© CyberThor Studios Ltd.

Release Notes: LLVM FOR RENESAS RL78 10.0.0.202203

31st of March, 2022

CyberThor Studios Ltd. is releasing the LLVM for Renesas RL78 10.0.0.202203, a cross compiler tool for Renesas RL78 micro-controllers.

SALIENT FEATURES

The LLVM for Renesas RL78 10.0.0.202203 toolchain is based on:

- LLVM 10.0.0 [released]
- Compiler-rt 10.0.0 [released]
- Libcxx 10.0.0 [released]
- Libcxx-abi 10.0.0 [released]
- Newlib 4.1.0 [released]
- GDB 7.8.2 [released]

LLVM RL78 comes with significant performance improvements (both code size and speed) compared to GCC RL78. It also comes with support for latest language standards: full support for C17 and C++17 and experimental support for the C2x (next C standard) and partial support for C++20.

The latest patches are applied to the LLVM sources.

ABOUT LLVM FOR RENESAS RL78 10.0.0.202203

Release Version:	LLVM for Renesas RL78 10.0.0.202203
Release Date:	31st of March, 2022
Platforms Supported:	Ubuntu 18.04 or later (or compatible distribution) Windows 7 or later
Language:	C, C++
Targets:	G23, G1X, I1X, D1X, LIN MCP, F1X, and L1X
Object File Format:	ELF

CHANGES IN LLVM FOR RENESAS RL78 10.0.0.202203

This toolchain is the successor of GCC RL78 toolchain, and it is meant as a direct replacement to GCC RL78.

This section describes the fixes made in the LLVM for Renesas RL78 10.0.0.202203 release.

1. **[Improvement] Updated newlib to version 4.1.0**

The newlib library included with the toolchain was updated to version 4.1.0.

2. **[Improvement] Various library and codegen optimizations**

Code speed and size optimizations were made both in the included libraries and in the compiler itself.

3. **[Improvement] Added -mfar-code command-line option**

The `-mfar-code` command-line option was added, alongside with its corresponding multilib.
This option changes the allocation of all functions, without an explicit address space, to `__far`.
Pointers to functions that have no address space specifiers will also become _far pointers.

Functions qualified explicitly or implicitly (as a result of this option) with `__far` will be placed in the `.textf` section, which needs to be present in the linkerfile that is used during linking.

This option also disables the usage of the PLT, which was used to handle calls and jumps that are allocated over the 64KB limit.

When using the option, the following considerations should be made:
- The linkerfile may require changes: there's no longer a need for the `.plt` section and the `.textf` section should be specified.
- Since `__far` function pointers are 4 bytes, any code relying on function pointers being 2 byte long will require adjustments (for example startup code that iterates over an initializer function list).
- When writing assembly code, to ensure correct linking, the `.textf` section should be declared with the correct flags, as seen below:

```
.section .textf,#alloc,#execinstr

.global _foo
.type   _foo,@function
_foo:
                    ;rest of the function
```

Currently the -mfar-code option is only supported when compiling and linking C files, C++ is not supported.

When using the -mfar-code option, function typedefs can't have address space specifiers, nor can an address space specifier applied later to them. This will be fixed in the next release.

4. **[Improvement] CC-RL compatibility has been finalized**

All relevant extensions supported by CC-RL are now supported by LLVM too. For details, please check the user manual.
To enable these extensions, use the `-frenesas-extensions` command-line option.

The `-frenesas-vaarg` option can be used to enable the promotion of `__near` pointers to `__far` pointers when they are passed as variable arguments. In the next release, the behavior of the `-frenesas-vaarg` option will be enabled with the `-frenesas-extensions` option and the option itself will be removed.

5. **[Bug Fix] Fixed -mfar-rom not affecting pointers to constants**

Using the -mfar-rom command-line option, only the global constant variables, having default address space, were qualified with `__far`. This was fixed in the current version, the option now affects both the aforementioned variables and all pointers that point to constant variables.

Support for far rom is still limited in the toolchain libraries. The following functions had their far rom equivalents added: `memchr, memcmp, memcpy, memmove, memset, strcat, strchr, strcmp, strcpy, strlen, strncat, strncmp, strncpy, strpbrk, strrchr, strspn, strstr.`
Please note that in order to call these `-mfar-rom` versions, users must include `<string.h>.`

A warning about rodata incompatibility will still be issued when trying to link the supplied libraries with the `-mfar-rom` option.

Notes:

This installer does not provide an option to integrate the LLVM RL78 toolchain with e2 studio, as the e2 studio IDE will automatically detect the LLVM RL78 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 toolchain with the HEW IDE.

KNOWN ISSUES IN LLVM FOR RENESAS RL78 10.0.0.202203

The following is a list of known issues for the tools we include for the LLVM for Renesas RL78 10.0.0.202203 toolchain:

1. **Assembly parsing issue of callt instruction operand.**

The assembly parser cannot handle any complex expressions as operands for callt instruction, only constant values are allowed at the moment.

