GNU Bare-Metal Targeted Tools for ARM 32-bit Embedded Processors

ARM GCC Release 07-December-2017

Release description

Current release is based on:

- gcc 6.3.1
- binutils 2.29
- newlib 2.5.0
- gdb 7.12.1

The GCC sources are comming from Linaro, they had backported ARMv8-M and Cortex-M33 support for their own gcc-6.3 branch. Additionally this release includes backported ARMv8-R and Cortex-R52 support from gcc trunk (future gcc-8.0 branch).

This distribution provides support for newlib and newlib-nano. Freescale EWL2 libraries are distributed independently as a part of S32 Design Studio IDE. Since distinct libraries are provided, the tools require setting explicitly `--sysroot` to perform MULTILIB resolution:

For newlib

```
--sysroot={INSTALL}/arm-none-eabi/newlib
```

For EWL2

```
--sysroot={INSTALL}/arm32_ewl2
```

This distribution provides tools for 32-bit host system. For 64 bit system, 32 bit libraries are required to run the tools.

Multilib

This toolchain is built and optimized for Cortex-A/R/M bare metal development. There are 26 multilbs in this toolchain:

ARM Core	Command Line Options	multilib
		<default>
	-mthumb	thumb
Cortex-M0+	-mcpu=cortex-m0plus	thumb/v6-m
Cortex-M0 Cortex-M1	-mcpu=cortex-m0	
	-mcpu=cortex-m1	
Cortex-M3	-mcpu=cortex-m3	thumb/v7-m
Cortex-M4	-mcpu=cortex-m4	thumb/v7e-m
Cortex-M7 (No FP)	-mcpu=cortex-m7	
Cortex-M4 (Soft FP SP)	-mcpu=cortex-m4 -mfpu=fpv4-sp-d16 -mfloat-abi=softfp	thumb/v7e-m/fpv4-sp/softfp
Cortex-M4 (Hard FP SP)	-mcpu=cortex-m4 -mfpu=fpv4-sp-d16 -mfloat-abi=hard	thumb/v7e-m/fpv4-sp/hard
Cortex-M7 (Soft FP SP)	-mcpu=cortex-m7 -mfpu=fpv5-sp-d16 -mfloat-abi=softfp	thumb/v7e-m/fpv5-sp/softfp

Cortex-M7 (Hard FP SP)	-mcpu=cortex-m7 -mfpu=fpv5-sp-d16 -mfloat-abi=hard	thumb/v7e-m/fpv5-sp/hard
Cortex-M7 (Soft FP)	-mcpu=cortex-m7 -mfpu=fpv5-d16 -mfloat-abi=softfp	thumb/v7e-m/fpv5/softfp
Cortex-M7 (Hard FP)	-mcpu=cortex-m7 -mfpu=fpv5-d16 -mfloat-abi=hard	thumb/v7e-m/fpv5/hard
Cortex-R8 (No FP)	-mcpu=cortex-r8	thumb/v7-ar
Cortex-R8 (Soft FP)	-mcpu=cortex-r8 -mfpu=vfpv3-d16 -mfloat-abi=softfp	thumb/v7-ar/fpv3/softfp
Cortex-R8 (Hard FP)	-mcpu=cortex-r8 -mfpu=vfpv3-d16 -mfloat-abi=hard	thumb/v7-ar/fpv3/hard
Cortex-M23 (No FP)	-mcpu=cortex-m23	thumb/v8-m.base
Cortex-M33 (No FP)	-mcpu=cortex-m33	thumb/v8-m.main
Cortex-M33 (Soft FP SP)	-mcpu=cortex-m33 -mfpu=fpv5-sp-d16 -mfloat-abi=softfp	v8-m.main/fpv5-sp/softfp
Cortex-M33 (Hard FP SP)	-mcpu=cortex-m33 -mfpu=fpv5-sp-d16 -mfloat-abi=hard	v8-m.main/fpv5-sp/hard
Cortex-M33 (Soft FP)	-mcpu=cortex-m33 -mfpu=fpv5-d16 -mfloat-abi=softfp	v8-m.main/fpv5/softfp
Cortex-M33 (Hard FP)	-mcpu=cortex-m33 -mfpu=fpv5-d16 -mfloat-abi=hard	v8-m.main/fpv5/hard
Cortex-R52 (No FP)	-mcpu=cortex-r52	v8-r
	-mthumb -mcpu=cortex-r52	thumb/v8-r
Cortex-R52 (Soft FP)	-mcpu=cortex-r52 -mfpu=fpv5-sp-d16 -mfloat-abi=softfp	v8-r/fpv5-sp/softfp
	-mthumb -mcpu=cortex-r52 -mfpu=fpv5-sp-d16 -mfloat-abi=softfp	thumb/v8-r/fpv5-sp/softfp
Cortex-R52 (Hard FP)	-mcpu=cortex-r52 -mfpu=fpv5-sp-d16 -mfloat-abi=hard	v8-r/fpv5-sp/hard
	-mthumb -mcpu=cortex-r52 -mfpu=fpv5-sp-d16 -mfloat-abi=hard	thumb/v8-r/fpv5-sp/hard

