Release Notes: LLVM FOR RENESAS RL78 10.0.0.202107

14th of July, 2021

CyberThor Studios Ltd. is releasing the LLVM for Renesas RL78 10.0.0.202107, a cross compiler tool for Renesas RL78 micro-controllers.

SALIENT FEATURES

The LLVM for Renesas RL78 10.0.0.202107 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 3.1.0 [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 10.0.0.202107

Release Version:	LLVM for Renesas RL78 10.0.0.202107
Release Date:	14th of July, 2021
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 10.0.0.202107

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 10.0.0.202107 release.

1. [Bug Fix] Interrupt functions save/restore issue.

In some cases, registers were not correctly saved/restored if the stack was used inside of the interrupt function. This issue is now fixed.

2. [Bug Fix] 8-bit indexed load issue.

In some cases, instructions using index load operands such as [hl+b] or [hl+c] would be incorrect due to the fact that the compiler would try to use an incorrect register.
The following is an example where this bug was present. The issue is now fixed.

```
typedef struct
{
  struct
  {
    unsigned char rCount;
    unsigned char rBuffSize;
    unsigned char *receive;
  } _private
} node;

void receive(node *instance, unsigned char byte)
{
   if(instance->_private.rCount < instance->_private.rBuffSize) {
   instance->_private.receive[instance->_private.rCount++] = byte;
}
}
```

3. [Bug Fix] 8-bit comparison issue.

In some cases, 8-bit comparison instruction generation would be incorrect due to a commutativity issue.
The following is an example where this bug was present. The issue is now fixed.

```
typedef struct
{
  struct
  {
     unsigned char rCount;
  } _private
} node;

unsigned char min = 5;
bool receive_check_min(node *instance)
{
```

return instance->_private.rCount >= min;
}

4. [Bug Fix] Register bank change issue.

In some cases, the generated code was using register addresses which meant the code could execute correctly on bank 0. This was affecting the bank=RBx specification for interrupts. This issue has been fixed and the code can be executed correctly on any register bank.

5. [Improvement] Added Arithmetic peripheral unit usage for S2 core.

The previous release of LLVM RL78 toolchain did not use the S2 core specific arithmetic peripheral unit. This behavior has now changed, so when the -mcpu=S2 is used, the arithmetic peripheral is used by default. A new option, -mdisable-mda, was added to revert to previous behavior.

6. [Improvement] Generation of bit manipulation instructions.

The compiler now generates bit manipulation instructions according to the following CCRL specification:
To output bit manipulation instructions without using intrinsic functions, satisfy all conditions shown below.
(a) A constant value is assigned.
(b) The value is assigned to a single-bit bit field of the char unsigned/char signed/char_Bool type in the near area.
(c) The bit field where the value is assigned is qualified with volatile.
For a variable qualified with volatile, the compiler does not output a bit manipulation instruction when a value is assigned to the variable or when the variable is read unless the other conditions are satisfied. For a variable that is not qualified with volatile, the compiler outputs a bit manipulation instruction according to the specified optimization settings.

7. [Improvement] New formats supported in llvm-objcopy.

llvm-objcopy now also supports ihex, symbolsrec and binary formats.

8. [Improvement] New libraries supported by libgen.

rl78-elf-libgen now supports all libraries present in the toolchain. The users may specify the type of library needed by using the proper argument to the –select-lib=<lib> option. Currently supported lib names are: newlib, newlib_nano, compilerrt, libcxx, libcxxabi.

INSTALLER:

1. The LLVM FOR RENESAS RL78 10.0.0.202107 Installer onwards supports installations for all users (if administrator privileges are present), as well as installations for the current user only.
2. In both cases, the toolchain should be detected by compatible versions of e2 studio and should integrate seamlessly with them.

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 10.0.0.202107

The following is a list of known issues for the tools we include for the LLVM for Renesas RL78 10.0.0.202107 toolchain:

1. Linker Map file generation

The map file generated by LLD has less information compared to the one generated by GNU LD. This will be improved in future releases.

2. Align attribute issue for variables with automatic storage duration.

attribute((aligned(*alignment*))) is not working for varaibles with automatic storage duration when *alignment* > 2. This also affects std::experimental::simd introduced in C++17 and vectore_size attribute: __attribute__ ((vector_size (N))).
Please note there is no real use case for those on RL78 as there's are no SIMD instructions on RL78.

3. 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.

4. llvm-addr2line address interpretation issue.

The llvm-addr2line documentation states "llvm-addr2line interprets all addresses as hexadecimal and ignores an optional 0x prefix". However, this is not true 0x prefix is required (not optional). This will be fixed in a future release.

5. 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.

6. Assembly parsing issue of hex numbers followed by a dot and a number (e.g.: 0xffef2.4)

In case of bit manipulation instructions, it should be possible to specify an address followed by a dot the number of the bit to be modified. The dot is causing the parser to think this is a floating-point number instead and handles it incorrectly.
The current workaround is to use a variable/symbol and set it to the corresponding hex value (e.g.: .set saddr, 0xffef2; XOR1 CY, saddr.4). This will be fixed in next release.

7. 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.

8. **C++ debugging member access issue.**

While debugging C++ projects It is sometimes necessary to use "this" pointer in order to access member variables even in cases where there's no ambiguity and this could be implied.

9. **-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.

10. **Linker script ASSERT command issue.**

ASSERT command may fail erroneously in cases where it sections sizes are checked as it is executed before final optimizations in the linker when the code size is reduced further. This will be fixed in next release.

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

There are no listing options implemented in LLVM at the moment. This will be implemented in a future version.
Current workaround is to use llvm-objdump to obtain a source code interleaved with assembly (-S, --source option).

12. **Linker script section ordering issue.**

If the sections that have load or virtual memory address specified in the linker script are not in ascending order of the load memory address inside the linker script the link process will fail.
Please check Bug 47853 for more details.

13. **Unsigned bitfields not supported**.

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

14. **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.

15. **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.

16. **llvm-objdump does not display symbol name instead of raw address where possible.**

llvm-objdump is not displaying symbol names instead of raw address where possible. This makes the disassembly slightly harder to read than in case of rl78-elf-objdump. This will be fixed in next release.

17. **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.

18. **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.

19. **lld may report wrong overlap between .text and .bss when no load memory region specified.**

When no load memory region is specified for .text and for .bss the llld linker may report that the sections overlap although they are in different memory areas.
The workaround is to add Load memory regions for the .text and the .bss sections.

20. **llvm-size is using NOLOAD sections when determining the total size.**

Currently, the lld NOLOAD sections are contributing to the object/executable size when calling llvm-size.

21. **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

22. 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 10.0.0.202107

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/