

5442: default arguments are not allowed on friend declarations that are not definitions

5443: redeclaration of entity-kind "entity" previously declared as a friend with default arguments is not allowed

5444: invalid qualifier for "type" (a derived class is not allowed here)

5445: invalid qualifier for definition of class "type"

5446: no prior push_macro for "xxxx"

5447: wide string literal not allowed

5448: "xxxx" is only allowed in C++

5449: "xxxx" is only allowed in C

5450: __ptr32 and __ptr64 must follow a "*"

5451: __ptr32 and __ptr64 cannot both apply

5452: template argument list of ”xxxx” must match the parameter list

5453: an incomplete class type is not allowed

5454: complex integral types are not supported

5455: __real and __imag can only be applied to complex values

5456: __real/__imag applied to real value

5457: entity-kind "entity” was declared "deprecated (”xxxx”)"

5458: invalid redefinition of entity-kind "entity”

5459: dllimport/dllexport applied to a member of an unnamed namespace

5460: __thiscall can only appear on nonstatic member function declarations

5461: __thiscall not allowed on function with ellipsis parameter

5462: explicit specialization of entity-kind ”entity”must precede its first use (at line xxxx)

5463: a sealed class type cannot be used as a base class

5464: duplicate class modifier

5465: a member function cannot have both the "abstract" and "sealed" modifiers

5466: a sealed member cannot be pure virtual

5467: nonvirtual function cannot be declared with "abstract" or "sealed modifier"

5468: member function declared with "override" modifier does not override a base class member

5469: cannot override sealed entity-kind "entity”

5470: entity-kind "entity” was declared with the class modifier "abstract"

5515: catastrophic error

5516: Catastrophic error

5517: command-line error

5518: Command-line error

5519: internal error

5520: Internal error

5521: -D

5522:

5523: Error limit reached.

5524: Internal error loop

5525: Loop in catastrophic error processing.

5526: generated C output

5527: temporary

5528: preprocessing output

5529: raw listing

5530: cross-reference

5531: intermediate language (1)

5532: intermediate language (2)

5533: intermediate language (3)

5534: intermediate language (4)

5535: intermediate language (5)

5536: intermediate language (6)

5537: intermediate language (7)

5538: intermediate language (8)

5539: intermediate language (9)

5540: PCH

5541: template information file

5542: exported template file

5543: instantiation request file

5544: definition list file

5545: missing cannot-redefine flag

5546: missing mode after ','

5547: missing macro name

5548: invalid cannot-redefine value

5549: duplicate function modifier

5550: invalid character for char16_t literal

5551: __LPREFIX cannot be applied to char16_t or char32_t literals

5552: unrecognized calling convention xxxx, must be one of:

5553: xxxx

5554: option to control char16_t/char32_t literals can be used only when compiling C

5555: attribute “xxxx” not allowed on parameter declarations

5556: underlying type of enum type must be an integral type other than bool

5557: some enumerator constants cannot be represented by "type”

5558: “xxxx” not allowed in current mode

5559: type traits helpers option can be used only when compiling C++

5560: attribute "sentinel" requires an ellipsis parameter

5561: argument must be a constant null pointer value

5562: insufficient number of arguments for sentinel value

5563: sentinel argument must correspond to an ellipsis parameter

5564: __declspec(implementation_key(...) can appear only between #pragma start_map_region and #pragma stop_map_region

5565: #pragma start_map_region already active: pragma ignored

5566: no #pragma start_map_region is currently active: pragma ignored

5567: entity-kind “entity” cannot be used to name a destructor (a type name is required)

5568: nonstandard empty wide character literal treated as L'\\0'

5569: "typename" may not be specified here

5570: a non-placement operator delete must be visible in a class with a virtual destructor

5571: name linkage conflicts with previous declaration of entity-kind "entity”

5572: alias creates cycle of aliased entities

5573: subscript must be constant

5574: a variable with static storage duration allocated in a specific register cannot be declared with an initializer

5575: a variable allocated in a specific register must have POD type

5576: predefined meaning of “entity” discarded

5577: the “xxxx” attribute can only appear on functions and variables with internal linkage

5578: designator may not specify a non-POD subobject

5579: enum qualified name is nonstandard

5580: anonymous union qualifier is nonstandard

5581: anonymous union qualifier is ignored

5582: __declspec ignored (no variable or function declared)

5583: __declspec(xxxx) ignored (it has no meaning for a C struct)

5584: specifiers after comma between declarations are nonstandard

5585: nonstandard specifier ignored

5586: attributes are ignored on an enum declaration that is not also a definition

5587: declaring a reference with "mutable" is nonstandard

5588: a condition declaration for an array is always true

5589: static assertion failed with “xxxx”

5590: visibility attribute ignored because it conflicts with a previous declaration

5591: field name resolves to more than one offset -- see "entity1” and "entity2”

5592: “xxxx” is not a field name

5593: case label value has already appeared in this switch at line xxxx

5594: a member function cannot have internal linkage

5595: declaration hides built-in entity-kind “entity”

5596: declaration overloads built-in entity-kind “entity”

5597: the option to list macro definitions may not be specified when compiling more than one translation unit

5598: unexpected parenthesis after declaration of entity-kind “entity” (malformed parameter list or invalid initializer?)

