ROME Target Builder (RTB)
USER MANUAL

May 1, 2001

Wolfgang Reißnegger
Distributed Systems Software Group
C&C Research Laboratories
NEC USA, Inc.
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

List of Figures

1 Introduction

1.1 What is ROME?

ROME stands for *Research Operating System for Multimedia Engineering*. ROME has been developed at NEC's Computer and Communications Research Laboratory (C&CRL) in Princeton, New Jersey , as an effort to design and implement an operating system as a basic platform for multimedia research applications.

ROME is an operating system that has been designed to manage high speed data streams within a multimedia environment. The system is highly modular, with functionality split between multiple processes. To ensure a high throughput with minimal overhead ROME provides a *zero copy architecture* where pointer references to data are passed around instead of data being copied. The goal of this approach is to maximize the utilization of a given hardware configuration making it possible to:

- Create high performance systems with much higher data throughput than conventional systems;
- Build fast applications for embedded, mobile systems with lower performance CPUs.

Considering the latter point — especially in a world with more and more mobile computing devices, a fast operating system able fully to exploit the available hardware is essential for reasonable performance. ROME is designed to serve this purpose.

1.2 So why do I need RTB?

Over time ROME has evolved away from the research state to a very stable and efficient micro-kernel OS. Many components have been added over time such as:

- Hardware drivers for Ethernet, SCSI, graphics chipsets and input devices;
- Toolkits for Audio/Video support;
- User windowing system.

As more and more components have been added to ROME, it has gotten more versatile but also has gotten more complex to design, build and configure a ROME system. RTB solves this problem by providing a GUI that abstracts ROME design, build and configuration procedures into an easy to use user interface. This reduces the learning curve and speeds up ROME software development.

With RTB you can easily create and manage ROME projects. You can create and add new components, using them together with existing components that are located on a CVS server. You can change ROME system configurations and play with different target system settings. All this functionality is integrated into RTB.

1.3 Where do I get RTB?

RTB is distributed in a `.tar.gz` archive format. You can get the package from:

`http://rome.sourceforge.net`

Browse the *Download* section of the site and look for the *RTB* package.

The source code for RTB is also available. You can obtain the source code via CVS. See the *CVS* section of the site for more information.

1.4 More useful information

Look at the *Getting Started Guide* for ROME to get more information on how to download, install and run a ROME development environment. You can get the guide at `http://rome.sourceforge.net/` in the *Documentation* section.

2 Overview

An overview of the RTB architecture is shown in Figure 1. RTB accesses both CVS repositories as well as the local disk for existing ROME project components such as *Modules*, *Targets* and *MessageSets*. These components will be described later in the document.

RTB does not integrate all the functionality that is needed to create a ROME system. For now, the code generation and code download happens through external processes. RTB takes care of project management and configuration and creating the necessary build files.

2.1 Project Build

Once a project is defined and configured, RTB creates the necessary build files such as the Makefiles, certain Header files and other configuration files that are needed to compile the code.

2.2 Directories

RTB creates and relies on a certain directory structure for keeping project information. Assuming `/home/rtbuser/rtb` as the root directory, RTB will use the following setup:

```
/home/rtbuser/rtb/HelloWorld
/home/rtbuser/rtb/HelloWorld/Modules
/home/rtbuser/rtb/HelloWorld/Targets
/home/rtbuser/rtb/HelloWorld/MessageSets
```

When executing the build option within RTB all the Makefiles will be created within the `Modules` directory. To build the ROME target executable, you will have to run `make` within that directory.

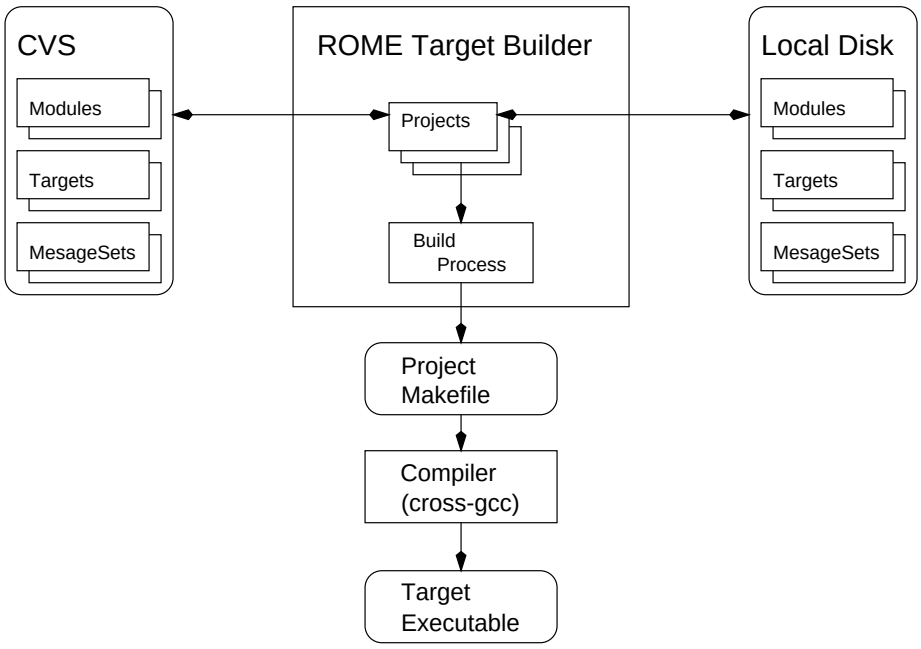

Figure 1: RTB architecture overview

2.3 Configuration file suffix

RTB stores all its information in specific configuration files. All RTB configuration file use the same syntax and have the suffix `.rtb`. Although these files only contain ASCII text and are easily readable using a standard editor, it is not recommended to change this files manually. A simple typing error could corrupt a configuration file, rendering it useless. As RTB frequently reads and *writes* configuration files, it might be possible that information is lost if configuration files have been corrupted.

2.4 User interface look and feel

Many of RTB features can be invoked using a **toolbar button**, the **toolbar menu** or the **context sensitive popup menu** (Figure 2) which can be brought up by clicking the **right mouse button**. It is also possible to use **hotkeys** which activate certain functions. The hotkey combination for a specific function is displayed at the right side of that function in the popup menu.

3 Initial setup

3.1 Starting RTB for the first time

When you start RTB for the first time, it will create the directory `.rtbrc` in your home directory (Figure 3). This directory is used to store RTB configuration data such as the *project list*, *preferences* and the *CVS repository list*. The files in this directory should never be changed manually as this can result in data loss or cause RTB to stop working.

Also, the *project list* file and the *CVS repository list* file will be created in this directory during the first start.

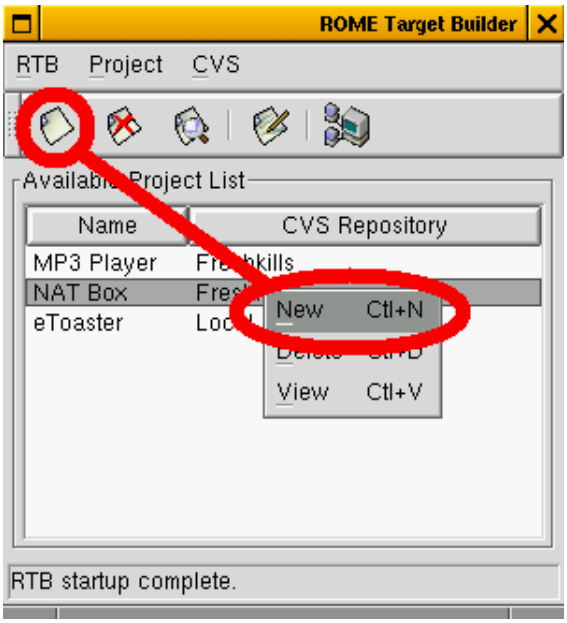

Figure 2: Activating functions in RTB

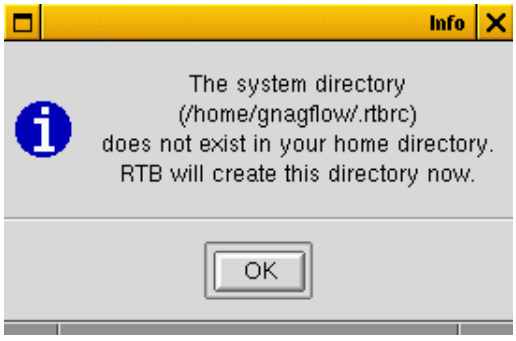

Figure 3: First startup of RTB should bring up this window

3.2 Basic configuration settings

At first startup RTB also initializes the configuration settings to default values. Usually these values work well, however, it is recommended that you configure RTB to your specific needs using the preferences dialog (Figure 4). To configure RTB use the **File->Preferences** menu or press the **preferences button** in the toolbar.

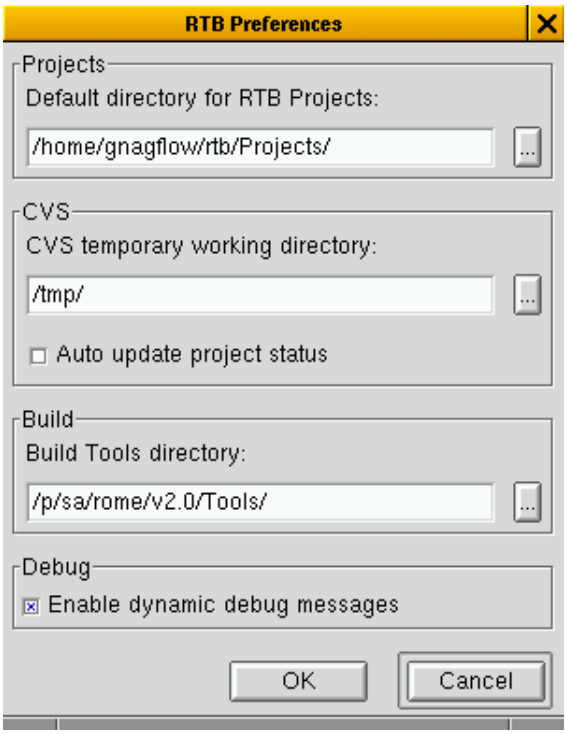

Figure 4: RTB preferences dialog

3.2.1 Projects directory

In the Projects tab you can define the directory where you want to keep all your ROME project files. Usually it's a good idea to create a dedicated RTB directory in your `$HOME` directory (e.g. `/home/gnagflow/rtb`).

