HighTec EDV-Systeme GmbH
Stammsitz & Entwicklung
Feldmannstraße 98
D-66119 Saarbrücken
Tel.: 0681/92613-16 Fax: -26
mailto:htc@hightec-rt.com
www.hightec-rt.com

RELEASE AND INSTALLATION NOTES

for GNU/TriCore 3.4.8

- 2011-07-14 -

Release Notes

The software contained in this distribution is mainly a port of the well-known GNU development tools to Infineon's TriCore family of μC/DSP processors; the port was done by HighTec EDV-Systeme GmbH on behalf of Infineon. The following software packages are included in this distribution:

Contents

- TriCore C/C++ Compiler, Assembler, Linker
- RLM V9.1 (License Manager)
- Eclipse-based IDE
  - DAvE Importer
  - Wizards for RAM/ROM projects
  - GUI for Linker Description File

Together, they provide a powerful development environment for the TriCore architecture, allowing to compile, run and debug programs, plus converting TriCore executables into various file formats understood by common programmers.

This release contains binaries for Windows 2000/XP/Vista and Windows 7 and Linux. It is recommended to replace any earlier TriCore Development Platform installation with the current release.

Changes

Compiler

At http://www.gnu.org/software/gcc/gcc-3.4/changes.html the GNU Changes, New Features, and Fixes of GCC are available.

The following changes and additions by HIGHTEC have been made between V3.4.8 and V3.4.7:

- Support for TriCore Aurix. One of the compiler options `-mcpu=<tc161, tc16p, tc16e, maurix>` has to be passed for enabling code generation for Aurix.
- Support for TriCore derivatives 1337, 1367, 1784, 1764, 1764, 1762 and 1783.
- If the compiler option `-fdata-sections` is enabled, the compiler will now respect the addressing modes, this means if a variable e.g. `foobar` is defined and e.g. a small addressing mode is enabled `-msmall=0` then the compiler will generate an input section `.sdata.foobar`.
- In the PCP-compiler the sign extend of bits sometimes takes the wrong destination register. This bug has been fixed.
- The compiler includes a license manager to support different license models. The licenses are located per default in the subdirectory `licenses` of the TriCore installation directory e.g. `C:\HighTec\TriCore\licenses`. This directory will be searched by the compiler for valid licenses. If you want to include a user-defined search directory you have to pass the compiler option `mlicense-dir=<directory>`. Alternatively you can set the environment variable `RLM_LICENSE` and assign as value the path and file name of the license.

Assembler

- Using binutils 2.20
- Integration of pseudo instruction for optimized handling of data page pointer loads in PCP programs.
- If extern symbols are defined in an assembler file the type e.g. data `STT_OBJECT` and its size or code `STT_FUNC` must be specified.

  Example

  ```
  # syntax: .extern <Symbol name>, <type>, <Size in Bytes>
  .extern MYFUNC, STT_FUNC, 0
  .extern MYVARIABLE, STT_OBJECT, 8
  ```

- The size `0` means unknown size

Linker

- Using binutils 2.20
- Architecture flags have been changed according to current EABI.

> **Note:**
>
> If you are using the version 3.4.8 of the TriCore Development Platform you have to relink your application.

- The memory alias functionality is provided by the keyword REGION_ALIAS. It creates an alias name 'alias' for the memory region region. This allows a flexible mapping of output sections to memory regions. If you define a memory region within your linker description file e.g. for the internal flash PMU_PLASH0 you can define a so-called memory alias REGION_ALIAS(alias, region)

  Example

  ```
  MEMORY
  {
    PMU_PFLASH0 (rx!p): org = 0x80000000, len = 4M
    PMI_SPRAM (rx!p): org = 0xc0000000, len = 32K
    DMI_LDRAM (w!xp): org = 0xd0000000, len = 128K
  }

  REGION_ALIAS("DATA_MEM", DMI_LDRAM)
  REGION_ALIAS("CODE_MEM", PMU_PFLASH0)
  REGION_ALIAS("ZDATA_MEM", DMI_LDRAM)
  ```

  This statement means that the memory region PMU_PFLASH0 can be accessed in addition by the name CODE_MEM.

  The memory alias can be used like a normal memory region within the linker description file.

  ```
    CORE(.zdata)  : ALIGN(8) FLAGS(awl)
    {
      ZDATA_BASE = . ;
     *(.zdata)
          *(.zdata.*)
      *(.gnu.linkonce.z.*)
      *(.bdata)
      *(.bdata.*)
      ZDATA_END = . ;
    } > ZDATA_MEM AT> CODE_MEM
  ```

- The new linker supports the check of section flags for input and output sections. If the section flags are not compatible then a warning will be issued.