5599: parentheses around a string initializer are nonstandard

5600: __interface

5601: a variable declared with an auto type specifier cannot appear in its own initializer

5602: cannot deduce "auto" type

5603: initialization with "{...}" is not allowed for "auto" type

5604: "auto" type cannot appear in top-level array type

5605: "auto" type cannot appear in top-level function type

5606: a member of type "type” cannot have an in-class initializer

5607: a member with an in-class initializer must be const

5608: cannot deduce "auto" type (initializer required)

5609: "auto" type is "type”1 for this entity, but was previously implied to be "type”2

5610: invalid constructor declaration

5611: invalid use of a type qualifier

5612: a union cannot be abstract or sealed

5613: "auto" is not allowed here

5614: definition of base class type not completed yet

5615: "extern template" cannot refer to a specialization of static entity-kind "entity”

5616: "extern template" cannot follow explicit instantiation of entity-kind “entity”

5617: __declspec(restrict) requires a function returning a pointer type

5618: the "report_gnu_extensions" option is only valid in GNU C and GNU C++ modes

5619: variable-length array types are nonstandard

5620: designators are nonstandard

5621: this designator syntax is a GNU extension

5622: compound literals are nonstandard

5623: statement expressions are a GNU extension

5624: asm name ignored for previously defined entity

5625: attributes are a GNU extension

5626: extended asm syntax is a GNU feature

5627: volatile asm declarations are a GNU extension

5628: asm name specifiers are a GNU extension

5629: the "__restrict" qualifier is nonstandard

5630: "typeof" is a GNU extension

5631: modifying the size or signedness of a typedef is nonstandard

5632: zero-length arrays are a GNU extension

5633: flexible array members are nonstandard

5634: attribute "nonnull" references nonpointer parameter

5635: argument for attribute "nonnull" is larger than number of parameters

5636: no parameter has pointer type

5637: null argument provided for parameter marked with attribute "nonnull"

5638: the destructor for "type1” has been suppressed because the destructor for "type2” is inaccessible

5639: the suppressed destructor for "type” is needed

5640: routine is both "inline" and "noinline" ("noinline" assumed)

5641: invalid cleanup routine

5642: attribute "cleanup" requires automatic storage duration

5643: attribute "cleanup" does not apply to parameters

5644: cleanup routine has invalid type

5645: call of cleanup routine requires suspect conversion

5646: __sptr and __uptr must follow a “*”

5647: __sptr and __uptr cannot both be specified

5648: widening pointer conversion from "type1” to "type2” extends sign bit

5649: __sptr and __uptr don't apply to pointer-to-member types

5650: the declaration of the copy assignment operator for "type” has been suppressed because entity-kind “entity” is const

5651: the declaration of the copy assignment operator for "type” has been suppressed because entity-kind “entity” has reference type

5652: the declaration of the copy assignment operator for "type1” has been suppressed because that of "type2” was suppressed

5653: the declaration of the copy assignment operator for "type1” has been suppressed because that of "type2” is ambiguous

5654: the declaration of the copy assignment operator for "type1” has been suppressed because that of "type2” is inaccessible

5655: the declaration of the copy constructor for "type1” has been suppressed because that of "type2” was suppressed

5656: the declaration of the copy constructor for "type1” has been suppressed because that of "type2” is ambiguous

5657: the declaration of the copy constructor for "type1” has been suppressed because that of "type2” is inaccessible

5658: the destructor for "type1” will not be called because it is inaccessible and the destructor for "type2” was suppressed

5659: definition at end of file not followed by a semicolon or a declarator

5660: first argument must be a pointer to integer or enumeration type

5661: synchronized operations are valid only on objects of size 1, 2, 4, or 8

5662: extra arguments ignored

5663: '=' assumed following macro name “xxxx” in command-line definition

5664: white space is required between the macro name “xxxx” and its replacement text

5665: result of call is not used

5666: attribute "warn_unused_result" is ignored for void return type

5667: attribute “xxxx” requires pointer-to-function type

5668: dllimport/dllexport is ignored on redeclaration using a qualified name

5669: too many characters in character literal -- extra leading characters ignored

5670: entity-kind “entity” cannot be declared inline after its definition at line xxxx

5671: a statement expression is not allowed in a template-dependent typeof specifier

5672: a statement expression is not allowed in a decltype specifier

5673: a template argument may not reference a type with no name linkage

5674: "virtual" is ignored here

5675: a template argument may not reference a variable-length array type

5676: a universal character name cannot designate a surrogate code point

5677: #include_next cannot be used in the primary source file

5678: "entity" cannot be specified in a template member definition -- "entity" assumed instead