3.2.2 CVS directory

You can define the temporary directory that CVS will use during your RTB session. The directory must be read/writable by you and should have at least 5 MB of available file space. Unless you have security issues, `/tmp` is a good place.

3.2.3 Auto update project status

Enable this checkbox, if you like RTB to update the CVS status of the project entries automatically once you open the project window. This way you will not have to update the status manually. However, you

should only use this option if you have a fast connection to the CVS server. If you use e.g. a modem dialup connection, you should not use auto update.

3.2.4 Build Tools directory

This entry specifies the root for the compiler tool chain utilities. Note that this is *not* the directory where you keep your tool chain executables (e.g. gcc). The path to the executables will be created using the CPU type and class you specify in the target description (see chapter).

3.2.5 Debug messages

With the “enable dynamic debug messages” switch you can define whether or not RTB should print debug messages on the standard output. This might be useful in case you encounter a problem with RTB and want to submit a bug report. However, enabling this option might clutter up your terminal window, so use it with caution.

4 RTB Main Window

Figure 5 shows the RTB application main window. The main window appears directly after RTB has been started. It shows a list of existing projects on your system. You can use the *Project menu* to *create*, *edit* or *delete* projects. As an example, the project “NAT Box” is used throughout this document. All mentioned

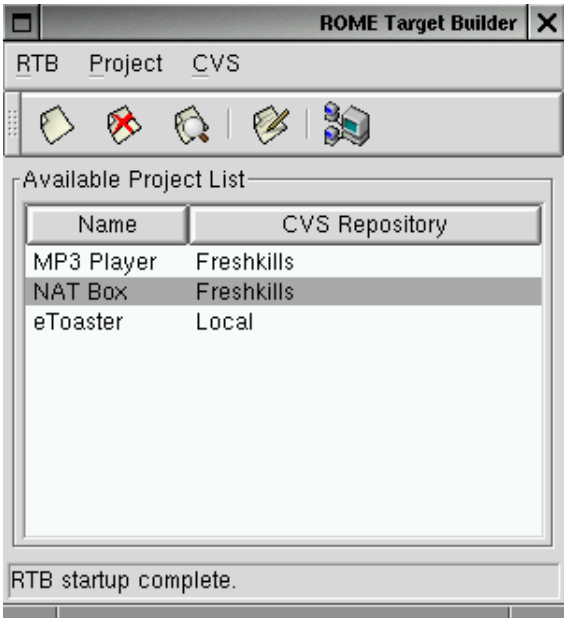

Figure 5: RTB main window

paths, directories and filenames refer to this particular example project.

4.1 Creating a new project

To create a new project, you need an existing CVS Repository. A project will always be associated with a CVS repository, and this relation can not be changed after the project has been created. So you should be sure to choose the right CVS repository right from the beginning. If you do not have defined a CVS repository at this point, see Chapter 9.1.1 on page 32 on how to add a new CVS repository to RTB.

To create a new project select the in the **Project->New** menu. A dialog (Figure 6) will pop up and you can enter the *name*, *working directory* and *CVS Directory* for the new project. Once you hit OK, the new project entry will appear in the project list and you can start editing the new project.

The project list will be automatically saved to disk.

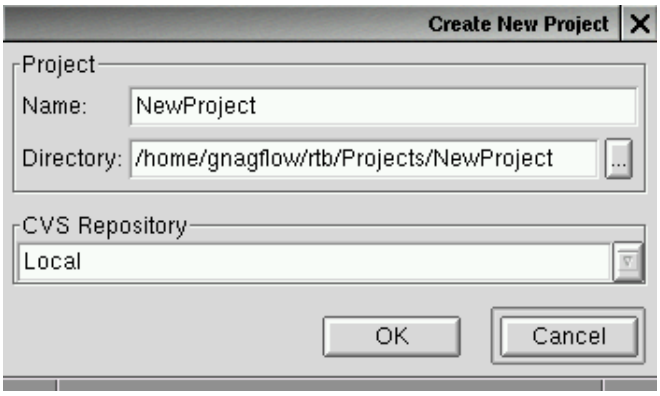

Figure 6: Creating a new project in RTB

NOTE: The directory entry will automatically expand using the default directory from your RTB preferences and the entered project name. However, you can change the directory according to your needs.

4.2 Editing a project

To edit a project, select it from the list and click on the **Project->View/Edit** menu. The project window for the selected entry will appear on the screen. In this window you can edit the properties of the project. See chapter 5.5 for how to do this.

4.3 Deleting a project

If you want to delete a project select the **Project->Delete** entry from the menu. You will be asked to confirm your intention. If you click YES, the project will be deleted from the project list.

Note: The project does still exist in the project directory. That means that the files are still available and are not deleted by RTB. This is a feature that should prevent the accidental loss of modified source files.

5 Projects

5.1 Project root directory

A project is always associated with a particular directory on your hard drive. The path of this directory is constructed using the default project directory as defined in the RTB preferences dialog and the project name. For example, if your default project directory is `/home/gnagflow/rtb/Projects`, our "NAT Box" project directory will be `/home/gnagflow/rtb/Projects/NAT_Box`. All non-alphanumeric characters of the project name will be converted to the underscore (_) character.

5.2 Project entry types

RTB (or ROME, depending on your view) defines three different project entry types:

- Targets

  A target contains all information that is machine dependent. Usually you create a target for a very specific hardware such as a single board, an embedded system or a specific configuration of an evaluation system.

- Message Sets

  A message set defines a set of ROME messages and events for inter process communication in a ROME system.

- Modules

  Modules are the basic building blocks of a ROME system. They contain the actual source code.

5.3 Project directory structure

Within the project directory the following directory structure is created to accommodate the project components like Targets, Message Sets and Modules. For the "NAT Box" example the following directories will be created:

```
/home/gnagflow/rtb/Projects/NAT_Box
/home/gnagflow/rtb/Projects/NAT_Box/Targets
/home/gnagflow/rtb/Projects/NAT_Box/Targets/DDB-VR4111U-eval-
board
/home/gnagflow/rtb/Projects/NAT_Box/MessageSets
/home/gnagflow/rtb/Projects/NAT_Box/MessageSets/Standard
/home/gnagflow/rtb/Projects/NAT_Box/MessageSets/Alarm
/home/gnagflow/rtb/Projects/NAT_Box/Modules
/home/gnagflow/rtb/Projects/NAT_Box/Modules/Clib
/home/gnagflow/rtb/Projects/NAT_Box/Modules/NAT_Engine
/home/gnagflow/rtb/Projects/NAT_Box/Modules/NetMon
```

This structure resembles the directory structure as it is created on the CVS server.

5.4 Project entry status

RTB defines a set of statuses for project entries. The status of an entry can be one or more of the following:

- unknown

  The status of the entry is currently unknown. This is usually the case, if an entry has been newly added or if RTB did not yet receive the entry's CVS status from the CVS server.

- up to date

  The entry is consistent with the entry on the CVS server.

- local only

  The entry only exists as a local copy. It has not been yet checked into CVS.

- locally modified

  The entry has been modified locally. This can either mean, that source files of a module have been modified or properties of the entry itself.

- files added

  This status only applies to modules. New files have been added to the module and have not been checked into CVS yet.

- files deleted

  This status only applies to modules. Files have been locally deleted from the module and the module has not been updated in CVS yet.

- internal error

  This status should never occur. It means that RTB can not make sense of an entry. If this status occurs, you should delete the entry and restart RTB.

5.5 Project Window

Each project in RTB can be displayed in its own project window (Figure 7). There is no limit in how many project windows can be opened at one time, except the machine's resources and the user's capability of keeping control. The project window consists of four parts:

- Menubar

  The menubar provides functions to manage the project.

- Toolbar

  Many functions of the menu are duplicated here for ease of use. Most frequently used functions can be invoked using a single button click.

- Project basic setting

  The name, source path and CVS repository are shown here for fast project navigation when using multiple project windows.