C Libraries usage

This toolchain is released with two prebuilt C libraries based on newlib: one is the standard newlib and the other is newlib-nano for code size. To distinguish them, we rename the size optimized libraries as:

libc.a --> libc_nano.a

libg.a --> libg_nano.a

To use newlib-nano, users should provide additional gcc link time option:

`--specs=nano.specs`

Nano.specs also handles two additional gcc libraries: `libstdc++_s.a` and `libsupc++_s.a`, which are optimized for code size.

For example:

```
$ arm-none-eabi-gcc src.c --specs=nano.specs ${OTHER_OPTIONS}
```

This option can also work together with other specs options like

```
--specs=rdimon.specs
```

Please note that `--specs=nano.specs` is a linker option. Be sure to include it in linker options if compiling and linking separately.

Additional newlib-nano libraries usage

Newlib-nano is different from newlib in addition to the libraries' name. Formatted input/output of floating-point number are implemented as weak symbol. If you want to use %f, you have to pull in the symbol by explicitly specifying "-u" command option.

```
-u _scanf_float
```

```
-u _printf_float
```

Users can choose to use or not use semihosting by following instructions.

semihosting

If you need semihosting, linking like:

```
$ arm-none-eabi-gcc --specs=rdimon.specs ${OTHER_LINK_OPTIONS}
```

non-semihosting/retarget

If you are using retarget, linking like:

```
$ arm-none-eabi-gcc --specs=nosys.specs ${OTHER_LINK_OPTIONS}
```

Installing executables on Windows

Some execuatbles built for by MinGW requires additional libraries. Thus to support out-of-the-box execution libwinpthread.dll comes with toolchain. This library can be moved in appropitate place if host system requires. Or MinGW package may be installed to get executables working.

Installing executables on Linux

“32-bit compatibility libraries” should be installed just in order to run 32-bit toolchains:

in Ubuntu 14:

```
$ sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0
```

in Ubuntu 16:

```
$ sudo dpkg --add-architecture i386
$ sudo apt-get update
$ sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1
```

in Debian:

```
$ sudo apt-get install lib32z1 lib32ncurses5 lib32stdc++6
```

in CentOS:

```
$ sudo yum install glibc.i686
```

GDB with Python

Additionally if you want to use gdb python build (arm-none-eabi-gdb-py) on Linux, you need to install 32 bit python2.7.

```
$ sudo apt-get install libpython2.7:i386
```

To use gdb python build (arm-none-eabi-gdb-py.exe) on Windows, you need to install 32 bit python2.7 no matter 32 or 64 bit Windows. Please get the package from https://www.python.org/download/.

Release history

Date	Description
08-August-2017	Initial release
07-December-2017	GDB with Python