ROME
build and configuration environment

December 20, 2001

Wolfgang Reißnegger
4 Independence Way
Princeton, NJ 08540-6634

You can get the current version of this document at http://rome.sourceforge.net
For questions and comments mailto:rome-admin@lists.sourceforge.net

Contents

1 Introduction

ROME is a lightweight, modular, multiplatform embedded operating system. It is intended to run on multiple platforms and target architectures. To achieve this goal, a ROME system is built from *components.* There are four types of components:

- **Targets**
  Targets define a hardware such as a particular evaluation board, single board computer or other target hardware.
- **Modules**
  Modules implement functionality in ROME. They contain the actual source code for the device drivers, protocol stacks and applications. Modules communicate witch each other via *messages* and *events.*
- **CPU plugins**
  CPU plugins define the build toolchain for a particular CPU architecture such as the MIPS 4k or Intel x86. CPU plugins are a sub-type of *modules.*
- **Message Sets**
  Message Sets are a collection of ROME messages and events. They define the message and event codes and their structure.

This modular design makes it easy to port ROME to a different architecture or platform. Many components in ROME are target independent and do not need to be changed at all. Other components, especially driver modules, require additional information about the target system. As most modules are implemented in C, minor or no changes to the code itself are necessary, it is sufficient to provide the target architecture dependent configuration information.

ROME components are used to design and build *ROME systems* that run on a target hardware. The ROME distribution provides a graphical configuration tool, the *ROME Target Builder* (RTB) to make the design and configuration of ROME systems easier, . RTB features many configuration dialogs to manage and configure your ROME projects. It is used to create configuration files for targets, CPU plugins, modules and messagesets. It is also used to create build configuration files for user projects.

2 Directory structure

Typically, ROME development takes place on a Linux system where the developer will create and cross compile the ROME sources to get an executable binary file for the target platform. Therefore the ROME sources and configuration files are located on the host system in dedicated directories. There exist three different types of directories in the ROME build environment: The *ROME system directory*, *project directories* and *build directories.*

2.1 ROME System Directory

The ROME system directory contains a set of *ROME components* and the *toolchains* for the individual target architectures.

Components

The components are typically installed using the ROME component packages such as `RCMP-all-1.0.0-1.noarch.rpm`. The default install directory for components is:

```
/usr/local/ROME/System/OS/
```

ROME components installed in the system directory are accessible to all user projects. If the ROME Target Builder is looking for a particular components, it will first look in the user project directory. If the component can not be found there, RTB will scan the system directory. This means that components in the local project directory have priority.

Toolchains

Toolchains for individual target architectures are also provided in the ROME distribution. Typically a toolchain directory is named after the architecture. For example, the default toolchain package for the MIPS architecture is called `rtb-toolchain-mips-1.0.0-1.i386.rpm`. The install directory for toolchain packages in general is:

```
/usr/local/ROME/System/Tools/
```

and for the `rtb-toolchain-mips-1.0.0-1.i386.rpm` package:

```
/usr/local/ROME/System/Tools/MIPS/
```

For toolchains which have been built using the GNU cross-compiler, you can find the usual GNU compiler tools such as `gcc, as, ld, strip, nm` etc. Toolchains for architectures that are not supported by GNU may contain other tools.

2.2 Project Directories

This is where you will create and edit your own components for your projects. Project directories can be located anywhere in the file system. Typically you want to create a main project directory such as `~/Projects` or `~/work/ROME/Prj` and put all your project directories in there.

Within the individual project directories you can define your own components such as *targets*, *CPU plugins*, *modules* and *messagesets*. You can create sub-directories to group component types any way you want. RTB will scan the whole project sub-tree to identify ROME components.

Once you have created all your project components, you will use RTB to create one or more *build configuration files*. Build configuration files will always be created by RTB in the project directory root.

2.3 Build directories

Build directories are generated by RTB using build configuration files. They are created within the project directory under the sub-tree `Builds`. For example, if your project directory is called `~/Projects/HelloWorld` and the build configuration file is called `myHello.build`, your build directory will be created as:

```
~/Projects/HelloWorld/Builds/myHello
```

Within `myHello` RTB generates sub-directories for all the components you used and the two additional sub-directories `init` and `include`. `init` contains the initialization table for module processes and `include` hosts all exported include files as well as the generated include files `Messages.h` and `Hardware.h`.

`Messages.h` contains all message and event definitions derived from the message sets used in the project. `Hardware.h` contains the hardware specific definitions derived from the *system configuration.*

3 ROME component configuration

3.1 Configuration file types and location

System configuration information is defined in two different formats: *RTB configuration files* and *target header files.*

RTB configuration files are created using the ROME Target Builder. They can be identified by their suffix and are generally named after the component they configure. For example, the module configuration file for a module `Hello` would be called `Hello.module`.

Target header files contain declarations and definitions as they would appear in a source header file. Depending on the build configuration the content of selected target header files will be included in the file `Hardware.h`.

Configuration information for a ROME system build is defined in three different locations:

- **CPU plugin**
  The CPU plugin contains the target architecture dependent code, such as initialization, low-level interrupt dispatching and context switching. The CPU plugin also defines target system constants such as the size of an integer and other information.
- **Target**
  The target contains all information that is necessary to build a system for a specific hardware (e.g. a specific board). It contains information about memory layout, IO spaces etc. It also defines the *CPU plugin* which is used to build ROME system for this target.
- **Module specific target configuration**
  Modules can contain target specific information. For example, the driver module for a serial chipset contains information about the memory base address of its registers and the register spacing on that particular target. Other modules, such as higher level application modules do not necessarily need this information and compile and run on any target.

3.2 CPU plugin

CPU plugins are special modules (see below). Additionally to the information provided by modules they provide information about the toolchain and the default assembler, compiler and linker flags for the particular architecture. These configuration options are provided in the RTB module configuration dialog.

In any other respect the CPU plugin configuration is identical to the module configuration. This means that the CPU plugin can also contain a `Targets` directory.

3.3 Target configuration

The target directory contains the RTB configuration file which is named after the target (`P4032.target`) and a set of target header files. The file `P4032.target` defines the required CPU plugin for the target. In some cases the target requires compiler, assembler or linker flags that differ from the definition in the CPU plugin. Therefore, they can be overridden in the target configuration.

The target header files contain the target board specific configuration such as memory layout, IO spaces. It is possible to create multiple target header files for a single target which represent different configurations. When building a ROME system using RTB, one or more of these files can be selected to be included in the build. In the case of the P4032 there are two files for different memory configurations:

```
cfg-16mb.h
cfg-8mb.h
```

3.4 Module configuration

The module configuration file is created by RTB. It contains information about *processes, exported header files* and *module options.* The information about the module processes will be used for creating the process initialization table in the build tree `init` directory. Exported module header files will linked into the build `include` directory and be visible for other modules. The module options can be selected in the build configuration dialog and will be included in the `Hardware.h` file.

`Targets` directory

The module directory contains a sub-directory named `Targets`. Here you can find all target specific configuration header files. For every target that is supported by the module, there exists one target header file which is named after the target. For example, if the module has support for the P4032 target, it should contain a file named:

```
P4032.h
```

The content of this file will be added to the `Hardware.h` file if a system is build for the P4032.