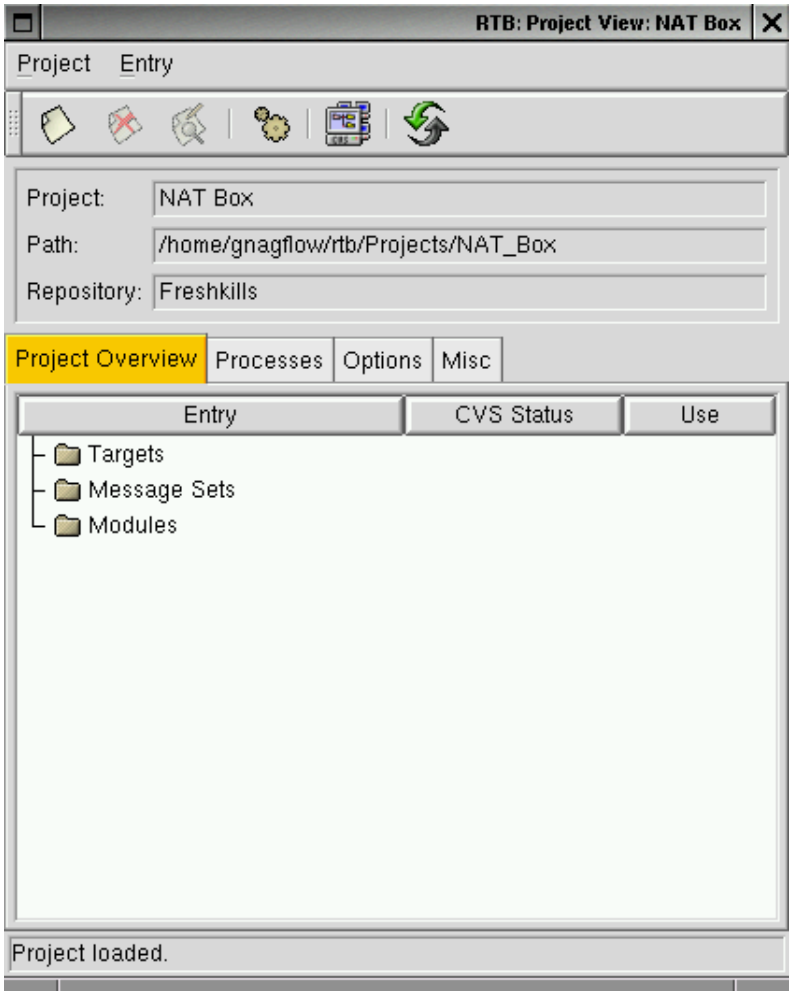

Figure 7: Project Window

- Detailed information notebook (tabs)

  Here more detailed information about the project is displayed:

  - A tree showing the project entries;
  - A list of processes defined in the project;
  - A list of options used for the project;
  - Whether or not the project target will contain debug information.

5.6 Project Overview Tab

The "Project Overview" tab (Figure 8) shows a tree containing all the targets, message sets and modules that are defined in the project. It displays the most important attributes of these entries in the following

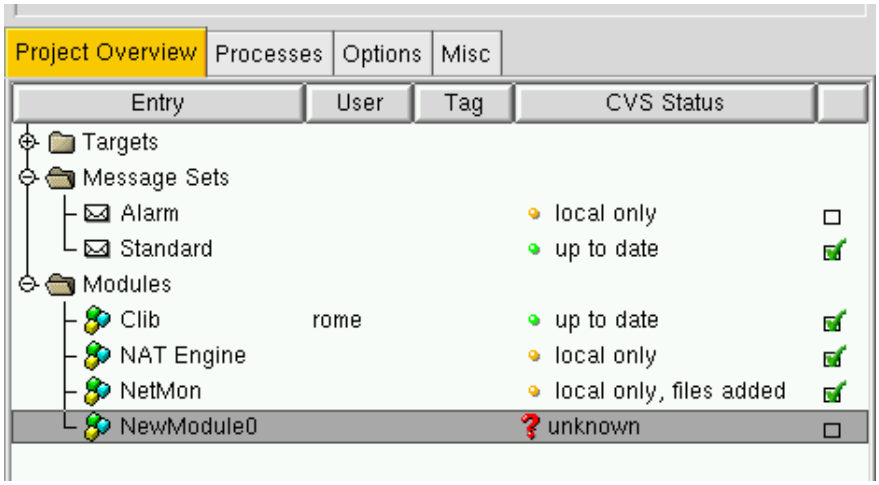

Figure 8: Project Overview tab

columns:

- Name

  The *name* of the entry.

- User

  Shows the name of the *user* who has currently *locked* this entry, if any.

- Tag

  The name of the *tag*, if the entry has been checked out using a tag definition.

- Status

  The current *status* of the entry. This can be a combination of one or more statuses as described in section 5.4.

- Checkbox

  The checkbox indicates, whether or not the entry is used in the current project.

5.6.1 Adding entries to a project

To add an entry to a project you have two options:

- Creating a new entry

  To create a new entry, select the category in the tree you want to create in (target, message set, module). Press the **New button** in the toolbar or use the **Entry->New** menu entry. When using the menu entry it is not necessary to select the category in the tree, as you can select it from the menu.

- Importing an entry

  To import an entry from your local filesystem, use the **Entry->Import->...** menu entry. A filedialog will ask you for the location of the `.rtb` configuration file of the entry you want to import. Select a file from the list and the entry will be added to your local entry list. An imported entry will be local and not associated with CVS. You will have to check in the entry if you want to manage it with CVS.

- Checking out an existing entry from CVS

  To check out an entry from CVS, press the **Repository Manager button** in the toolbar or select the **Entry->CVS check out** menu entry. This will bring up the CVS Repository Manager window where you can select the entries you want to check out. See chapter 9.3 on page 35 on how to do this.

When checking out an entry from CVS, it will be active by default (see Section 5.6.4) and therefore automatically included in the next build.

5.6.2 Editing entries of a project

If you want to edit an entry, select the entry from the list and press the **View/Edit button** in the toolbar or select the **Entry->View/Edit** menu entry. Depending on the type of the selected entry, the appropriate view/edit dialog for the entry will appear. Edit the values for the entry and press OK to close the view/edit dialog.

5.6.3 Deleting entries from a project

To delete an entry from a project, click the **delete button** in the toolbar or select **Entry->Delete** from the menu. RTB will ask you to confirm the deletion of the entry.

Note: The entry will be deleted from the project entry list, however, it will not be deleted from the disk. This is a safety feature to prevent entries from being deleted accidently.

5.6.4 Defining active entries

When an entry appears in the entry list, it does not necessarily mean, that it also used to build the project. You can select the entries that will be used for building the target system by selecting the checkboxes at the

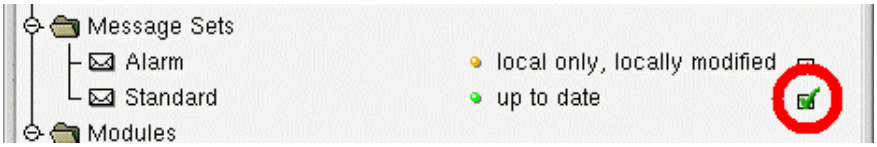

Figure 9: Activated project entry

right side of the entry. A green checker will appear to indicate that the entry will be used when building the system (Figure 9). Checked out entries will be active, newly added entries will be inactive by default.

This is useful if you want to select between similar, but slightly different, modules or target definitions for building a system. You can include both targets into your project and select either one of them to test out the effects. The same is, of course, true for modules and message sets.

5.7 Processes Tab

This tab (Figure 10) shows a list of currently defined processes which will run in the modules contained target system. It only shows processes that are in *active* modules, it will not show processes of *inactive* modules.

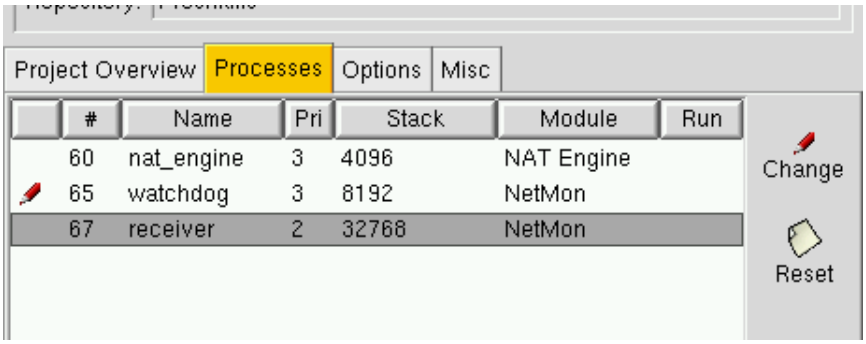

Figure 10: Project Processes Tab

When a module containing processes is added to the project and activated, its processes will appear in the process list with their default settings as they are defined in the module.

The list will be sorted by the order in which the `init` functions of the processes will be called during initialization of ROME.

5.7.1 Changing Process settings

You can change the process setting on a per project basis. Select the process in the list and press the **Change** button or **double click** the process entry. The process edit window (see Figure 24) will appear and you can change the settings for the process. The changes will only affect *the current project* and will not transfer to any other project.

A changed process will be indicated by a little pencil on the left side of the process entry.

5.7.2 Reverting changes

If you want to *reset* the values of a process to its default settings (as defined in the module where the process is defined), select the process from the list and press the **Reset** button.

5.8 Options Tab

The "Options" tab (Figure 11) shows a list of all available options in all *active* modules in the project. You can select whether or not an option is activated. Active options will generate a `#define` in the `Hardware.h` include file. This way you can pass definitions to source files without changing the source code itself.

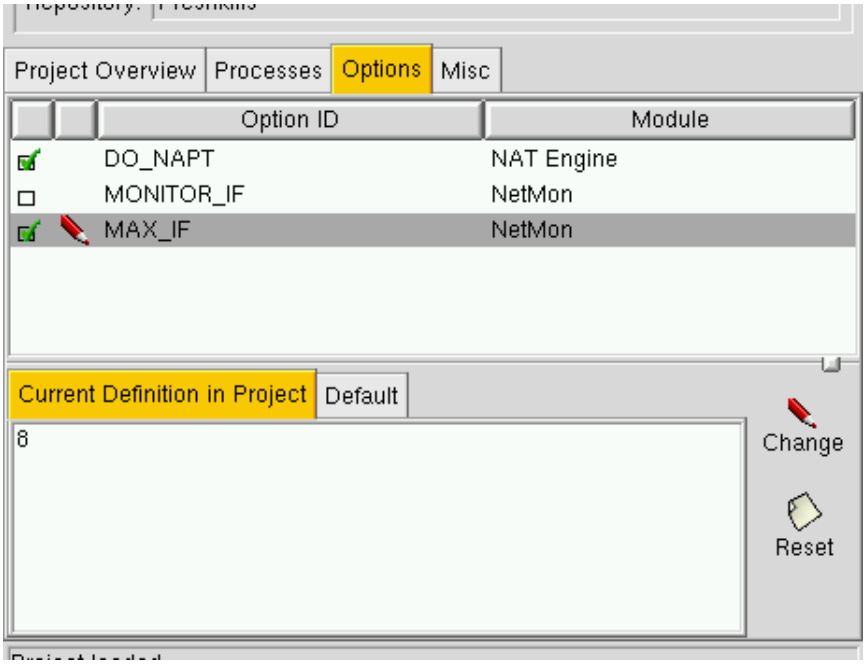

Figure 11: Project Options Tab

5.8.1 Changing Option values

You can change the value of an Option by entering the new value in the 'Current Definition in Project' textfield and pressing the 'Change' button. The Option value will then be changed *for this project only*. In other projects the changes here will have no effect. A changed Option value will be indicated by a little pencil near the Option ID.

