Technologic SYSTEMS

www.embeddedarm.com (480) 837-5200

Getting started with RTAI on TS-7000

For users and developers

Some Notes

Dr. Michael Neuhauser has written most of the code for the Adeos port to the Cirrus Logic EP9301 processor. Adriano Winter Bess and Ronald Gomes have made further modifications to port Adeos to Technologic System's TS7200 and TS7250 boards.
The development is not yet finished and we welcome any contributions or improvements. This guide has been written by Ronald Gomes.

Very quick review of Real-Time, ADEOS and RTAI

Real-time systems are special applications that have to respect specific timing requirements and must be precise in their periodicity (small jitter) and/or be very fast in serving real-time events such as interrupts or IO signals (small latency). Hard real-time applications require latency and jitter accuracy of less than 10us (micro-seconds).

Adaptative Domain Environment for Operation Systems (Adeos) implements a layer between hardware and operation system software that allows multiple operation systems sharing the same hardware while running at the same time. A software pipeline is used to service interrupts and dispatch them to the proper destination; one of its applications is real-time.

Real Time Application Interface (RTAI) is a set of kernel modules that provide real-time services such as inter-process communication, a serial line driver, and an API for real time applications. RTAI services run under the Adeos kernel, so it is necessary to build and run Adeos first. The Adeos kernel is based on the Linux kernel, providing the ability to make it fully preemptable. The RTAI version used in this guide is magma/vulcano.

Useful Reading

http://people.mech.kuleuven.be/~bruyninc/rthowto/
http://www.adeos.org/
http://www.opersys.com/ftp/pub/Adeos/adeos.pdf
http://www.rtai.org/
http://www.rtai.org/modules.php?name=Content&pa=showpage&pid=6
http://www.rtai.org/documentation/magma/html/api/
http://www.rtai.org/
http://www.rtai.org/modules.php?name=Content&pa=showpage&pid=6
http://www.rtai.org/documentation/magma/html/api/

Installing the real-time files

Before loading and running the real-time kernel, you should copy the necessary files to the file system in use on the target. These files include the Linux modules for the new kernel, real-time service modules, test files and documentation.

Technologic Systems does provide tarballs containing all the necessary files. You can download these tarballs from ftp://oz.embeddedarm.com login as anonymous. There are avaiable options for development or user environment installation for both ARM based boards. The first requires about 12MB, while the user files are about 6MB. They are loacated at the /rt/arm-ts8 directory:

ts72xx_rtfiles_deb.tar.gz for TS72XX running debian development filesystem.
ts7200_rtfiles_obf_dev.tar.gz OR ts7200_rtfiles_obf_usr.tar.gz for TS7200 with on board flash fs.
ts7250_rtfiles_obf_dev.tar.gz OR ts7250_rtfiles_obf_usr.tar.gz for TS7250 with on board flash fs.

Extracting the tarball at / directory of your target system will install the 2.4.26-vrs1-cirrus-1-2-1-ts8-rt files in /lib/modules, as well as the RTAI tests and documentation in /usr/realtime.

```
$ cd /
$ tar -xzf ts72xx_rtfiles_deb.tar.gz
```

It would be better if the user were able to compile RTAI regarding his needs or at least select only the necessary files from the shipped tarball. For example, to save space and if you don' tneed real time support in user space, you can install only these modules: rtai_hal, rtai_up, rtai_fifos, rtai_sem. The footprint would be about 1.5 MB.

Running the ADEOS real-time kernel

You can obtain the real-time kernel from ftp://oz.embeddedarm.com login as anonymous and download the files:

/rt/arm-ts8/ts7200_rtzimage OR /rt/arm-ts8/ts7200_rtvmlinux.bin for TS7200 boards.
/rt/arm-ts8/ts7250_rtzimage OR /rt/arm-ts8/ts7250_rtvmlinux.bin for TS7250 boards.