5679: attribute "xxxx" is ignored on local function declaration

5680: concatenation with "xxxx" in entity-kind "entity" does not create a valid token

5681: "entity" is ambiguous (entity-kind "entity" assumed)

5682: a type qualifier is not allowed on a static member function

5683: a type qualifier is not allowed on a constructor or destructor

5684: a type qualifier is not allowed on operator new or operator delete

5685: a type qualifier is not allowed on a nonmember function

5686: __assume expression with side effects discarded

5687: unrecognized Unicode source kind (must be one of UTF-8, UTF-16, UTF-16LE, UTF-16BE):

5688: Unicode character with hex value xxxx not representable in preprocessing output

5689: requested constructor/destructor priority is reserved for internal use

5690: unrecognized GCC pragma

5691: unrecognized GCC visibility pragma directive

5692: unrecognized visibility kind

5693: visibility pragma was still active

5694: no matching visibility push

5695: typeid of incomplete type

5696: __declspec attributes are not allowed on enum specifiers in C mode

5697: array entity-kind "entity" assumed to have one element

5698: vector_size attribute requires an arithmetic or enum type

5699: vector size is too large

5700: vector size must be a power of two

5701: vector size must be a multiple of the element size

5702: mixed vector-scalar operation not allowed

5703: operation requires two vectors of the same size

5704: template-dependent vector size is not allowed

5705: vector_size attribute is not allowed on a template-dependent type

5706: vector_size attribute is not allowed here

5707: vector_size attribute is not allowed with a complex element type

5708: vector size must be an integer constant

5709: vector operation requires identical element types

5710: vector operation does not apply to vector with non-integral type

5711: cannot open xxxx file “xxxx”

5712: cannot open xxxx file “xxxx”: xxxx

5718: error while writing xxxx file: xxxx

5727: IL output

5728: conversion drops "__restrict" qualifier

5729: unable to obtain mapped memory for “xxxx”: xxxx

5730: restrict qualifier is ignored

5731: attributes ignored here

5732: array of elements containing a flexible array member is nonstandard

5733: a template parameter may not have a vector type

5734: the initialization of entity-kind “entity” will be done before that of entity-kind “entity”

5735: inheritance kind is not allowed in C

5736: inheritance kind is ignored on an enum specifier

5737: modifier is not allowed on an enum specifier

5738: modifier is ignored on an enum specifier

5739: identifier character cannot be represented in Unicode

5740: header name contains characters that cannot be represented in Unicode

5741: “xxxx” is not a valid locale name

5742: declaring a void parameter list with a template parameter is nonstandard

5743: lambdas option can be used only when compiling C++

5744: explicit capture matches default

5745: entity-kind “entity” is not a variable

5746: a variable with static storage duration cannot be captured in a lambda

5747: "this" cannot be captured by reference

5748: "this" cannot be used inside the body of this lambda

5749: a member of an outer-scope anonymous union cannot be referenced inside the body of a lambda

5750: an enclosing-function local variable cannot be referenced in a lambda body unless it is in the capture list

5751: invalid reference to an outer-scope local variable in a lambda body

5752: a local variable outside the current function scope cannot be captured

5753: the enclosing-function "this" cannot be referenced in a lambda body unless it is in the capture list

5754: the body of a value-returning lambda with no explicit return type must be a single return statement

5755: lambda captured variable of type "type1” cannot be copied to closure class field of type “type2”

5756: invalid template directory:

5764: enumeration value is outside the range of its underlying type (“type”)

5765: "\\" followed by white space is not a line splice

5766: this dynamic_cast cannot be done without runtime type information, which is disabled

5767: conversion to "type” is ambiguous; direct base selected

5768: an internal buffer would be too large

5769: C++ exception handler used, but exception handling semantics have not been specified

5770: type qualifier ignored on constructor

5771: a variable captured by a lambda cannot have a type involving a variable-length array

5772: conversion between incompatible vector types

5773: expected a "{" introducing a lambda body

5774: rvalue references option can be used only when compiling C++

5775: a type qualifier is not allowed on a lambda

5776: a name cannot appear more than once in a capture-list

5777: explicit template arguments ignored

5778: a lambda is not allowed in a constant expression

5779: "type” is not a class type

5780: "delete" applied to a pointer-to-array type treated as delete[]

5781: "delete" applied to a pointer-to-array type is nonstandard; treated as delete[]

5782: entity-kind “entity” cannot be called with the given argument list

5783: an rvalue reference cannot be bound to an lvalue

5784: a nontype template parameter cannot have rvalue reference type

5785: type qualifiers are ignored (underlying type is a reference)

5786: entity-kind “entity”, declared using a local type, must be defined in this translation unit

5787: entity-kind “entity”, declared using a type with no linkage, must be defined in this translation unit

5788: the operand of an rvalue reference dynamic_cast must have a complete class type

5789: "= default" can only appear on default constructors, copy constructors, copy assignment operators, and destructors

