Steps to Build and Install RTAI

- Download Linux kernel 3.4.67 and RTAI 4.0 files into /usr/src/ and unpack the files
  - sudo -s <Enter Password>
  - tar xfv linux-3.4.67.tar.bz2
  - tar xfv rtai-4.0.tar.bz2
- Rename the directories to a more descriptive name
  - mv linux-3.4.67 linux-build-3.4.7-rtai-4.0-<Roll-Nos>
  - mv rtai-4.0 rtai-4.0-<Roll-Nos>

Steps to Build and Install RTAI

- Create symbolic links to the two new folders
  - ln -snf linux-build-3.4.67-rtai-4.0-101102 linux-101102
  - ln -snf rtai-4.0-101102 rtai-101102
- Patch the kernel source with RTAI
  - cd /usr/src/linux-101102
  - patch -p1 -b < /usr/src/rtai-101102/base/arch/x86/patches/hal-linux-3.4.67-x86-4.patch

Steps to Build and Install RTAI

- Make the new kernel configuration the most similar to the already installed one.
  - cp /boot/config-`uname -r` ./.config
  - make oldconfig # then press Enter to all the prompts
- Configure the new kernel
  - make menuconfig

Steps to Build and Install RTAI

- Linux Configuration to support RTAI:
  - Enable loadable module support> Module versioning support= no
  - General setup> Local version-append to kernel release= -rtai-4.0-101102
  - Processor type and features> Interrupt pipeline= yes
  - Processor type and features> Processor family= < Pentium Classic >
  - Processor type and features> Multi core scheduling= no/yes
  - Processor type and features> Support sparse irq numbering= no
  - Processor type and features> Enable-fstack-protector buffer overflow detection= no
  - Power management options> CPU Frequency scaling> CPU Frequency scaling = no
  - Power management options> APM BIOS support= no

Steps to Build and Install RTAI

- Make a backup for the new configuration file
  - cp .config /boot/config-3.4.67-rtai-4.0-101102
- Compile the kernel and modules.
  - make -j 2 or (make bzImage; make modules)
- Install the modules, create initrd image. Copy the kernel bzImage to /boot.
  - make modules_install
  - mkinitramfs -o /boot/initrd.img-3.4.67-rtai-4.0-101102 -v 3.4.67-rtai-4.0-101102
  - cp arch/x86/boot/bzImage /boot/vmlinuz-3.4.67-rtai-4.0-101102

Steps to Build and Install RTAI

- Make an entry for the new RTAI kernel in /boot/grub/menu.lst or /boot/grub/grub.cfg or use update-grub command
- Reboot and choose the new RTAI patched kernel in Grub, at boot time.

Steps to Build and Install RTAI

- Create a build tree separated from the source tree of RTAI.
  - cd /usr/src/rtai-101102; mkdir build; cd build
- Configure and compile RTAI
  - make -f ../makefile menuconfig
  - General > Linux source tree = /usr/src/linux-101102
  - General > Installation directory = /usr/realtime-101102
  - Machine (x86) > Number of CPUs (SMP-only) = 2
  - Base system > Scheduling options > One-shot timer mode = yes
  - Base system > Other features > task switches specific signal = yes
  - Add-ons > Real Time COMEDI support in user space = no

Steps to Build and Install RTAI

- Install RTAI
  - make install
- Backup the RTAI device files
  - cp -a /dev/rtai_shm /lib/udev/devices/
  - cp -a /dev/rtf[0-9] /lib/udev/devices/
- Configure RTAI dynamic libraries to be wide available
  - echo /usr/realtime-101102/lib/ >/etc/ld.so.conf.d/rtai.conf
  - ldconfig

Steps to Build and Install RTAI

- Add RTAI 'bin' directory to the $PATH variable
  - export PATH=$PATH:/usr/realtime-101102/bin
- Copy rtai headers
  - cp -r /usr/realtime-101102/include/* /usr/src/linux-101102/include/.
- RTAI Test
  - cd /usr/realtime-101102/testsuite/kern/latency/
  - ./run (Press Ctrl-C to stop)

RTAI Modules

- A module is a kernel component, dynamically loadable, which runs locked in kernel space.
  - It gets the same kernel privileges and access rights.
  - A module, qualified at compilation time, must declare the well known init_module & cleanup_module functions.
  - A module is installed/uninstalled via nsmod/rmmod commands.

CPU Frequency Scaling

- This allows the CPU frequency to be modulated with workload.
- Many CPUs change the TSC counting frequency also, which makes it useless for accurate timing when the CPU clock can change.

Module Versioning Support

- When a module is loaded into the kernel, insmod (or modprobe) can accomplish its task only if the checksum added to each symbol in the kernel matches the one added to the same symbol in the module.
-
- For example, the symbol printk is exported to modules as something like printk_R12345678 when version support is enabled, where 12345678 is the hexadecimal representation of the checksum of the software interface used by the function.

Advanced Power Management

- The APM model assigns power management control to the BIOS.
-
- BIOS code is never written with RT-latency in mind. If configured, APM routines are invoked with SMI(System Management Interrupt) priority, which breaks the rule that adeos-ipipe must be in charge of such things.
-
- Note: SMI interrupts run at higher priority than RTAI core code or even the I-pipe layer.

Support Sparse IRQ Numbering

- The SPARSE-IRQ considerably adds overhead to critical path of IRQ handling.

CONFIG_CC_STACKPROTECTOR

- This option turns on the -fstack-protector GCC feature. This feature puts, at the beginning of critical functions, a canary value on the stack just before the return address, and validates the value just before actually returning.
-
- Stack based buffer overflows (that need to overwrite this return address) now also overwrite the canary, which gets detected and the attack is then neutralized via a kernel panic.