5.8.2 Option Default values

You can check the default value of the Option by selecting the 'Default' tab. If you want to revert the Option value for this project to its original default setting, press the 'Reset' button.

5.9 Misc Tab

As the name might let you expect, the “Misc” tab (Figure 12) show miscellaneous features, options and selections for the project. Currently there is only one checkbox which lets you decide whether or not a symbol table for the ROME debugger should be generated and included into the target. Of course, this only makes sense, if you have implemented a debugger (or another module that needs the symbol table) for your target.

Figure 12: Project Misc Tab

6 Targets

Targets describe the configuration of specific target hardware setups. Therefore, a target contains all information that is machine dependent. Usually one creates a target for a very specific hardware such as a single board, an embedded system or a specific configuration of an evaluation system. The target contains the following information:

- CPU type and class (e.g. MIPS r4300);
- Compiler/Linker settings such as compiler/linker flags, warning/optimization level and debug info flag;
- Definitions for the `make` process;
- Definitions for a specific install task;
- Hardware definitions like memory layout, IO spaces etc.

Most of the time you *can not* re-use a target for another system, however, you might be lucky if the systems are closely related.

6.1 Target configuration file

In our example, the target configuration file will be stored as

`/home/gnagflow/rtb/Projects/NAT_Box/Targets/DDB-VR4111U-eval-board/DDB-VR4111U-eval-board.rtb`

At first, it seems a bit odd, that the name of the target actually appears *twice* in the path, as a directory name and as the filename itself. However, this naming makes CVS operations much more intuitive and easier.

6.2 Target dialog

You can open the target dialog (Figure 13) by selecting a target from the list in the project window and clicking the **Edit button** or selecting the **Entry->View/Edit** menu entry. This dialog provides textfields

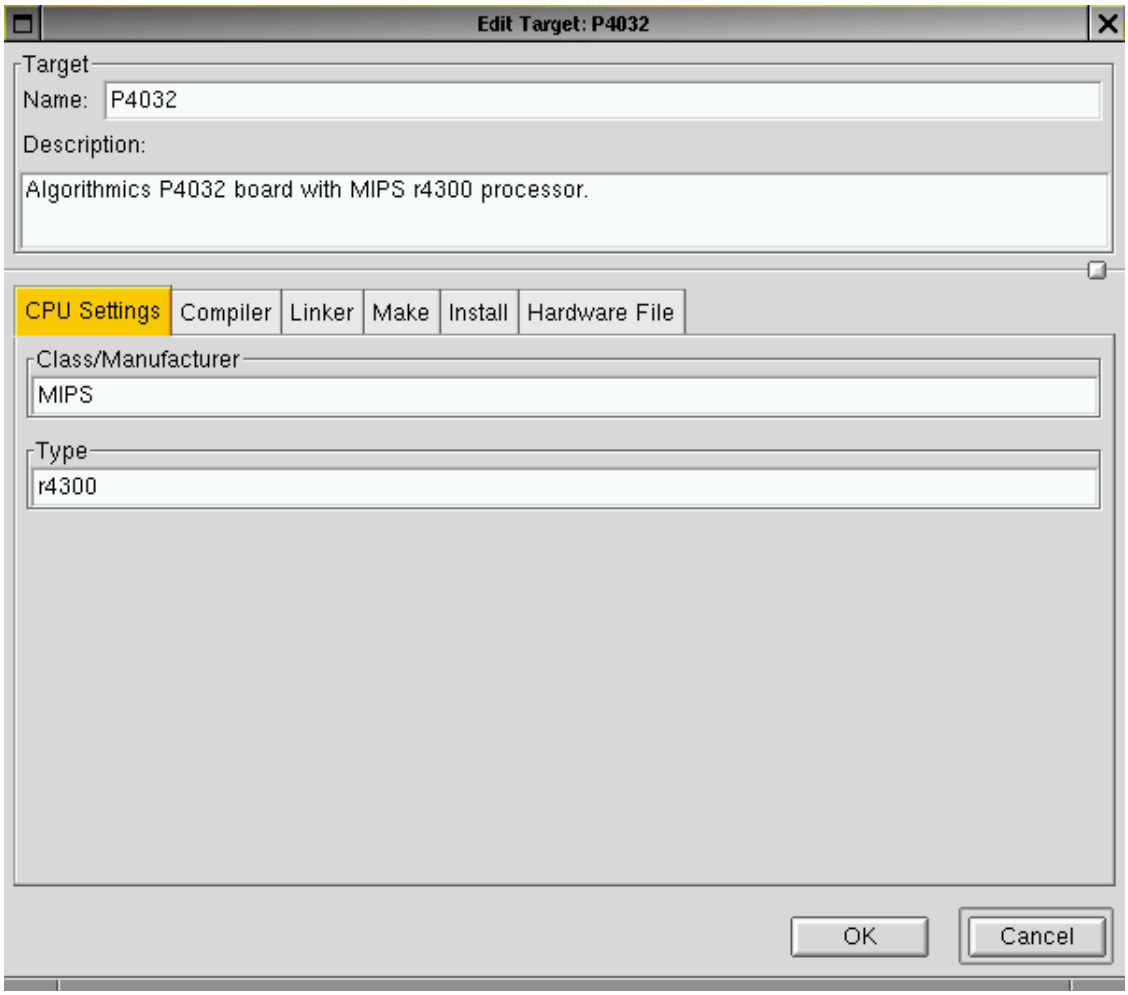

Figure 13: Target Dialog

to enter a name and description for the target as well as a set of notebook tabs to enter more specific information such as:

- CPU class and type
- Compiler and Linker settings
- Makefile definitions
- Install definitions (such as scripts)
- Hardware configuration

6.2.1 CPU class and type

The CPU class and type define, together with your toolchain base path setting in the preferences dialog, the path to the toolchain for a specific processor. If, for example, your toolchain base path is `/p/sa/rome/v2.0/Tools`, your CPU class is MIPS and the CPU type is set to r4111, your toolchain path will expand to:

```
/p/sa/rome/v2.0/Tools/MIPS/r4111
```

When RTB creates `Makefiles` for the target build, it will include this path. All toolchain executables must be located in this directory in order to run a successful `make`.

6.2.2 Compiler and Linker settings

In the "Compiler/Linker" tab (Figure 14) you can define the compiler's behaviour when compiling your source files. You can enter the following information:

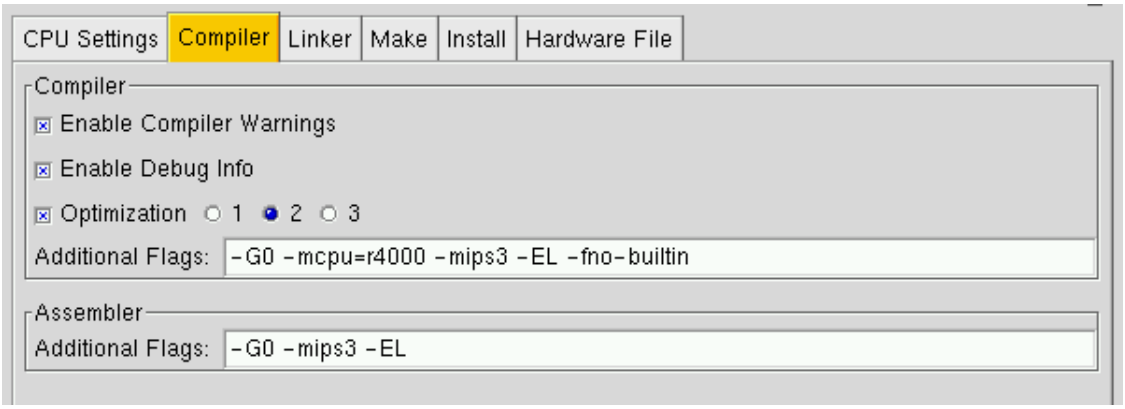

Figure 14: Target Dialog Compiler Settings

- Enable Compiler Warnings

  This checkbox defines, whether or not the compiler will generate warning messages when compiler your code. If you enable the checkbox it will add the "`-Wall`" flag to the list of compiler flags.

- Enable Debug Info

  Enabling this checkbox is equivalent to adding the "`-g`" flag to the compiler flags.

- Optimization

  This line controls the amount of compiler code optimization. The settings add the "`-O1`", "`-O2`" or "`-O3`" compiler flags, respectively.

- Additional Compiler Flags

  In this textfield you can enter more specific compiler flags, which will be added to the compiler flag list.

- Additional Assembler Flags

  These flags will be added to the assembler's flag list.

- Linker Flags

  Enter addition linker flags here.

- Input File

  RTB generates an input file for the linker. The input file contains all information the linker needs to know such as output file format, memory layout of the target executable and defines certain labels such as `_bss` or `_end`. The correctness of this entry is crucial for creating a working executable. If you are not familiar with the linker input file syntax or semantic, you should not change this entry. If you want to create a new target for a processor class for which an input file definition already exists, it *might* work in some cases to copy this entry. Refer to the linker documentation, if you have any problems.

Enter or change the values and press OK to update the target settings.