5790: "= delete" can only appear on the first declaration of a function

5791: entity-kind “entity” (declared at line xxxx) cannot be referenced -- it is a deleted function

5792: a lambda is not allowed in an unevaluated expression

5793: __builtin_va_arg_pack/__builtin_va_arg_pack_len can appear only in an inline function with an ellipsis parameter

5794: "= default" cannot be specified on a friend declaration

5795: expected a C++ keyword

5796: this cast to an rvalue reference type is invalid because the underlying type "type” is incomplete

5797: offset is not constant

5798: unrecognized #pragma comment type “xxxx”

5799: option to control whether "auto" is a type specifier can be used only when compiling C++

5800: option to control whether "auto" is a storage class can be used only when compiling C++

5801: the type specifier and storage class specifier meanings of "auto" cannot both be disabled

5802: invalid string in #pragma comment

5803: deleted function overrides nondeleted entity-kind “entity”

5804: nondeleted function overrides deleted entity-kind “entity”

5805: the default constructor of "type” cannot be referenced -- it is a deleted function

5806: an rvalue reference is not allowed as a catch type

5807: default arguments of entity-kind “entity” is incompatible with a declaration in another translation unit

5808: default arguments of entity-kind “entity” were different during compilation of “xxxx”

5809: entity-kind "entity” cannot be specialized because it is deleted

5810: initializer for entity-kind “entity” is different in another translation unit

5811: initializer for entity-kind “entity” was different during compilation of “xxxx”

5812: a designator into a template-dependent type is not allowed

5813: unrecognized conformance kind

5814: expected "on" or "off"

5815: #pragma conform(forScope) stack is empty

5816: no previous #pragma conform(forScope) entry matches “xxxx”

5817: forScope behavior is nonstandard

5818: forScope behavior is standard

5819: function "main" cannot be deleted

5820: type qualifiers are meaningless here

5821: invalid type for defaulted assignment operator

5822: function templates cannot be defaulted

5823: invalid type for defaulted constructor

5824: function call requires one argument

5825: function call requires a real floating-point argument

5826: a copy constructor with a default argument cannot be defaulted

5827: a predeclared function cannot be deleted

5828: empty dependent statement in if-statement

5829: empty dependent statement in "else" clause of if-statement

5830: entity-kind “entity” (declared at line xxxx), required for copy that was eliminated, cannot be referenced -- it is a deleted function

5831: nonstandard first parameter "type” of "main", expected "int"

5832: nonstandard number of parameters for "main", expected zero or two parameters

5833: nonstandard second parameter "type” of "main", expected "char *[]" or "char **"

5834: “xxxx” was specified as both a system and non-system include directory -- the non-system entry will be ignored

5835: option to control move constructors and move assignment operators can be used only when compiling C++

5836: a nonpublic or explicit member function cannot be defaulted in its parent class definition

5837: "packed" attribute ignored on class with non-POD entity-kind "entity”

5844: function *<function_name>* **has no prototype**

5848: GNU-style inline assembly detected. This feature is not supported.

The Diab compiler does not support GNU-style inline assembly extended syntax. This warning is emitted if such syntax is detected.

5849: suspicious extension of a 32-bit value when assigned to a 64-bit integral type (potential portability problem).

This warning is reported as 2273 for previous versions.

4. ASSEMBLER ERROR MESSAGES

1. Assembler Error Messages

Assembler messages have the format:

```
"file", line #: severity: message
```

Three kinds of messages are generated. The severity values for each as they appear in messages are as follows.

warning

Warning: a message will be printed, assembly will continue, and an output file will be produced.

error

Error: a message will be printed, assembly will continue, but no output will be generated.

fatal

Fatal: a message will be printed and assembly aborted.

Assembler messages are intended to be clear in the context of the error and are not listed here. Please report unclear assembler error messages to Customer Support.

5. LINKER ERROR MESSAGES

1. Linker Message Format

Linker messages have the format:

```
DLD.EXE: message
```

Where relevant, the file and line are included in the message.

The severity level for each message is shown in parentheses in the message description. A warning (w) generates a diagnostic message, but linking continues and an output file is produced. An error (e) causes the linker to abort.

2. Linker Message Detail

The following is a list of possible linker error messages.

"." (0x...) is assigned invalid value: 0x...

