# Freescale

The following Freescale compilers are supported:

- Freescale C/C++ Compiler v. 5.1 for Embedded ARM
- Freescale CodeWarrior ANSI-C/cC++ Compiler 5.0.x for HC12
- FreeScale CodeWarrior ColdFire v 6.0

# Freescale C/C++ Compiler v. 5.1 for Embedded ARM

- Compiler acronym: cwarm\_5\_1
- Host OS: Windows
- The CodeWarrior IDE is not supported.
- Supported practices: static analysis
- Support level: Extended

## Freescale CodeWarrior ANSI-C/cC++ Compiler 5.0.x for HC12

- Compiler acronym: cwhc12\_5\_0
- Host OS: Windows
- The CodeWarrior IDE is not supported.
- Supported practices: static analysis
- Support level: Extended

## FreeScale CodeWarrior ColdFire v 6.0

- Compiler acronym: cwcf\_6\_0
- Host OS: Windows
- Supported languages: c89, c99
- Supported practices: static analysis
- The CodeWarrior IDE is supported (static analysis only)
  Support level: Standard

#### ...

## Additional Support Information

To use any supported distribution, the directory containing the command line compiler driver executable must be included in the \$PATH environment variable.

In C++, non-standard "direct constructor calls", such as BitSet::BitSet(. . .) in the following example, are not supported:

```
class BitSet { protected:
    enum BS_op {
        BS_and = (int) '&', BS_or = (int) '|',
        };
    BitSet(const BitSet& x, const BitSet& y, enum BS_op op);
    friend BitSet operator & (const BitSet& x, const BitSet& y);
        };
BitSet operator & (const BitSet& x, const BitSet& y)
    {
    return BitSet::BitSet(x, y, BitSet::BS_and);
    }
```

The Freescale HC(S)12 compiler accepts this non-standard construction. It is used in the bitset.h C++ header file shipped with the compiler. Code that uses this header file cannot be analyzed; it is currently not supported.

The Freescale HC(S)12 compiler selects an overloaded function with a non-plain char (signed char, unsigned char) parameter for a function call with a plain char argument. This is not supported. Example:

```
class istream {
   istream& get(signed char p);
   istream& get(unsigned char p) {
        return get((char)'a');
        }
   };
```

The Freescale HC(S)12 compiler accepts this construction. It is used in the isstream.h C++ header file shipped with the compiler. Code that uses this header file cannot be analyzed; it is currently not supported.

In C++, non-standard constructions that use multiple type specifiers, such as streamsize int in the following example, are not supported:

```
class istream {
  typedef int streamsize;
  istream& read(streamsize int n);
      };
```

The Freescale HC(S)12 compiler accepts this non-standard construction. It is used in the isstream.h C++ header file shipped with the compiler. Code that uses this header file cannot be analyzed; it is currently not supported.

In C++, implicit conversions from void\* pointer types to other pointer types are not supported. Example:

```
class ios {
    int i;
        };
int vscan(ios* stream = ((void *)0));
```

The Freescale HC(S)12 compiler accepts this construction. It is used in the stream.h C++ header file shipped with the compiler. Code that uses this header file cannot be analyzed; it is currently not supported.

The built-in long long double type is not supported.

### **About Support Levels**

- Extended: Support has been validated with extended testing and is approved for use in safety-critical software development.
- Standard: Support has been validated with standard testing and is approved for use in non-safety critical software development.