6.2.3 Makefile definitions

In the "Make" tab (Figure 15) you can define the make rules for compiling your source files. As usually

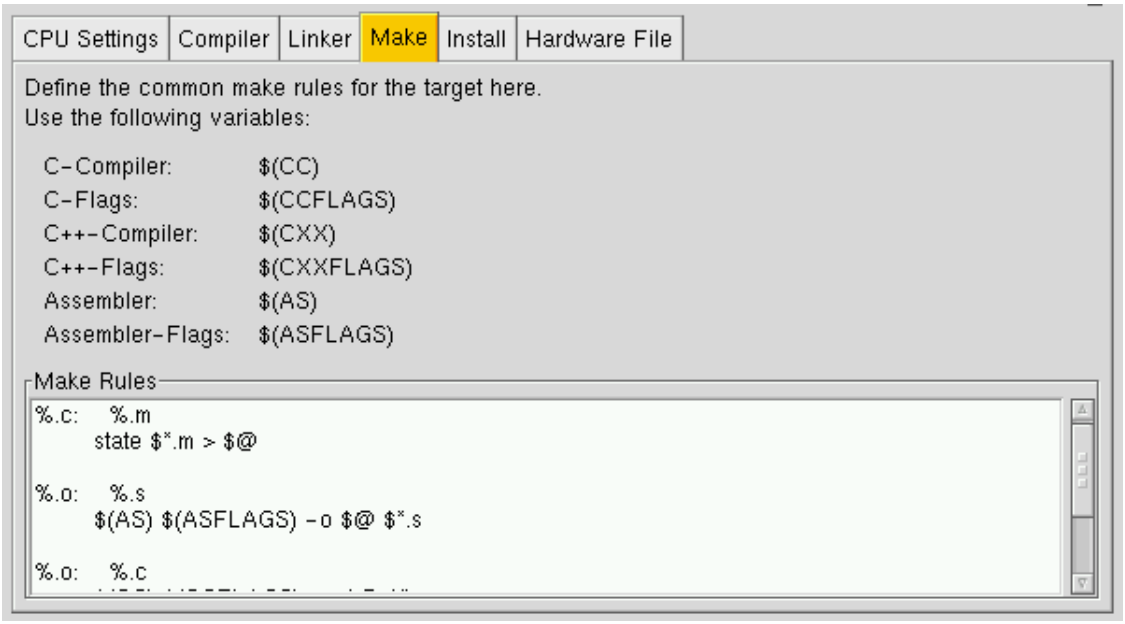

Figure 15: Target Dialog Make tab

all ROME source files are written in Assembler, C or C++, these rules are always the same and can be re-used from project to project. however, you might want to add certain rules, if you, for example, have a specific input language which has to be pre-processed or compiled by a specific tool, you can add the build rule for these file here.

6.2.4 Install definitions

The “Install” tab shown in Figure 16 allows you to enter commands for installing the executable on the target. These commands will be added to the `install:` section of the `Makefile`. So you can execute

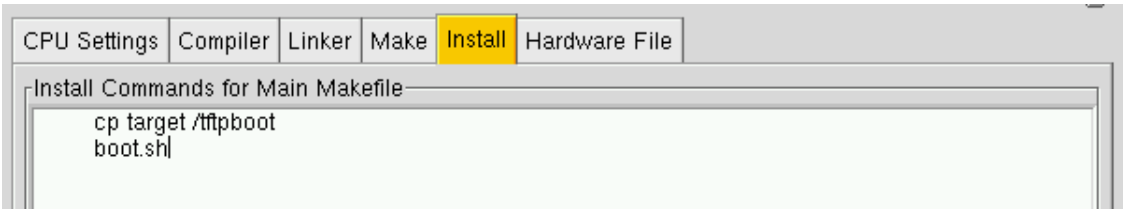

Figure 16: Target Dialog Install tab

these commands with the `make install` command line in your shell.

6.2.5 Hardware definitions

Figure 17 shows the “Hardware File” tab. Here you can add the hardware definitions for your target. These settings are *very* specific. You will have to look up your target hardware specifications in order to

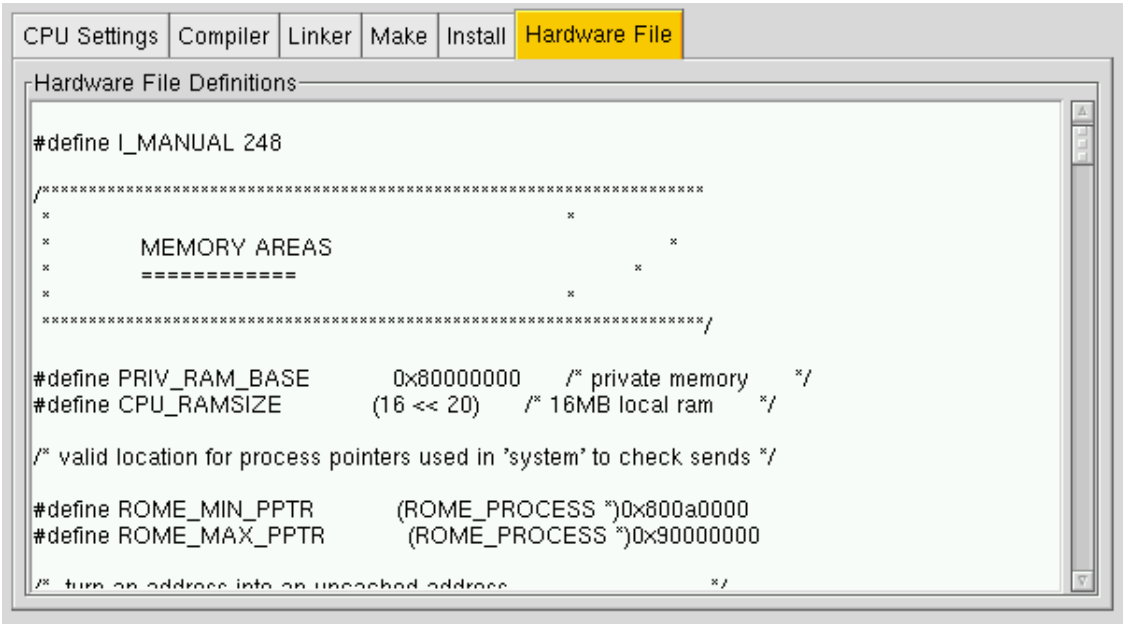

Figure 17: Target Dialog Hardware File tab

know which values to enter here.

7 Message Sets

A message set defines, as expected, a set of ROME messages and events. Although it is called a message set, it actually contains definitions for both messages *and* events. These messages and events are used for

inter process communication of modules in a ROME system. Some message sets, like “Standard”, are required for almost any conceivable system. Other message sets are only used, if specific functions are needed (e.g. timer functionality).

7.1 Message Set configuration file

In our example, the message set configuration file for the “Standard” message set will be stored as

`/home/gnagflow/rtb/Projects/NAT_Box/MessageSets/Standard/Standard.rtb`

As mentioned for target configuration files, the path naming is defined by CVS.

7.2 Message Set Dialog

To open the message set Dialog (Figure 18), select a message set from the list in the project window and click the **Edit button** or select the **Entry-View/Edit** menu entry.

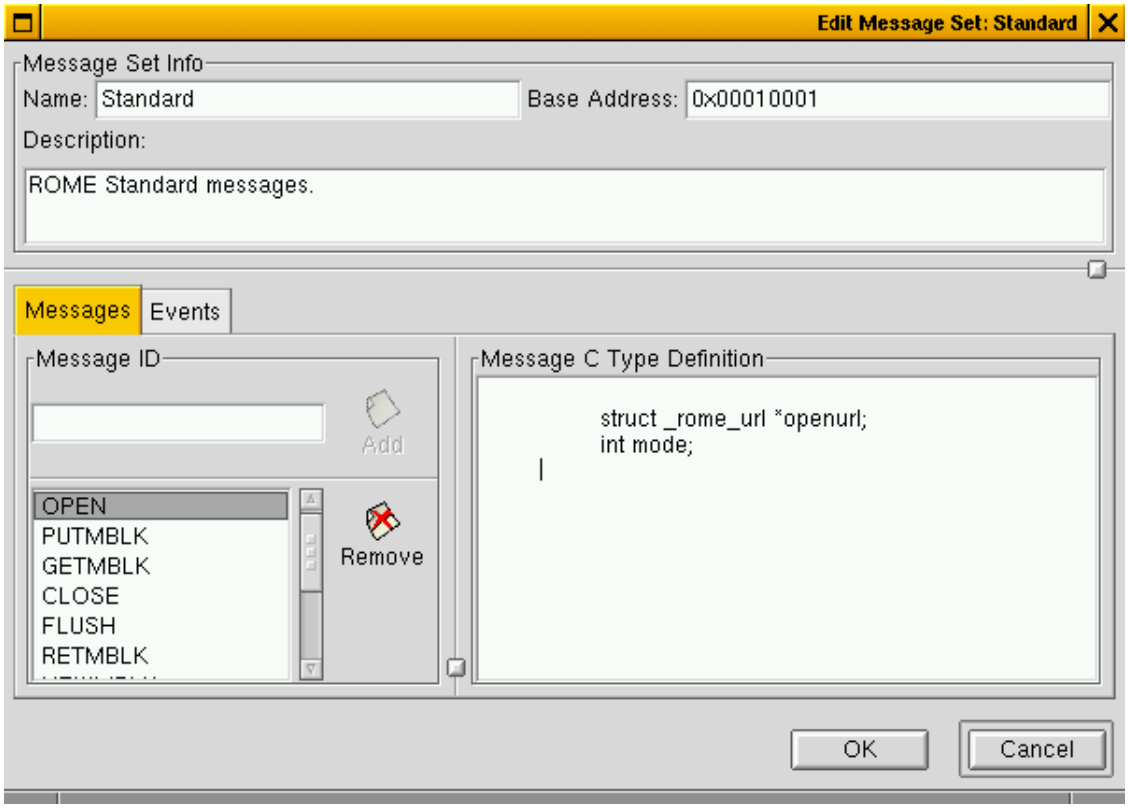

Figure 18: Message Set Dialog

You can define the name of the message set and the base address for message and event numbering. RTB will create `#defines` and `C structs` in the `Messages.h` file and use the information provided here for message and event code numbering. You must make sure that message set base addresses do not overlap to prevent conflicts.