Assignment to "." creates a gap in section data. The size of this gap should not be negative and should be less 0x4000000. (e)

Absolute section has invalid name: name

Absolute section name must be ".abs.*hexNumber*". (e)

An unknown or incorrect option has been provided

The linker does not recognize an option flag that has been passed to it. (w)

Archive file *filename* **does not have symbol tabl**e

An archive file must have a symbol table to be usable by the linker. Use **dar** to create the table. (e)

ASSERT failed: *assertion*

(Message may include the **assert** expression.) Contact Customer Support. (e)

Assignment to symbol "*symbol***" in the LECL file is ignored** The symbol is defined in an input object file

The linker command file cannot redefine a symbol that is already defined in an input object file. (w)

Cannot allocate 0x... bytes of memory for "*name*"

The **MEMORY** directive in the linker command language is used to specify the regions from which the linker can allocate memory. When there is not enough space to contain a group, section, or **NEXT** directive, an error message is generated. (e)

Cannot allocate branch island

The linker cannot calculate the address or size of a branch island. The circular dependencies are too complex. (e)

Cannot calculate address of group

Complex circular dependencies cannot be resolved. Linker command language and implicit linking rules constitute an equation system which can be unsolvable, resulting in this or similar error message. (e)

Cannot calculate address of section *section*

Complex circular dependencies cannot be resolved. Linker command language and implicit linking rules constitute an equation system which can be unsolvable, resulting in this or similar error message. (e)

Cannot calculate OVERFLOW size expression

Complex circular dependencies cannot be resolved. An expression value depends on the address or size of a symbol or section, which in turn depends directly or implicitly on the expression value. Example:

```
X = SIZEOF(Y); Y (DATA) : { . = . + X; }
```

Linker command language and implicit linking rules constitute an equation system which can be unsolvable, resulting in this or similar error message. (e)

Cannot calculate size of group

Complex circular dependencies cannot be resolved. Linker command language and implicit linking rules constitute an equation system which can be unsolvable, resulting in this or similar error message. (e)

Cannot create branch island - section *section* **is too large**

Branch islands are created between input sections. If an input section is too large it might not be possible to create an island for that branch.

Cannot create Branch Island for Arm to Thumb call, function *name*

Contact Customer Support. (e)

Cannot create Branch Island for Thumb to Arm call, function *name*

Contact Customer Support. (e)

Cannot create position independent branch island: __SDA2_BASE_ is undefined

-Xpic-only needs the symbol **__SDA2_BASE_** to be defined. (e)

Cannot evaluate expression

Complex circular dependencies cannot be resolved. Linker command language and implicit linking rules constitute an equation system which can be unsolvable, resulting in this or similar error message. (e)

Cannot evaluate fill value expression

Complex circular dependencies cannot be resolved. Linker command language and implicit linking rules constitute an equation system which can be unsolvable, resulting in this or similar error message. (e)

Cannot evaluate value of symbol *symbol*

Complex circular dependencies cannot be resolved. Linker command language and implicit linking rules constitute an equation system which can be unsolvable, resulting in this or similar error message. (e)

Cannot find matching input sections for "..."

Input section specification does not match any input. (w)

Cannot find overflow output section "*section* "

Invalid section name in **OVERFLOW** statement. No such section defined in linker command file. (e)

Cannot get current directory name

Call to **getcwd( )** failed. (e)

Cannot rename "*filename***", error:** *message*

The host operating system reported an error renaming the file. Check the permissions on the directory where the file resides. This usually means that you are not permitted to write in that directory. (e)

Cannot write relocation table: relocation type 0*x***... is not supported by COFF**

This can occur when input and output have different formats (ELF to COFF) and some relocations cannot be converted. (e)

Cannot allocate memory (NEXT)

The **MEMORY** directive in the linker command language is used to specify the regions from which the linker can allocate memory. When there is not enough space to contain a group, section, or **NEXT** directive, an error message is generated. (e)

Cannot calculate size of section "*section***":** "." **(0x...) is assigned invalid value: 0***x***...**.

Can't calculate size of section *section* : **it depends on section address ...**

Can't calculate size of section *section* : **it depends on section address....** The section might require alignment specification

Complex circular dependencies cannot be resolved. Linker command language and implicit linking rules constitute an equation system which can be unsolvable, resulting in this or similar error message. (e)

Can't create file *name*

Can't create file *name* : ...

The host operating system returned an error when **dld** tried to create a file. The permissions in the current directory probably don't allow your **dld** command to write in the directory. (e)

Can't create tempfile *name* : ...

The host operating system returned an error when **dld** tried to create a file. The permissions in the current directory probably don't allow your **dld** command to write in the directory. (e)

Can't find file: *filename*

The linker cannot locate the specified file. (e)

Can't find library: *libname* **.a**

The linker cannot locate the specified library. (e)

Can't find output section *section*

Invalid section name in linker command language expression. (e)

Can't find section *section*

Invalid section name in linker command language expression. (e)

Can't lseek on *name* : ...

Possibly an external task has shortened the file. More likely, this represents an internal error in the **dld** code. Please collect a test case to reproduce the problem and contact Customer Support. (e)

Can't open *filename* : ...

The host operating system returned an error when **dld** tried to read the file. Check the permissions on the file and the full pathname to the file. Perhaps there is a spelling error in the path. (e)

Can't open tempfile *name* : ...

