

## Release Notes: GCC 8.3.0.202305-GNURX

31<sup>st</sup> of May, 2023

CyberThor Studios Ltd. is releasing the GCC 8.3.0.202305-GNURX, a cross compiler tool for Renesas RX micro-controllers.

### SALIENT FEATURES

The GCC 8.3.0.202305-GNURX toolchain is based on:

- ❖ GCC 8.3.0 [released]
- ❖ Binutils 2.36.1 [released]
- ❖ Newlib 4.1.0 [released]
- ❖ GDB 7.8.2 [released]

The latest patches are applied to GCC, Binutils and GDB sources.

### ABOUT GCC 8.3.0.202305-GNURX

Release Version:	GCC 8.3.0.202305-GNURX
Release Date:	31 <sup>st</sup> of May, 2023
Platforms Supported:	64bit RedHat GNU/Linux v8.0 or later (or compatible distribution) Windows XP, Windows 7, Windows 8, Windows 10
Language:	C, C++
Targets:	RX100 RX200 RX600 RX64M RX700
Object File Format:	ELF



This section describes the fixes made in the GCC 8.3.0.202305-GNURX release.

### 1. [Improvement] Added new options

New compile options added in the current release:

-mtfu-version={v1, v2}, -mnosave-tfu

For details, please check:

<toolchain install location>\rx-elf\rx-elf\share\doc\gcc\RX-Options.html

### 2. [Improvement] Added new builtins

New builtin functions added in the current release:

\_\_builtin\_rx\_sincosfx, \_\_builtin\_rx\_atan2hypotfx, \_\_builtin\_rx\_sinx, \_\_builtin\_rx\_cosfx,  
\_\_builtin\_rx\_atan2fx, \_\_builtin\_rx\_hypotfx

For details, please check:

<toolchain install location>\rx-elf\rx-elf\share\doc\gcc\RX-Built\_002din-Functions.html

### 3. [Improvement] Added dfpu multilib

The current release of the toolchain includes a new prebuilt library with the following options:

-m64bit-doubles -mdfpu -misa=v3

### 4. [Improvement] macOS Ventura Apple M1 support(experimental)

Toolchain version built for Apple M1 was added.

### 5. [Bug fix] Wrong .data section allocation

In some cases the LMA of the .data section was wrongly placed in RAM instead of ROM.

This was fixed in the current release.

### 6. [Bug fix] Linker producing executable despite relocation errors

In some cases, despite displaying relocation errors, the linker still produced an executable.

This was fixed in the current release.

### 7. [Un-deprecation] Automatic interrupt vector entry generation

The GCC 8.3.0.202204-GNURX release deprecated automatic interrupt vector entry generation, due to concerns about how it would interfere with linker garbage collection (`--gc-sections`).

We have identified two workarounds that permit the use of `--gc-sections` in conjunction with automatic interrupt vector entry generation, which we have documented here: [https://lvm-gcc-renesas.com/wiki/index.php?title=RX\\_automatic\\_interrupt\\_vector\\_entry\\_usage](https://lvm-gcc-renesas.com/wiki/index.php?title=RX_automatic_interrupt_vector_entry_usage)

Consequently the deprecation warning has been removed, and will no longer show up when using the `interrupt((N))` function attribute.

### 8. [Change] Versioning change

The version numbering was changed from indicating Q2 or Q4 to indicate the release month.

#### INSTALLER:

The GNURX ABI (Application Binary Interface) is made available on our GNU Tools support website (<https://lvm-gcc-renesas.com>) and also provided along with Linux and Windows installer.

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:

[http://www.renesas.com/products/tools/ide/ide\\_e2studio/index.jsp](http://www.renesas.com/products/tools/ide/ide_e2studio/index.jsp)



Integration with the HEW IDE is not supported by the installer.

## KNOWN ISSUES IN GCC 8.3.0.202305-GNURX

This section describes all known issues for this particular release:

### 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` results

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:

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

#### 6. Section to segment mapping issue

In some rare cases there's an issue in the section to segment mapping which can cause load problems in the debugger.

This issue is being investigated and will be fixed in the next release.



## **FREE SUPPORT FOR GCC 8.3.0.202305-GNURX**

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/>