On the lower left side you can select from the list of messages and events for the message set.

7.2.1 Adding a message

To add a message or event simply select the appropriate tab and enter the name and C structure definitions in the textfields. Press the **Add button** to add the message or event to the list.

7.2.2 Edit a message

To change a message or event, select it from the list and simply change its values in the textfield. The changed values will automatically be updated in the message set once you leave the dialog or select another message/event.

7.2.3 Delete a message

To delete a message, select the message from the list and press the **Remove button**.

8 Modules

Modules are the main building blocks of a ROME system. A module contains a module description which contains:

- A list of processes that are running in the module as well as their priority, stack size and key entry points;
- A list of source files that are used for building the module;
- Definitions for creating a library version of the module (not implemented yet);
- A list of options that can be applied to the module.

8.1 Module configuration file

The module configuration file for the module “NetMon” in our example will be stored as

`/home/gnagflow/rtb/Projects/NAT_Box/Modules/NetMon/NetMon.rtb`

Within the `/home/gnagflow/rtb/Projects/NAT_Box/Modules/NetMon/` directory, RTB will also store all source and header files associated with the “NetMon” module.

8.2 Module Dialog

The module dialog (Figure 19) gives you the possibility to edit all important attributes of a module. When the module dialog is activated for the first time, it comes up with the “Processes” tab active. If you select another tab, this setting will be remembered for the next time you open the module dialog, even if you open another module.

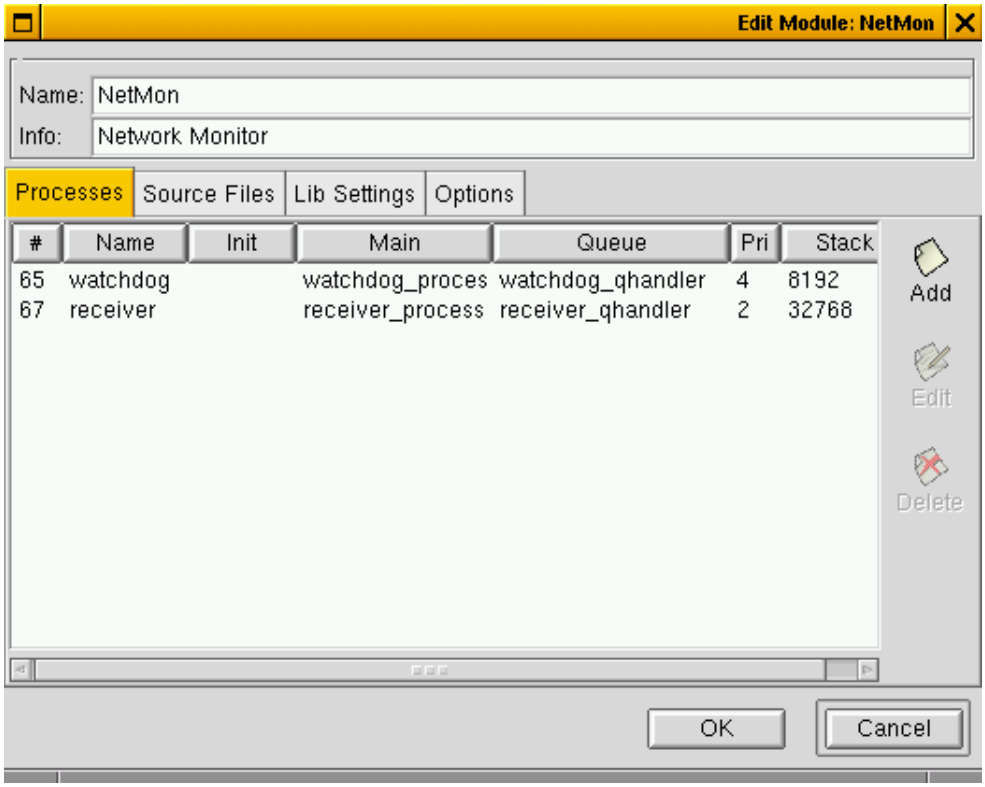

Figure 19: Module Dialog

8.2.1 Processes tab

The "Processes" tab (Figure 20) shows a list of the processes defined in the module. It displays the information about:

- Init order of the process
- Name of the process
- Name of the init function, the main process function and the queue handler function
- Process priority
- Process stack size

These represent the default values that are used when a new module is added to a project. They are defined on a per module basis. However, they can be changed on a per project basis in the project window (see section 5.7, page 19). Changes on project basis *will not* affect the settings of the module configuration.

You can *add*, *edit* and *delete* process entries from the module configuration by selecting a process from the list and clicking the one of the buttons on the right side of the process list.

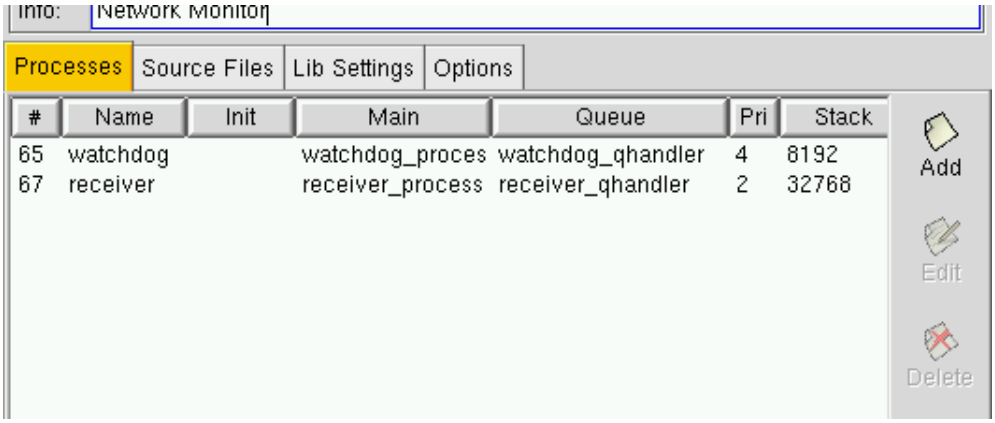

Figure 20: Module dialog “Processes” tab

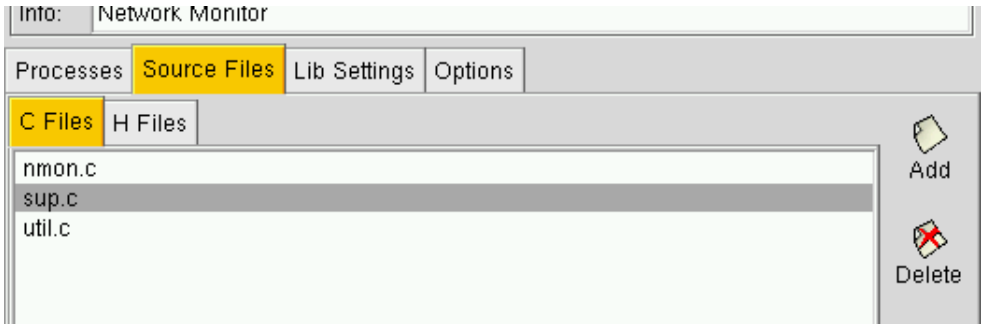

Figure 21: Module Window “Source Files” tab

8.2.2 Source Files tab

The “Source Files” tab (Figure 21) gives you the ability to *add* or *remove* source files from the module. You can select between C (which include Assembler) and H(eader) files and *add* or *remove* files using the buttons on the right side of the dialog. When adding files you will be asked, if you want to copy the files into the module directory, or just add them to the configuration. Usually you want to copy them from another location into the module directory. However, in some cases (e.g. you are adding new files into an already existing module directory) you just want to make the new files known to RTB. In this case you don’t need to copy the files (although it would do no harm).

For Header files you can select which files should be *exported*, e.g. which files should be placed into the main `include` directory when the target system is built. It makes sense to define the interface of a module in a specific header file, which will then be exported, while keeping definitions, that should only be known inside the module, in *private* header files. To export a header file you can check the checkbox at the left side of the file entry. In the example in Figure 22 the file `nmon.h` is exported, while the other two files are private to the module.

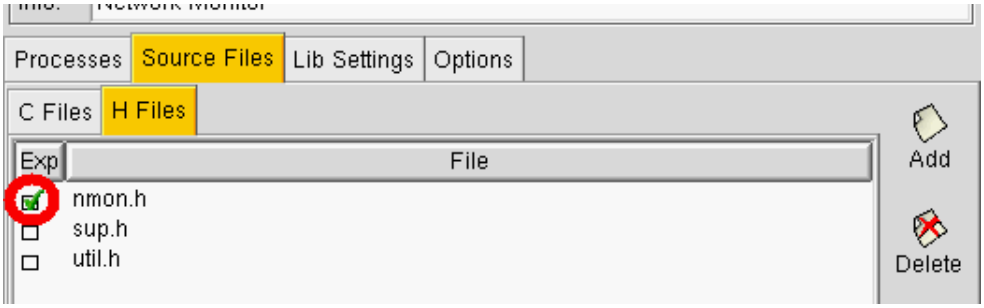

Figure 22: Exporting Header files

8.2.3 Library Settings tab

The “Lib Settings” tab has no function yet, as this functionality is not yet implemented.