The host operating system returned an error when **dld** tried to read the file. Check the permissions on the file and the full pathname to the file. Perhaps there is a spelling error in the path. (e)

Can't search unused sections, main entry symbol "*symbol***" is undefined**

This warning should not be generated since the current linker deletes such symbols silently. (w)

Can't search unused sections, main entry symbol "*symbol***" has absolute address**

This warning should not be generated since the current linker deletes such symbols silently. (w)

COMMON object is eclipsed by a function definition:

Function name: *name*

File: *filename*

A symbol of type **function** is defined with the same name as a **COMMON** object. (w)

Compression switch function "*function*" **is undefined**

PowerPC compressed code only. When **-Xmixed-compression** is on, symbols **__switch_to_uncompressed** and **__switch_to_compressed** must be defined in an input object files. (e)

Don't know where to allocate input section:

no matching input specification found in linker command file.

Section name: *section*

File: *filename*

Change linker command file to include explicit instructions on how to link this section. If the "section name" referred to in the message is **.ctors** or **.dtors**, you may be using an old linker command file that specifies **.init** and **.fini** instead of **.ctors** and **.dtors**. (w)

Don't know where to put COMMONs! No .bss and no COMMON directive

Found a **COMMON** variable but linker command file has no **.bss** nor **COMMON**. (e)

Don't know where to put small COMMONs! No .sbss and no SCOMMON directive

Found a small **COMMON** variable but linker command file has no **.sbss** nor **SCOMMON**. (e)

End of memory

All internal structures used in the linker are dynamically allocated. When the host operating system cannot provide more memory, the linker aborts with an error message. On UNIX, change the amount of memory your shell allows with the **limit** or **ulimit** command; if that does not work, increase your swap area. On Windows, increase your swap area (virtual memory). (e)

Environment variable "RTAPROJECT" must be set

The variable must be set when **-Xgenerate-vmap** is used. (This option is not intended to be set by the user.) (e)

Failed to read file *name* : ...

The host operating system reported a read error. Perhaps the file's permissions were changed by another task after **dld** opened it successfully. (e)

Failed to read file *name* : **file is empty**

The host operating system reported less data in the input file than **dld** expected. Probably the file is corrupted or was only partially written because the file system filled up before its writes were completed. You should recreate the file and retry your **dld** comm**and. (e)**

Failed to read file name from archive *name*

The host operating system reported a read error. Perhaps the file's permissions were changed by another task after **dld** opened it successfully. (e)

Failed to read from file *name* : ...

Failed to read from file *name* **(**...**)**: ...

The host operating system reported a read error. Perhaps the file's permissions were changed by another task after **dld** opened it successfully. (e)

Failed to read from file *name* : **end of file**

The host operating system reported less data in the input file than **dld** expected. Probably the file is corrupted or was only partially written because the file system filled up before its writes were completed. You should recreate the file and retry your **dld** command. (e)

Failed to write to file *name* : ...

The host operating system reported a write error. Perhaps the file's permissions were changed by another task after **dld** opened it successfully. Perhaps the file partition has filled up, leaving insufficient room for the file. (e)

File *filename* **does not have symbol table section**

File *filename* **(**...**) does not have symbol table section**

Invalid input file: no symbol table. (e)

File *filename* **has invalid relocation section File** *filename* **(**...**) has invalid relocation section**

Invalid input file: invalid reference to relocation information. (e)

File has wrong byte order, file *filename*

Invalid ELF header: Byte order neither big-endian nor little-endian. (e)

File has wrong class, file *filename*

Invalid or unsupported ELF class in input file header. (e)

File has wrong version, file *filename*

Invalid or unsupported ELF version in input file header. (e)

File is not an ELF file, file *filename*

Linker assumed file to be ELF but it does not have valid ELF header. (e)

File *filename* **is not of known format**

Supported formats are COFF, ELF, archive, and linker command language. (e)

File "*filename***", section "***section***", offset 0***x* ...**: Invalid relocation:** ....

Input object file has relocation entry which cannot be processed. (e)

File type is not COFF, file *filename*

Contact Customer Support. (e)

File type is not ELF, file *filename*

Contact Customer Support. (e)

Generation of relocation entries without a symbol table is not possible

Invalid **-s** option. (e)

... has BIND address, "> *area-name* **" specification is ignored**

Contact Customer Support. (w)

Hole moved because of section overflow

A warning is given if a "hole" has to be moved away. (w)

For further details, please refer to OVERFLOW Specification.

Illegal -B option

-B must be followed by "**=**". (e)

Illegal expression

Contact Customer Support. (e)

Illegal filename *prefix* [**COMMON**]**, only * is allowed**

Input specification must be *****[**COMMON**], not **xyz.o**[**COMMON**]. (e)

Illegal option *option*

Option is not recognized. (w or e)

Illegal option -X*option*

Option is not recognized. (e)

Illegal usage of HEADERSZ in LECL file

Contact Customer Support. (e)

Illegal -Y option

-Y must be followed by "**,**". (e)

In file "*filename* **", Section** "*section* **Section offset 0***x* **..,** **Symbol "***symbol* **"** **Invalid relocation entry**

Input file has broken symbol table or relocation information. (e)

In file *filename* **, symbol** *symbol* **has invalid value: symbol is undefined (state 0***x* **...), but value is not zero - 0***x* ...