2. **Assembly parsing of %lo16, %hi16 limited support.**

When using %lo16, %hi16 operators only 2 basic operations are currently allowed, symbol + constant and symbol - constant.
For example:
.short %lo16(_my_symbol) + %lo16(my_const)
The parse will be enhanced in future versions to support more complex expressions.

3. **Assembly parsing missing operand.**

There is currently no assembly operand to extract 8-bit data from symbols in order to be used with MOV ES, #byte, MOV CS, #byte etc. Only immediate values can be used with those instructions at the moment. The current workaround is to use %lo16, %hi16 to set the value in 16-bit register and then move the relevant part in ES/CS. A new operand will be implemented in next release.

4. **-finput-charset=/-fexec-charset= unsupported character encodings.**

Unlike GCC, Clang accepts only UTF-8 encoding (which is also the default) as valid arguments for -finput-charset and -fexec-charset options. More encoding will be added in future releases.

5. **Missing assembly listing support (-a[cdhlns] option in GNU AS).**

There's no equivalent in the LLVM for the -a[cdhlns] GNU AS option.
The alternative solution is to use llvm-objdump to obtain source code interleaved with assembly (-S, --source option).

6. **Unsigned bitfields not supported.**

Currently, there is no support for unsigned bitfields in LLVM. This will be implemented in a future release.

7. **Binding of references to packed fields.**

Creating references to struct members which are declared as packed, e.q. __attribute__ ((packed)), is incorrect as it can cause unaligned access issues. The compiler should return an error in this case however this is not currently the case. This will be fixed in a future release.

8. **Warnings generated using -mfar-rom option**.

When using -mfar-rom the linker will generate several warnings like "warning: incompatible rodata area flags...". The root cause of this issue is that there are no prebuilt libraries with this option. The same issue exists in GCC with the corresponding option, -mes0, the differences being that GCC ignores the problem silently. There are currently no known issues caused by this however this will be investigated further in next release and handled accordingly.

9. **llvm-objdump missing features**

llvm-objdump does not currently support -i and -g options supported by rl78-elf-objdump. This will be fixed in a future release.

10. **Inline assembly missing constraints and modifiers support for parameters.**

Clang supports, in case of RL78, inline assembly constructs without parameters, e.q. asm (AssembleTemplate).
The extended syntax available in GCC, asm (AssemblerTemplate : OutputOperands : InputOperands : Clobbers : GotoLabels), is not currently supported. This will be fixed in the next release, however the constraints will not be the same as in GCC.

11. **Other issues, non-specific to RL78**

LLVM tries to be a complete replacement. As such there are still a couple of missing features from GCC which will be implemented in future releases. In particular, the following issues should be noted:

lld: Information printed using --print-gc-sections is not as nice as when using the GNU ld. Bug 46783

llvm-ar: Errors when printing multiple members with the same name. Bug 42521

llvm-dwarfdump: Does not print section attribute flags yet. Bug 38488

llvm-nm: Unable to understand symbols built with gcc-lto Bug 41437

llvm-nm: Needs support for --line-numbers to llvm-nm Bug 40001

llvm-objcopy: Unknown argument '--change-section-address'. Bug 45217

llvm-objcopy: Objcopy zero-size section might cause huge binaries. Bug 46299

llvm-objdump: Prints wrong line number info for obj file compiled with -ffunction-sections. Bug 40703

llvm-objdump: Wrong behavior for non-relocatable objects when suing llvm-objdump with -r option. Bug Bug 41901
llvm-readobj: Make GNU style symbol printing invalid symbol section indexes match GNU readelf Bug 43850

llvm-readelf: Relocation addends printed style does not match GNU readelf Bug 45235

llvm-string: Short option with argument grouping not GNU compatible Bug 42942

llvm-string: Allow "-<integer>" as an alias for "-n <integer>" Bug 42964

llvm-symbolizer: Shows incorrect source line info if --gc-sections used Bug 41124

llvm-symbolizer: llvm-addr2line does not exit when passed a non-existent file Bug 42754

12. Other issues

Finally, for better understanding regarding the status of the toolchain please visit https://bugs.llvm.org/ . In particular, the following queries will help better understand the status of each tool.

https://bugs.llvm.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=ELF&product=lld&query_format=advanced&resolution=---

https://bugs.llvm.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=llvm-ar&product=tools&query_format=advanced&resolution=---

https://bugs.llvm.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=llvm-dwarfdump&product=tools&query_format=advanced&resolution=---

https://bugs.llvm.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=llvm-nm&product=tools&query_format=advanced&resolution=---

https://bugs.llvm.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=llvm-objcopy%2Fstrip&product=tools&query_format=advanced&resolution=---

https://bugs.llvm.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=llvm-objdump&product=tools&query_format=advanced&resolution=---

https://bugs.llvm.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=llvm-readobj&product=tools&query_format=advanced&resolution=---

https://bugs.llvm.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=llvm-size&product=tools&query_format=advanced&resolution=---

https://bugs.llvm.org/buglist.cgi?bug_status=NEW&bug_status=CONFIRMED&bug_status=REOPENED&component=llvm-symbolizer&product=tools&query_format=advanced&resolution=---

FREE SUPPORT FOR LLVM FOR RENESAS RL78 10.0.0.202203

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/