8.2.4 Options tab

Every module can contain a set of *options*. A module option will be translated in to a `#define` when the system is built. Options therefore enable the possibility of creating conditional code or of providing certain values to take effect at compile time. The option tab (Figure 23) gives an overview of the options defined in the module. You can select options, *edit* their values, *add* or *delete* them. To edit an option, simply select it from the list and change the option value on the right side of the dialog. To delete an option, select it from the list and press the delete button. To add a new option, enter an option name into the textfield, enter the option value (if appropriate) and press the add button. Option values can be anything from single digit numbers to complex structures or code. Remember that the option value will appear on the right side of the `#define`. If you want to use multiple lines, you will have to add the backslashes (\) at the line endings

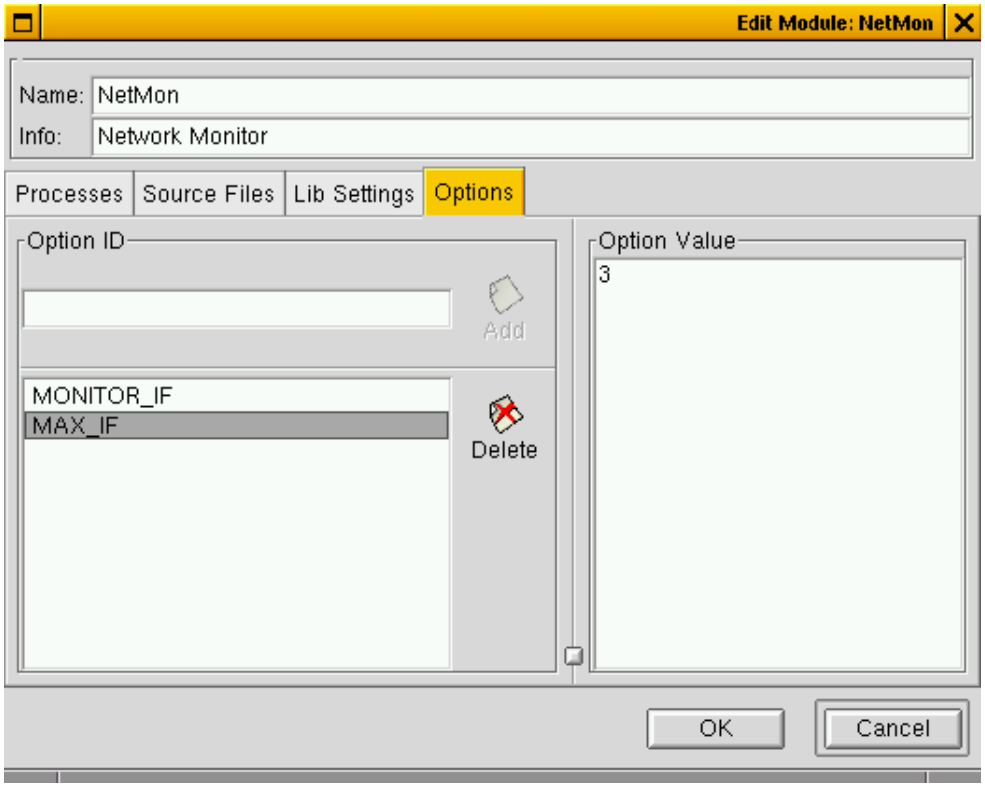

Figure 23: Module dialog “Options” tab

8.2.5 Process view/edit dialog

This dialog (Figure 24) gives you the option of creating or changing a process in a module. Simply enter the values you want into the appropriate fields and press OK.

The “Process uses stdio” checkbox must be enabled if your process uses IO functions like, for example, `printf()`. If this checkbox is enabled, the ROME initialization code will open a connection to the system console for this process. This presumes, of course, that a console process is included in the system.

Note: The names of the init function, the process main function and the queue handler expand as you type in the process name. Usually it’s a good idea to keep these names, as it will produce a consistent naming scheme throughout modules. However, for whatever reason, you can change the names of the functions after you completed typing the process name.

9 CVS Repositories

CVS repositories define CVS serves in the network that contain ROME targets, message sets or modules. You can check out ROME components from a CVS repository and use it in your project. It does not matter where the CVS repository is actually located. As long as you know the server name, the root path for the ROME repository on the server and the username/password combination, you can use the CVS repository.

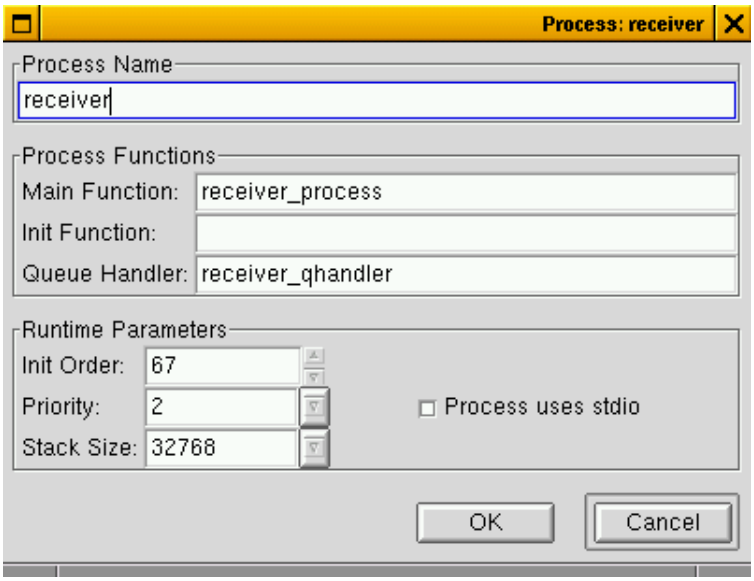

Figure 24: Process edit dialog

9.1 CVS Repository Window

When you first start RTB and open up the CVS Repository Manager window it will be empty. You will have to add CVS repository server entries in order to use them. After you have done that you get a window like shown in Figure 25. This window shows a list of all known repositories as well as the server location.

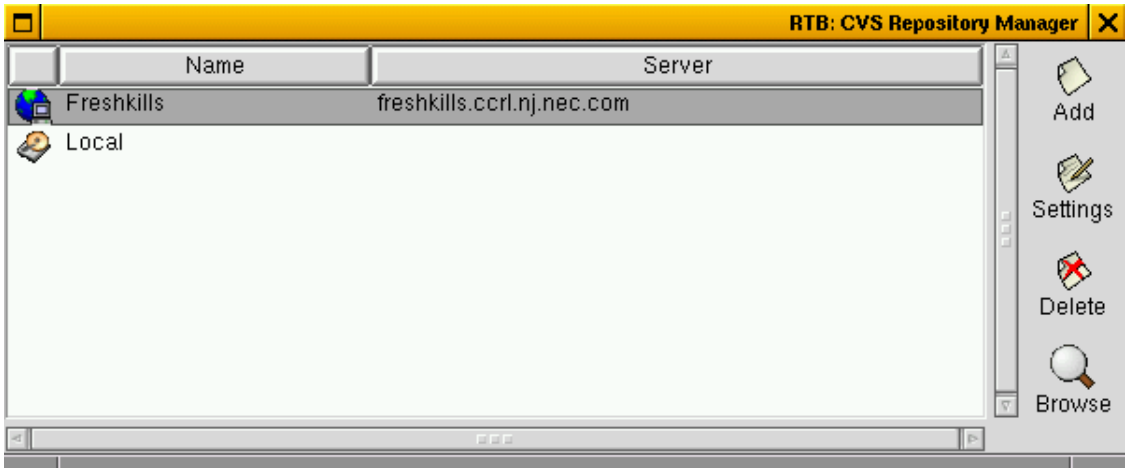

Figure 25: CVS Repository Manager Window

9.1.1 Adding a CVS Repository

To add a new repository, press the **Add button**. You will be presented with a dialog (Figure 26) to enter the information about the repository and the server on which the repository resides. You will have to enter the following information:

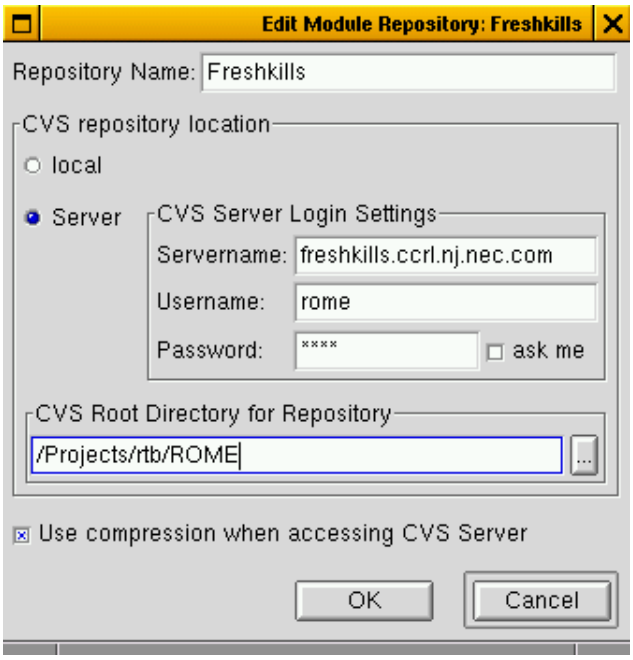

Figure 26: CVS Repository add/edit/view dialog

- Repository Name

  The name of the repository *entry*. This is the name as it will appear in the repository list in the repository manager window. This is *not* the name of the server.

- CVS repository location

  The place where the repository resides. This can be your local machine or a server. If you select server you also have to provide:

  – Servername

    The name of the server machine. This is the real name of the actual CVS server. In our example the name of the *machine* and the *entry* are the same. This is, of course, *not* necessary.

  – Username/Password

    The username password combination for the CVS server. If you set up your own server, you will know it, otherwise ask the administrator of the server. If you do not want to store your password in the RTB configuration file, you can activate the **ask me** checkbox. In this case RTB will ask you for the password if it needs it. It will not be stored your disk.

- CVS Root Directory for Repository

  Here you have to enter the CVS root directory as it is defined for the ROME repository on the server. Again, if you set up the repository yourself, you will know the root directory. If not, you will have to ask your the system administrator.