Invalid input file: The symbol table is defective. (w)

In LECL file "*filename* **", line** *number* **,** *name* **is not allocable, ">** *name* **" specification is ignored**

The name of a section or group is a reserved word. Change the name of the section or group to a word that is not linker file format keyword. Refer to the compiler user's guide for information on linker file formats. (w)

Input contains mix of little-endian and big-endian object files: **Aborted**...

Linking a mix of little-endian and big-endian object files is not supported. (e)

Input contains mix of PPC COFF and ELF object files: **PPC COFF and ELF object files have incompatible calling conventions**

Mixing PowerPC COFF and PowerPC ELF is dangerous. (w)

Input files contain code for mixed processors: Only one file for each processor type is listed

Mixing code generated for different CPU types is dangerous. (w)

Insufficient memory

All internal structures used in the linker are dynamically allocated. When the host operating system cannot provide more memory, the linker aborts with an error message. On UNIX, change the amount of memory your shell allows with the **limit** or **ulimit** command; if that does not work, increase your swap area. On Windows, increase your swap area (virtual memory). (e)

Internal error: cannot calculate COFF header size

Contact Customer Support. (e)

Internal error: cannot calculate ELF header size

Contact Customer Support. (e)

Internal error: can't ADD symbol to non-hashed table

Contact Customer Support. (e)

Internal error: error counting undefines

Contact Customer Support. (e)

Internal error: illegal output file type

Contact Customer Support. (e)

Internal error: illegal/unsupported output format ...

Contact Customer Support. (e)

Internal error: no output file type set

Contact Customer Support. (e)

Internal error: not relocinfo

Contact Customer Support. (e)

Internal error: output buffer overflow

Contact Customer Support. (e)

Internal error: should not happen

Contact Customer Support. (e)

Invalid archive format, file *filename*

Archive file has invalid format. (e)

Invalid archive symbol table, file: *filename*

Invalid input file: The symbol table is defective. (e)

Invalid file header, file *filename* **in archive** *archive*

Contact Customer Support. (e)

Invalid fill pattern alignment, must be 1, 2, or 4

Invalid fill specification in section definition (**SECTIONS** command). (e)

Invalid fill pattern size, must be 1, 2, or 4

Invalid fill specification in section definition (**SECTIONS** command). (e)

Invalid option format: *option*

Valid format is *-optionName*[*=number*]. (e)

Invalid relocation info: File "*filename***" Section "***section***" Section address 0***x***...size 0***x***... Relocating reference at address 0***x***... Can't relocate**

Input object file has broken relocation information. (e)

Invalid section header in file "*filename*"**, section name** "*name*"

Invalid input file: Invalid **COMDAT** section header. (e)

Invalid value of -Xmax-long-branch= option

The option sets the maximum branch offset which does not need a branch island. Some targets (like the PowerPC) have short and long branch instructions. Valid values are 2..0x7fffffff; using the option without a value is an error. (e)

Invalid value of -Xmax-short-branch= option

Valid values are 2..0x7fffffff. Using the option without a value is an error. (e)

Machine type not supported, file *filename* **Machine type not supported, file** *filename* **(...)**

Invalid input file: unsupported target CPU. (e)

Memory area "*area-name*" **is full**

Memory area specified in "**>** *area-name*" is full. (e)

Memory area "*area-name*" **is undefined**

Invalid name in "**>** *area-name*" specification. (e)

Memory block extends over 32 bit address range: ...

memory address + *memory size* >= 0x100000000. (w)

Next alignment with zero!

Invalid argument of **NEXT( )**. (e)

No main entry point defined

Executable output needs an entry point. (e)

No section names in file *filename*

Invalid input file: no section names string table. (e)

No string table in file *filename*

Invalid input file: no string table. (e)

Not all files are compiled with same value of "-Xsmall-data-registers"

If all the source files (including assembly sources) were not compiled with the same setting of -Xextra-base-registers=N, then this warning is shown.

Nothing to link

No object files are given in the command line. (e)

Only one COMMON allowed in LECL file

More than one input specification like ***[COMMON]** is not allowed in the linker command file. (e)

Only one SCOMMON allowed in LECL file

More than one input specification like ***[SCOMMON]** is not allowed in the linker command file. (e)

Out of memory reading archive *archive*

All internal structures used in the linker are dynamically allocated. When the host operating system cannot provide more memory, the linker aborts with an error message. On UNIX, change the amount of memory your shell allows with the **limit** or **ulimit** command; if that does not work, increase your swap area. On Windows, increase your swap area (virtual memory). (e)

Output file format not specified

Contact Customer Support. (e)

Output section *"section"* **contains mix of compiled for compression and normal sections: The output section will not be prepared for compression**

Mixing compressed and normal code in one section is illegal. (w)

Output sections: have overlapping load addresses

Incompatible specification of output sections. (e)

Output sections: have overlapping run-time addresses

Incompatible specification of output sections. (e)

Overlapping memory block *block*

Two or more **MEMORY** directives define the same memory area. (w)

Redeclaration of *symbol*

More than one definition of a symbol which is not **COMMON** or weak.