After downloading the real-time kernel, you can load it using redboot's load command then execute the kernel by using redboot's exec command. Below is an example of how to load and execute a kernel via one TFTP server:

```
$ load -r -b 0x00218000 -h 192.168.0.11 /tftpdir/ts7250_rtvmlinux.bin
$ exec -c "console=ttyAM0,115200 ip=dhcp root=/dev/hda1"
```

Running the RTAI test suite

You can verify if the real-time kernel is running by executing " uname -r " this should print the kernel release version. If the release version ends in " -rt " then the real-time kernel is running.

```
$ 2.4.26-vrs1-cirrus-1-2-1-ts8-rt
```

Once you have the necessary files in your file system and the real-time kernel is running you can try the default test suite from RTAI. For example, go to /usr/realtime/testsuite/kern/latency/ and execute ./run.

If your output matches the output listed in the next session (Output of RTAI kernel latency test), then you have succeeded in building your real-time system and should be able to build your own RTAI-based real-time applications! If you want to become a RTAI expert, you should read the related documentation and API. It is outside the scope of this guide to explain how to program using the RTAI API.

Output of RTAI kernel latency test

```
ts7200:/usr/realtime/testsuite/kern/latency# ./run
*
* Type ^C to stop this application.
*
RTAI[hal]: mounted (PIPED).
Adeos: Domain RTAI registered.
RTAI[hal]: 3.2 mounted over Adeos 2.4r17c3/arm-ep9301.
RTAI[hal]: compiled with gcc version 3.3.4.
RTAI[malloc]: kmalloced extent c5580000, size 131072.
RTAI[malloc]: loaded (global heap size=131072 bytes).
RTAI[sched_lxrt]: loaded (PIPED, UP, KERNEL SPACE).
RTAI[sched_lxrt]: timer=periodic (TIMER1),.
RTAI[sched_lxrt]: standard tick=100 hz, CPU freq=983040 hz.
RTAI[sched_lxrt]: timer setup=3052 ns, resched latency=3052 ns.

## RTAI latency calibration tool ##
# period = 1000000 (ns)
# avrgtime = 1 (s)
# check overall worst case
# do not use the FPU
# start the timer
# timer_mode is oneshot

RTAI Testsuite - UP latency (all data in nanoseconds)
RTH|    lat min|    lat avg|    lat max|   overruns|  freq.cntr
RTD|          0|      15347|     113932|          0|       1000
RTD|          0|       1727|     113932|          0|       1000
RTD|          0|       1826|     113932|          0|       1000
RTD|          0|       1820|     113932|          0|       1000

CPU USE SUMMARY
# 0 -> 5097
END OF CPU USE SUMMARY

RTAI[malloc]: kfreed extent c5580000, size 131072.
RTAI[malloc]: unloaded.
RTAI[sched_lxrt]: unloaded (forced hard/soft/hard transitions: traps 0, syscalls 0).
Adeos: Domain RTAI unregistered.
RTAI[hal]: unmounted.
```

WARNING

You may find some error when running the RTAI testsuite if you are using the onboard flash filesystem due missing or old libraries, specially those related to float point support. However, you can still install the RTAI modules in the kernel and develop your real-time application using RTAI API.

Browsing RTAI Magma API

The API documentation is included in the tarball for your convenience. The documentation is written in HTML. If you want to host it using Apache from the target system, check where the www root directory is and create a symbolic link to /usr/realtime/share/doc/rtai-3.2/html/api/ , as the following examples does:

```
$ ln -s /usr/realtime/share/doc/rtai-3.2/html/api/ /var/www/rtai
$ ln -s /usr/realtime/share/doc/rtai-3.2/html/api/ /www/apache/htdocs/rtai
```

You can then use your preferred web browser to view the API documentation at http://192.168.0.50/rtai/ (assuming your IP address is 192.168.0.50). Or you can just locally navigate within the documentation files if you can access them from your host PC.