- Use compression when accessing CVS Server

  CVS can use compression when transferring files. This is very useful, if you use a slow network connection such as a modem or you have a server on the other side of the globe. If in doubt, activate the checkbox. It won't hurt.

Once you have entered all information, press OK to add the entry to the list.

9.1.2 Edit a CVS Repository entry

To edit an entry, select it from the list and press the **Settings button**. The same dialog as described for *adding* entries will be used for changing an entry.

9.1.3 Delete a CVS Repository entry

To delete a repository entry, select it from the list and press the **Delete button**. You will be asked to confirm the deletion and the entry will be removed from the list.

9.2 Browsing a CVS Repository

To browse the contents of a CVS repository, press the **Browse button**. A repository browser window (Figure 27) appears on the screen and shows the content of the repository in form of a entry tree. The

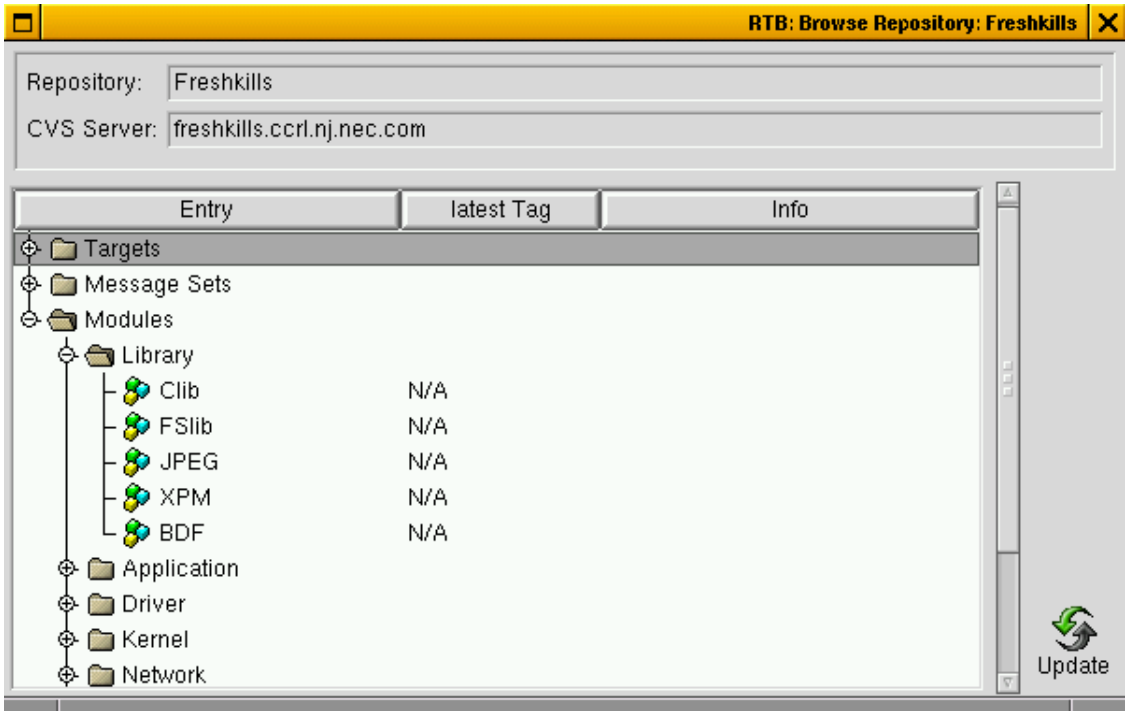

Figure 27: Unassociated CVS Repository Browser Window

tree looks very similar to the tree in the *project overview* tab of the project window. However it shows the entries located on the CVS server.

The repository browser window shown in Figure 27 is *not associated* with any project. You can browse the repository and update the tree, but you can not check out any modules at this point. In order to check out modules, you have to associate the browser to a certain project. See section9.3 on how to do that.

9.3 Checking out an entry

To check out an entry, press the **Repository Browser button** in your project window. This will open a repository browser window (or raise an already existing window of that repository) and associate it with the project. As you can see in Figure 28, it now contains the name of the project and a checkout button. To check out one or more entries, simply select them and press the **Checkout button**.

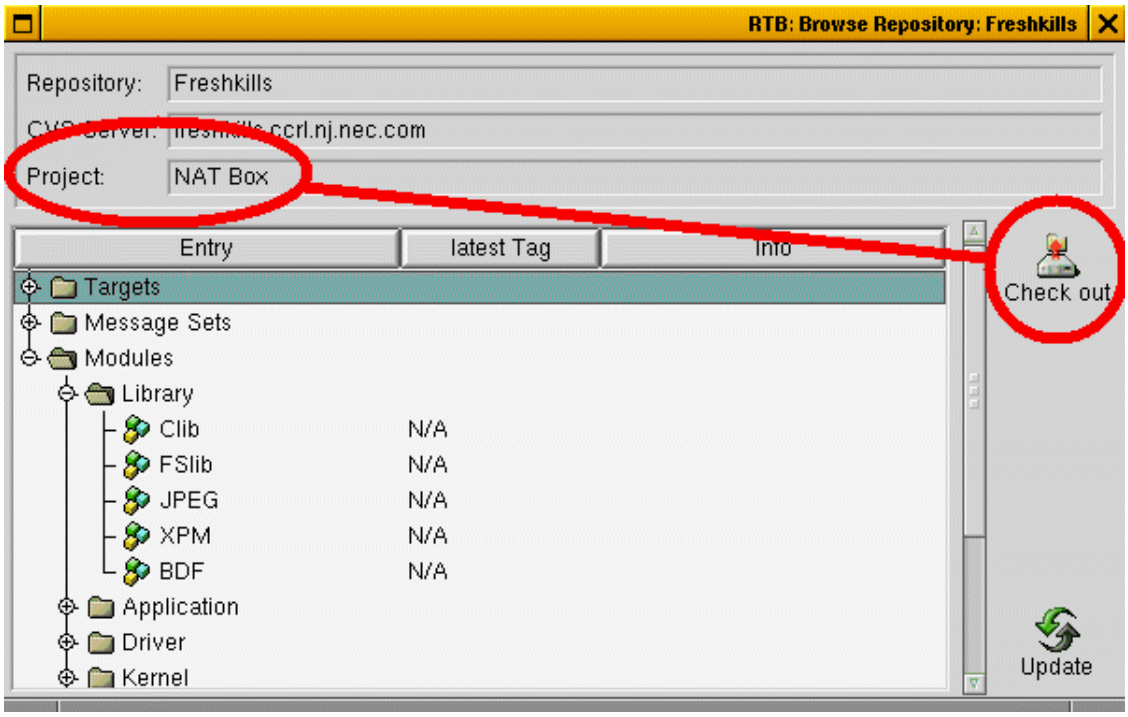

Figure 28: Associated Repository Browser Window

9.3.1 Checking out a single entry

If you want to check out a single entry, you will be presented with a checkout dialog as shown in Figure 29. If you want to check out the current version of the entry, you can simply click OK. However, you have the option to check out a specific version of the entry by selecting the “Choose Tag” radiobutton and selecting a tag from the list. The “Entry Info” textfield will show additional information about the selected tag, as it has been entered when the entry was checked in.

9.3.2 Checking out multiple entries

If you select more than one entry, you will not have the option to choose specific tags for each entry. A reminder dialog (Figure 30) will appear on the screen to confirm that you want to check out the *latest version* of *all* selected entries.

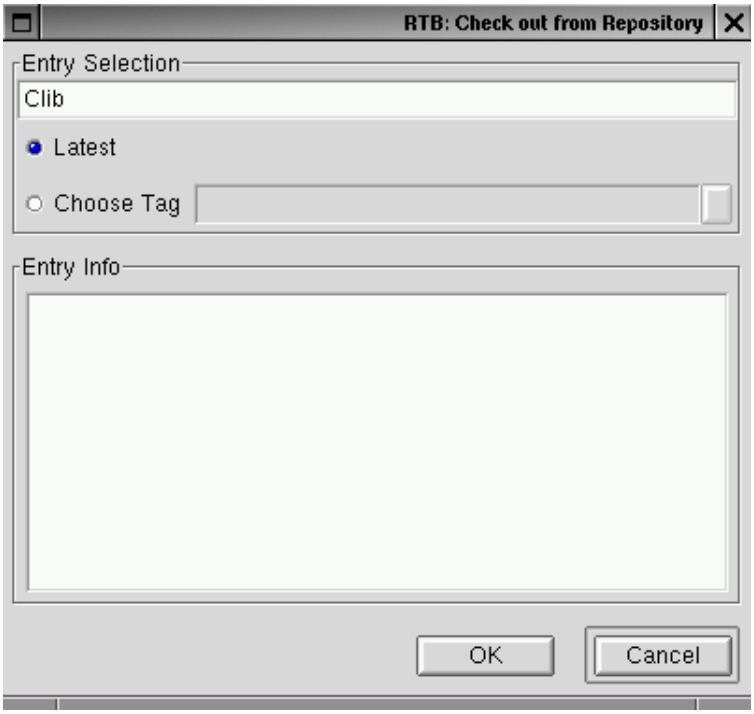

Figure 29: Checkout dialog for a single entry

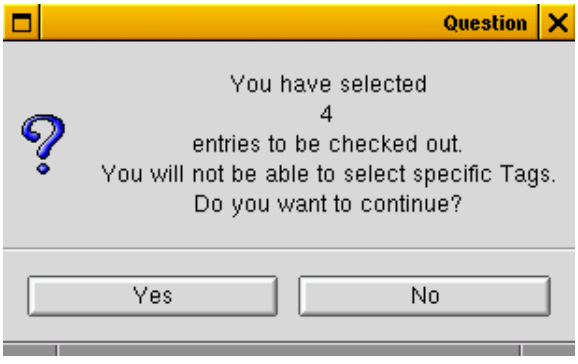

Figure 30: Info dialog when checking out multiple entries