

Renesas RX GNU Toolchain

14.2.0.202607

Release Notes

30th of July, 2026

About Renesas RX GNU Toolchain 14.2.0.202607

CyberThor Studios Ltd. is releasing the Renesas RX GNU Toolchain 14.2.0.202607, a cross compiler tool for Renesas RX micro-controllers.

Release Version: Renesas RX GNU Toolchain 14.2.0.202607

Release Date: 30th of July, 2026

Platforms Supported:

- Ubuntu 22.04 or later (or compatible distribution)
- Windows 7 or later
- macOS Ventura 13.4 on Apple M1

Language: C, C++

Targets: All RX devices

Object File Format: ELF

The Renesas RX GNU Toolchain 14.2.0.202607 toolchain is based on:

- gcc 14.2.0 [released]
- binutils 2.44 [released]
- newlib 4.4.0 [released]
- gdb 16.2 [released]

The latest patches are applied to the sources.

Changes in Renesas RX GNU Toolchain 14.2.0.202607

This section describes the fixes made in the Renesas RX GNU Toolchain 14.2.0.202607 release.

1. [Improvement] Added Renesas RXv3e Protect Zone support

The current release of the toolchain adds support for RXv3e Protect Zone via the `-mprotect-zone` command line option. Please consult the included GCC manual and RXv3e software/hardware manuals for details.

2. [Improvement] Added `-mtfu-security-attribute=[secure/nonsecure]` command line option

The `-mtfu-security-attribute` command line option was added to allow selection of the memory region address to be used as the TFU base address.

3. [Change] `rx-elf-libgen` and `-mtfu` option interaction

All MTFU related options are filtered out from the `rx-elf-libgen --compiler-options` command line option, via the associated makefiles. This is done to avoid possible address inconsistencies with the introduction of Protect Zone.

4. [Bugfix] Fixed `-mtfu=intrinsic,mathlib` not working in C++

The `-mtfu=intrinsic,mathlib` command line option did not correctly replace calls to the `sinf`, `cosf`, `atan2f`, `hypotf` functions with their builtin versions when compiling C++ code. This was fixed in the current release.

5. [Bugfix] Fixed 64bit subtraction bug

Certain 64bit subtractions could cause the compiler to fail to select an instruction and crash. This regression is present in the 202505 and 202511 releases of the toolchain. This was fixed in the current release.

Notes

This installer does not provide an option to integrate the GNURX toolchain with e2 studio, as the e2 studio IDE will automatically detect the GNURX toolchain installation on start-up for integration. Alternatively, you may use the 'Toolchain Management' feature in e2 studio to achieve this.

For details on e2 studio, please visit the following link below: <https://www.renesas.com/eu/en/software-tool/e-studio>

There is no support in this installer to integrate the toolchain with the HEW IDE.

Known Issues in Renesas RX GNU Toolchain 14.2.0.202607

The following is a list of known issues for the tools included in the Renesas RX GNU Toolchain 14.2.0.202607 toolchain:

1. -Wreturn-type is enabled by default

G++ now assumes that control never reaches the end of a non-void function (i.e. without reaching a return statement). This means that you should always pay attention to -Wreturn-type warnings, as they indicate code that can misbehave when optimized.

To tell the compiler that control can never reach the end of a function (e.g. because all callers enforce its preconditions) you can suppress -Wreturn-type warnings by adding `__builtin_unreachable`:

```
char signchar(int i) // precondition: i != 0
{
    if (i > 0)
        return '+';
    else if (i < 0)
        return '-';
    __builtin_unreachable();
}
```

Because -Wreturn-type is now enabled by default, G++ will warn if main is declared with an implicit int return type (which is non-standard but allowed by GCC). To avoid the warning simply add a return type to main, which makes the code more portable anyway.

2. Stricter rules when using templates

G++ now diagnoses even more cases of ill-formed templates which can never be instantiated (in addition to the stricter rules in GCC 7). The following example will now be diagnosed by G++ because the type of `B<T>::a` does not depend on `T` and so the function `B<T>::f` is ill-formed for every possible instantiation of the template:

```
class A { };
template <typename T> struct B {
    bool f() const { return a; }
    A a;
};
```

In member function `bool B<T>::f() const`:

```
error: cannot convert 'const A' to 'bool' in return bool f() const { return a; }
```

Ill-formed template code that has never been tested and can never be instantiated should be fixed or removed.

3. Changes to alignof result

The `alignof` operator has been changed to return the minimum alignment required by the target ABI, instead of the preferred alignment (consistent with `_Alignof` in C).

Previously the following assertions could fail on 32-bit x86 but will now pass. GCC's preferred alignment for standalone variables of type `double` or `long long` is 8 bytes, but the minimum alignment required by the ABI (and so used for non-static data members) is 4 bytes:

```
struct D { double val; };
static_assert(alignof(D) == alignof(double), "...");
struct L { long long val; };
static_assert(alignof(L) == alignof(long long), "...");
```

Code which uses `alignof` to obtain the preferred alignment can use `__alignof__` instead.

4. Associative containers check the comparison function

The associative containers (`std::map`, `std::multimap`, `std::set`, and `std::multiset`) now use static assertions to check

that their comparison functions support the necessary operations. In C++17 mode this includes enforcing that the function can be called when const-qualified:

```
struct Cmp {  
    bool operator()(int l, int r) /* not const */ { return l < r; }  
};  
std::set<int, Cmp> s;
```

In member function `bool B<T>::f() const`:

```
error: static assertion failed: comparison object must be invocable as const static_  
↪ assert(is_invocable_v<const _Compare&, const _Key&, const _Key&>,    bool f() const  
↪ { return a; } )
```

This can be fixed by adding `const` to the call operator:

```
struct Cmp {  
    bool operator()(int l, int r) const { return l < r; }  
};
```

5. The following feature has been removed: Optlib library

The OPTLIB library feature is now removed, due to the following reasons:

- It does not contain all the headers and the defines of the ANSI/ISO standard.
- Partial implementation of library functions (e.g. standard I/O functions are not all implemented)
- The math library sacrifices precision for speed/code size (not IEEE754 compliant)

Free support for Renesas RX GNU Toolchain 14.2.0.202607

For free technical support, please register at <https://llvm-gcc-renesas.com>

For your feedback and suggestions, please visit <https://llvm-gcc-renesas.com/help/contact-us/>