`--warrn-flags, --no-warn-flag`
: With these two flags a warning for incompatible input and output section flags can be enabled or disabled.

`--mem-holes`
: This option enables the spreading of data or code over different output sections. The goal is to improve the performance of your application by placing frequently called code in the internal code RAM (e.g. PMI_SPRAM) and frequently used data in the internal data RAM (e.g. DMI_LDRAM). The remaining and rarely used code should be located in the flash.

  Code

  With the option `-ffunction-sections` every function can be put in a separate input section, and then spreading of code per function is possible in the linker description file. The corresponding option

for data is `-fdata-sections`. To enable the spreading over different memory regions, the linker option `-Wl,--mem-holes` must be set.

If code is spread over different memory regions, a so-called relaxing pass is required to enable long jumps. The relaxing can be enabled by the linker option `-relax`. The following snippet of the linker description file will put frequently used functions e.g. from the object file ∗mymodule.o in the internal code RAM.

```
._int_text :
 {
   *mymodule.o(.text)
  *(.text)
  *(.text*)
 } > PMI_SPRAM =0

.text :
{
  *(.text)
  *(.text*)
} > PMU_PFLASH0
```

Data

In our example, different arrays are put in the input sections .mysec. The output sections .mysec1 and .mysec2 are redirected to different memory regions MYRAM1 and MYRAM2. If the memory region MYRAM1 is full, then the next output section MYRAM2 should be used to locate the remaining data of the input sections .mysec. To enable the spreading of data over different memory regions, the linker option `-Wl,--mem-holes` must be set.

```
/* Module a.c */
#pragma section ".mysec" aw 4
int my_arr[1024];
#pragma section
...

/* Module b.c */
#pragma section ".mysec" aw 4
int b_arr[2048];
#pragma section

/* Module c.c */
#pragma section ".mysec" aw 4
int c_arr[256];
#pragma section
```

Snippet of linker description file

```
MEMORY
{
...
  MYRAM1 (w!xp): org = d0000000, len = 8K
  MYRAM2 (w!xp): org = d0004000, len = 8K
...
```

```
}

SECTIONS
{
  .mysec1 :
  {
    c.o(.mysec)
    *(.mysec)
  } > MYRAM1

  .mysec2 :
  {
    *(.mysec)
  } > MYRAM2
}
```

Note:

The order of the object files passed to the linker have an effect on the resulting locating process. If you want to avoid such dependencies, you should use the keyword SORT to sort the object files in alphabetic order.

```
SECTIONS
{
  .mysec1 :
  {
    SORT(*)(.mysec)
    SORT(*)*(.mysec)
    SORT(*)*(.mysec*)
  } > MYRAM1

  .mysec2 :
  {
    SORT(*)(.mysec)
    SORT(*)*(.mysec)
    SORT(*)*(.mysec*)
  } > MYRAM2
}
```

Binary utilities

- All binutils have been rebuilt using binutils 2.20 and bfd version 2.1.

The GNU software contained in this release is distributed under the terms and conditions described in the GNU General Public License (GPL); see the file License.txt for details. Please feel free to contact HighTec should you have any legal questions about this release.

Installation Notes

To install the TriCore Development Platform, go to the win32 directory and start the `Setup.exe` program. When the installation is complete, you should reboot your PC for the changes to take effect.

Please send all comments, questions, etc. regarding this release to:

mailto:support@hightec-rt.com
www.hightec-rt.com

If you want the debugger to connect to a TriCore evaluation board, you also need to install and activate the included JTAG driver. In order to install it on your system you must have administrator rights.

Linux

Become `root` and mount your cdrom device. Change to the directory `linux` on the CD and execute

```
./install.sh <install dir> install.tar.bz2
```

The script unpack the file `install.tar.bz2` and adds the `bin` directories of the <installdir> to your PATH variable. The changes become effective in future sessions.

To test the successful installation, go to the `examples` directory and build/run the sample programs.

Releasemanagement

The following table contains the Tool Version of each component of the TriCore Development Platform.

Components of the TriCore Development Platform

Release	**3.4.8**
Releasedate	**2011-07-01**
compiler	**3.3**
assembler	**2.2**
linker	**2.2**
addr2line	**1.0**
ar	**1.0**
nm	**1.0**
objcopy	**2.0**
objdump	**2.0**
ranlib	**2.0**
size	**1.0**
strings	**1.0**
strip	**1.0**
msys	**1.0.8**
make	**3.80**