Register number in REGISTER( ) section specification must be in 0.*n* **range**

Invalid register specification. (e)

Relocation error in file *filename* **: section** *section* **refers to local symbol** *symbol* **in section** *section* **and section** *section* **is not taken to output**

Linker failed to remove unused sections properly. file a SPR. Contact Customer Support. (e)

Relocation error in file *filename* : **section** *section* **refers to local symbol** *symbol* **at section** *section* **and section** *section* **is purged COMDAT section**

Linker failed to remove unused **COMDAT** sections. Contact Customer Support. (e)

Relocation info is not properly sorted, file *filename* **, section** *section* **Relocation info is not properly sorted, file** *filename* **(...), section** *section*

Input file has broken relocation information. (e)

Section .data (DATA) is not defined

COFF output must have a **.data** section. (e)

Section e_shstrndx is not a SHT_STRTAB in file "*filename***"**

Section e_shstrndx is not a SHT_STRTAB in file "f*ilename***(...)"**

Invalid input file: invalid ELF header. (e)

Section *section* **extends over 32-bit address range**

section address + *section size* >= 0x100000000. (w)

Section .text (**TEXT**) **is not defined**

COFF output must have a **.text** section. (e)

Symbol "*symbol*" **can't be declared relative Symbol is declared as "... @ ... = ... Section "***section***" is empty - can't be used for relative declaration**

A section must have some input section to make relative declaration possible. (w)

Symbol "*symbol*" **can't be declared relative Symbol is declared as "... @ ... = ..." Symbol "***symbol***" is absolute - can't be used for relative declaration**

Base symbol must be declared inside a section. (w)

Symbol definition "*name*" **not found**

Symbol name is used in linker command file but symbol is undefined. (e)

Symbol definitions missing at index *index* **in** *name*

Contact Customer Support. (e)

Symbol "*symbol*" **has unknown binding type**

Contact Customer Support. (e)

Symbol *symbol* **has unknown section index**

Invalid symbol table in input ELF file. (w)

Symbol *symbol* **has unknown symbol type**

Input file has a symbol of an unknown or unsupported type. (e)

Symbol *symbol* **in** *name* **is defined in unknown section**

Invalid section table in input ELF file. (w)

Symbol *symbol* **is declared with more than one size Symbol** *symbol* **is declared with more than one size (***n* **and** *m***)**

Conflicting definition for a **COMMON** variable. (w)

Symbol *symbol* **is undefined but not used**

This warning should not be generated since the current linker deletes such symbols silently. (w)

Symbol *name* **missing. Must be defined when using shared libraries.**

This message is no longer used. (e)

Symbol or section "*name*" **not found**

Invalid name in relative symbol definition in linker command file. (e)

Symbol _SDA_BASE_ is undefined Symbol _SDA2_BASE_ is undefined Symbol _SDA3_BASE_ is undefined

The symbol **_SDAx_BASE_** is needed to process SDA (Small Data Area) relocations. (e)

Target architecture is not specified

Unknown target. (e)

The total size of the small data and constant for this program requires using N additional SDA base registers. Consider compiling with the option -Xextra-base-registers=N

The total size of near data and near consts used by the program cannot be addressed with default base registers. However, the program can still link if the user uses -Xextra-base-registers=N as suggested.

Undefined symbol "*symbol*" **Undefined symbol** "*symbol*" **in file** "*filename*" **Undefined symbol** "*symbol*" **in file** "*filename* **(...)**"

An undefined symbol is referenced. (w)

Undefined symbols found - no output written

The **MEMORY** directive in the linker command language is used to specify the regions from which the linker can allocate memory. When there is not enough space to contain a group, section, or **0NEXT** directive, an error message is generated. (e)

Unknown relocation type in *name*

Contact Customer Support. (e)

Unsupported file format: "*name*"

Supported formats are COFF, ELF, archive, and linker command language. (e)

Unsupported file type in archive

Supported formats in archives are COFF and ELF. (e)

Unsupported output file format

Selected combination of object-file format and target is not supported. (e)

Unsupported relocation type ... Unsupported relocation type in file "*filename*"

Input file has unsupported relocation type. (e)

Unused symbols search failure, symbol: *symbol*

The linker failed while attempting to find and delete unused symbols in object files. This could be caused by a linker bug, or by an object file that is corrupt, invalid, or in an unsupported format. (e)

Use -Xmixed-compression command line option to enable generation of compression switches

PowerPC compressed code only. The switches are codes which change the CPU mode from compressed code to normal code and back. (e)

Value of "." is undefined outside a section or group

Illegal use of "**.**" in linker command file. (e)

-Xstop-on-warning is on, linking aborted

The linker stopped after issuing a warning because the **-Xstop-on-warning** option is enabled. (e)