RELEASE NOTES: GNURL78 v15.02

30th September 2015

KPIT Technologies Limited is releasing the GNURL78 v15.02, a cross compiler tool for Renesas RL78 micro-controllers.

SALIENT FEATURES:

1. The GNURL78 v15.02 toolchain is based on
   GCC 4.9.2 [snapshot dated 23rd April 2015],
   Binutils 2.24 [snapshot dated 23rd April 2015],
   Newlib 2.1.0 [snapshot dated 23rd April 2015] and
   GDB 7.8.2 [released].

2. The GNURL78 toolchain comes with code optimizations and bug-fixes.

ABOUT RL78 v15.02:

Release Version:	GNURL78 v15.02
Release Date:	30th September 2015
Platforms Supported	Red Hat GNU/Linux v8.0 or later (or compatible distribution) Windows XP, Windows 7, Windows 8, Windows 10
Language:	C, C99, C++
Targets	G1X, I1X, D1X, LIN MCP, F1X, and L1X
Object File Format	ELF

CHANGES IN THIS RELEASE:

This section describes the enhancements made and the issues fixed in this release.

GCC/Binutils:

1. The GNURL78 toolchain comes with two new code factoring optimizations:

   a. Local code factoring: Moving of identical instructions from basic blocks to their common predecessor or successor.

   Enabled using the "-frtl-lfact" option.

   b. Sequence abstraction: Finding of identical sequences of code, which can be turned into procedures and then the sequences replaced with calls to the newly created subroutine.

   Enabled using "-frtl-seqabstr" and/or "-ftree-seqabtr" options.

2. The GNURL78 toolchain generated an internal compiler error for certain test scenarios when the '__far' pointer was used.

   This issue has been fixed.

3. The GNURL78 toolchain generated an internal compiler error (ICE), "internal compiler error: in reload_cse_simplify_operands, at postreload.c:411" for all optimizations for certain test scenarios.

   This issue has been fixed.

4. The GNURL78 toolchain generated internal compiler error for the below list of builtins:
   e.g.,

```
void hoge(char* ptr)
{
    __builtin_rl78_mov1(ptr, 4, 3, 2);
    __builtin_rl78_and1(ptr, 4, 3, 2);
    __builtin_rl78_or1(ptr, 4, 3, 2);
    __builtin_rl78_xor1(ptr, 4, 3, 2);
    __builtin_rl78_rol1(4);
    __builtin_rl78_ror1(4);
}
```

   This issue has been fixed.

5. The GNURL78 toolchain generated an internal compiler error on using "m" in the inline assembler constraints for the C++ testcase with the optimization '-O1' and above:

```
// Test case: test.cpp
   void piyo(int& i)
   {
      __asm __volatile( "/* %%0 = %0 */\n" : : "m"(i) );
   }
```

This issue has been fixed.

6. The GNURL78 toolchain generated an internal compiler error on using "p" in the inline assembler constraints for the C/C++ testcase with the optimization '-O1' and above:

```
 // Test case: test.c
   void piyo(long l)
   {
      __asm __volatile( "; %%0 = %0\n" : : "p"(l) );
   }
```

This issue has been fixed.

7. The GNURL78 toolchain returned incorrect value for the __builtin_rl78_pswie() function when not in-lined or when invoked via another function using the '-fno-inline' command line option.

This issue has been fixed.

8. The GNURL78 toolchain generated incorrect code for the __builtin_rl78_set1() function when an external variable is used as the first argument for the below test case:

```
  void piyo()
    {
        extern char* ptr;
        __builtin_rl78_set1(ptr, 1);
    }
```

This issue has been fixed.

9. The GNURL78 toolchain generated large code for a C++ project due to some exception handling code being added for certain test scenarios.

This issue has been fixed.

10. The GNURL78 toolchain generated very large redundant code for increment and decrement operations for a 'long' data type variable with the '-Os' optimization option for the following testcase:

```
 /*test.c*/
   extern long l;
   void hoge()
   {
      l++;
   }
   void piyo()
   {
         l--;
   }
```

This issue has been fixed.

11. The GNURL78 compiler option, '-mes0' placed only the const data that was declared using the '__far' keyword in the .frodata section instead of placing *all* the read-only (const) data in the .frodata section i.e., even without using the '__far' keyword. All const data declared without '__far' keyword was placed by default in the .rodata section when the '-mes0' option was used.

    This issue has been fixed.

12. The GNURL78 compiler option, '-fno-zero-initialized-in-bss' did not work as expected. The zero initialized variable was placed in the .data section instead of placing in the .bss section.

    This issue has been fixed.

13. The GNURL78 toolchain saved and restored the frame pointer register inside the functions even when this register was not being used.

    This issue has been fixed.

14. The GNURL78 toolchain did not save and restore the frame pointer register inside the functions which used the frame pointer register via inline assembly code.

    This issue has been fixed.

15. The GNURL78 toolchain generated an "error: value of xx too large for field of 2 bytes at x" for an inline asm code "__asm __volatile(".space 0xfff0\n")"

    This issue has been fixed.

16. The GNURL78 Windows toolchain generated an error "cc1.exe: error: no iconv implementation, cannot convert from UTF-8 to MS932" when the option '-fexec-charset=MS932' is used.

    This issue has been fixed.

17. The GNURL78 libgcc div/mod emulation routines behaved inconsistently for char, short and long data types.

    This issue has been fixed.