Building the ADEOS real-time kernel

1. Getting TS-Linux source (same file for TS-72xx boards):

```
ftp server: oz.embeddedarm.com
user: anonymous
file: /linux/linux24-ts8-kernelsource.tar.gz
```

2. Getting ADEOS patch:

```
ftp server: oz.embeddedarm.com
user: anonymous
file: /rt/arm-ts8/ts72xx_ts8_adeos.patch
```

3. Patching the kernel:

Copy the downloaded files to your development directory, extract the kernel source tarball and, inside the kernel directory, apply the patch:

```
$ tar -zxvf linux24-ts8-kernelsource.tar.gz
$ cd linux24
$ patch -p1 < ../ts72xx_ts8_adeos.patch
```

Now you have the kernel sources ready to be compiled. Note the additional adeos subdirectory, which contains some generic ADEOS sources.
4. Installing the cross-compile environment:

Obtain the cross toolchain at Technologic System website at:

http://www.embeddedarm.com/downloads/Linux/ARM/crosstool-linux-0.28rc39.tar.bz2

Install it in your preferred directory (this document assumes / as the installation directory).
You also need to cross-compile and install the modutils for 2.4 kernel to succeed in the installation of the target modules. You can get the latest version of modutils at:

http://www.kernel.org/pub/linux/utils/kernel/modutils/v2.4/

If you get the 2.4.27 modutils version, you can use the following commands to compile and install it (assuming / as the cross-compile and modutils installation root).

```
$ tar xjf modutils-2.4.27.tar.bz2²
$ cd modutils-2.4.27
$ ./configure --build=i386-linux --host=arm-linux --prefix=/usr/local/opt/crosstool/arm-
linux/gcc-3.3.4-glibc-2.3.2
$ make
$ make install
```

5. Configuring and cross-compiling the kernel:

First, check the CROSS_COMPILE and MOD_UTILS variables in your Makefile to confirm that they match the location of your development system. Using this guide definitions, they must be set to:

```
CROSS_COMPILE=/usr/local/opt/crosstool/arm-linux/gcc-3.3.4-glibc-2.3.2/bin/arm-linux-
MOD_UTILS=/usr/local/opt/crosstool/arm-linux/gcc-3.3.4-glibc-2.3.2/sbin/depmod
```

After that, you can configure the source:

```
$ make ts7200_config  OR  $ make ts7250_config
$ make menuconfig
```

Go to General Setup section and set Adeos Support to built-in kernel option. Change anything else regarding yours needs, save configuration and exit menuconfig. For compiling, type:

```
$ make dep
$ make zImage && make modules
```

Now, you are able to install the modules in your preferred directory:

```
$ make modules_install INSTALL_MOD_PATH=/lib/modules
```

Building RTAI

1. Get the latest stable vulcano RTAI version from rtai website (www.rtai.org) or use the link below:

http://www.rtai.org/modules.php?name=Downloads&d_op=getit&lid=32

You can also try the vulcano version from CVS, but regarding it is under development and might have some errors. To install the vulcano source from CVS in your development directory, type:

```
$ cvs -d:pserver:anonymous@cvs.gna.org:/cvs/rtai co vulcano
```

2. Cross-compiling RTAI system
Once you have the cross-compile environment installed, you can type:

```
$ make ARCH=arm CROSS_COMPILE=/usr/local/opt/crosstool/arm-linux/gcc-3.3.4-glibc-
2.3.2/bin/arm-linux-
```

The menuconfig will appear and you can set the Linux Build and Real Time Installation directories. Configure any other options you may need then save and exit. To install run the following command.

```
$ make install
```

WARNING

Read the message with subject rtai-3.2 build error posted by Adriano Winter Bess to RTAI mailing list if you find related problems in the file base/include/asm/adeos_hal.h during the compilation step:

https://mail.rtai.org/pipermail/rtai/2005-June.txt

SUPPORT

Email: support@embeddedarm.com

DOCUMENT HISTORY
==========
07.29.2005 – Created
08.10.2005 – Revision #1
08.18.2005 – Revision #2