18. The GNURL78 assembler did not check correctly the range of relative jumps performed by the 'br' instruction in certain scenarios.

    This issue has been fixed.

19. The GNURL78 assembler generated the following error when '0b' prefix is used to define a binary number with the '.byte' assembler directive:
    ```
    "Error: junk at end of line, first unrecognized character is `0'"
    ```

    This issue has been fixed.

20. The GNURL78 linker did not generate any warning or error message when the end of .bss section overlapped with the stack section.

    This issue has been fixed.

21. The GNURL78 objdump utility displayed incorrect disassembly for base addressing mode instructions.

    For example,
    the instruction "mov a, [de + 0]" was disassembled as "mov a, [de]", here "+ 0" was omitted by the toolchain.

    This issue has been fixed.

22. The GNURL78 objdump utility generated incorrect output for base addressing using stack pointer (SP) register.

    For example,
    the instruction "mov a, [sp + 0]" was disassembled as "mov a, [sp]", here "+0" was omitted.

    This issue has been fixed.

23. The GNURL78 objdump utility generated address instead of SFR names in the disassembly view for certain test scenarios.

    For example,
    the instruction "mov 0xffffa, a" was disassembled as "mov 0xffffa, a", here "0xffffa" should have been replaced by "psw".

    This issue has been fixed.

24. The program execution failed with the following error for a certain testcase, when combination of code statements such as malloc() and definition of "const unsigned char * " is used in an application:

    ```
    "Misalignment addr 0xf3339 val 0x3333 pc 005c0
    Program received signal SIGTRAP, Trace/breakpoint trap.
    0x000005c0 in _malloc_r ()"
    ```

    This issue has been fixed.

25. The GNURL78 simulator generated following error when project was built using pre-built optimized libraries for certain test scenarios:

    ```
    "Wild jump to 0x0 from 0x133b!"
    ```

    This issue has been fixed.

26. The GNURL78 Assembler did not check correctly the range of relative jumps performed by the 'br' instruction. The RL78 processors allowed 8-bit and 16-bit relative branches (-128/+127 or -32768/+32767 bytes from PC) but GAS seemed to calculate the distance using unsigned values and therefore allowing a forward jump of, say, 200 bytes without complaint which actually ended up as a backwards jump of 56 bytes (i.e. -56). A build-time error occurred only once the distance is too large to fit in an 8 or 16-bit value (as appropriate).

    This issue has been fixed.
    The GNURL78 assembler now generates the following errors:

    ```
    test.s:2: Error: value of 128 too large for 8-bit branch
    test.s:4: Error: value of -132 too large for 8-bit branch
    test.s:6: Error: value of 32768 too large for 16-bit branch
    test.s:8: Error: value of -32774 too large for 16-bit branch
    ```

Libraries:

1. The GNURL78 optimized library is enhanced by adding more functions such as ldexpf(), frexpf() and exp10f().

2. The project build library with Newlib source is enhanced by adding the library functions nexttoward() and nexttowardf().

3. The GNURL78 toolchain had mismatch in the values for macros such as M_LOG2E, M_INVLN2, M_LN2 and M_LOG2_E between the optimized and newlib libraries.

   This issue has been fixed.

KNOWN LIMITATIONS:

This section describes the known limitations in this release. We intend to fix these limitations in our future releases.

1. The GNURL78 toolchain does not save/restore some control registers while in the interrupt service routines. This causes corruption of result in case multiplication/division registers are used in main code as well as interrupts.

2. The GNURL78 toolchain calculates wrong higher 8/16 bits for operation hi8/hi16 respectively for below test case:

```
int main ()
{
  asm ("MOVW HL, #(%hi16(0x0EFFE2))");// FAILS => FFE2 in place of 0EFF
  asm ("MOVW AX, #(%hi8(0x0EFFE2))"); // FAILS => E2 in place of 0E
  asm ("MOVW HL, #(%lo16(0x0EFFE2))");// Work as expected   => FFE2
  return 0;
}
```

3. The GNURL78 toolchain does not handle relocation related calculations when a combination of different data types is used, for example
   `.hword %hi16(0xF0A80) + %lo16(Sym1)`

For Windows OS only:

1. The GNURL78 v14.03 Installer, "Default Installation" mode will install the toolchain to a default location "C:\Program Files\KPIT".

   The "Custom Installation" mode allows user to install toolchain at custom directory.

   User will not be prompted for IDE selection page while installing a toolchain. Instead, the e2 studio IDE will automatically detect the GNURL78 v14.03 toolchain on start-up for registration. You may also use the Toolchain Management feature in e2 studio IDE to achieve this.

2. In-order to integrate the toolchain with Renesas e2 studio version 2.0 and above, please use the Renesas toolchain management feature in e2 studio IDE ('Help' > Add Renesas Toolchains')

3. The registry entry for Windows-7 64-bit system differs to Windows-7 32-bit system.

NOTE:

WINDOWS and GNU/LINUX:

1. The optimized libraries provided along with the Newlib libraries in the toolchain do not require a separate download.

2. The optimized libraries ('liboptm.a' and 'liboptc.a') are not provided under GNU GPL. The source code of these optimized libraries is neither released nor available on request.

3. The ‘libgen’ utility is not provided under GNU GPL. The source code of the "libgen" utility is neither released nor available on request.

   For free technical support, please register at http://www.kpitgnutools.com
   For your feedback and suggestions, please visit http://www.kpitgnutools.com/feedback.php.