CAPTAIN'S UNIVERSE



Cosmic Ray Muon Detector TeleGarden Pages

Time on Mars

Bryophyllum Plants Ancient Pages

Jupiter Radio Astronomy Salzburg Tourist Guide

Earth Magnetometer

Serving science and more since 1996

HOME

AJAX & MORE LINUX & MORE RTAI XENOMAI ADEOS IPIPE

JAVA & BROWSERS *NIX ELECTRONICS REVIEWS ARTEMIA FAIRY SHRIMP

∷ RTAI

- RTAI MANUAL
- High latencies with SMI not disabled
- RTAI LXRT Parallel Port Interrupt FIFO Stress Test
- RTAI versus Xenomai comparison
- RTAI maximum task frequency test
- RTAI Error opening /dev/rtf
- RTAI 3.3 RTDM Serial Port Example
- RTAI Serial Port Communication Example
- RTAI DCF77 Simulator
- RTAI unresolved symbols
- RTAI APIC Error on kernel 2.6.9
- Extending the RTAI API
- RTAI Serial Port Interrupt -Kernel module
- RTAI Parallel Port Interrupt -LXRT
- RTAI Parallel Port Interrupt -Kernel module
- RTAI Hard real time test
- RTAI Real Time Linux Kernel installation (kernel 2.4.25)

API Inspections

API certified inspections complying with OSHA's PSM regulations. www.api-inspection.com

Artemether Manufacturer

WHO and USFDA, compliant facility 99.5 % purity bulk supply with

www.AanBio.com





Extending the RTAI API is easy. In my case I wanted a function returning the current unixtime in a LXRT program. Instead of adding the function to RTAI directly, I had a look at the "showroom" example "signal" and derived the following files from it. (developed on RTAI3.2-test1 and kernel 2.6.8.1 with gcc 3.3.4 on debian sarge testing)

Testing the API extension is done with the Parallel Port Interrupt LXRT Example for RTAI 3.2 - MAGMA example.

```
# make
[...]
# ./loadmods.sh
# ./parlxrtmagma
GIVE THE NUMBER OF INTERRUPTS YOU WANT TO COUNT: 2
unixtime = 1106392139025058000
RETVAL 0, OVERRUNS 0, INTERRUPT COUNT 1
unixtime = 1106392149040604000
RETVAL 0, OVERRUNS 0, INTERRUPT COUNT 2
TEST ENDS
```

The unixtime is provided in nanoseconds, so 1106392139025058000 ns = 1106392139.025058000seconds.

Using the unixtime-extension is as easy as this:

```
#include "rtai unixtime.h"
```

- RTAI Real Time Linux Kernel installation (kernel 2.6.7)
- RTAI FIFO & shared memory examples

... REALTIME

- REAL TIME LINUX
- :: Stress tests

```
[...]
RTIME unixtime;
unixtime = rt_getunixtime();
printf("unixtime = %lld\n", unixtime);
```

Remember: RTIME is a long long, so we need to use %lld for printf'ing.

The RTAI kernel module "rt_getunixtime.c" with the additional function "static RTIME rt_getunixtime(void)":

```
RTAI kernel module rt getunixtime.c:
       extending the API with rt getunixtime, which will provide
       the current unixtime in the RTIME format
       Derived from the showroom example "signal" for RTAI3.2
       http://www.captain.at/programming/
 * /
#include <linux/kernel.h>
#include <linux/module.h>
#include <rtai schedcore.h>
#include "rtai unixtime.h"
MODULE LICENSE ("GPL");
#define MODULE NAME "RTAI GETUNIXTIME"
static RTIME rt getunixtime (void)
       struct timeval tv;
       do gettimeofday(&tv);
       return (RTIME) ( tv.tv sec * (long long) 1000000000 + tv.tv usec * (long long) 1000 );
static struct rt fun entry rtai getunixtime fun[] = {
       [GETUNIXTIME] = { 1, rt getunixtime }
};
int init module(void)
       if (set rt fun ext index(rtai getunixtime fun, RTAI GETUNIXTIME IDX)) {
               printk("Wrong index module for lxrt: %d.\n", RTAI GETUNIXTIME IDX);
               return -EACCES;
       printk("%s: loaded.\n", MODULE NAME);
       return 0;
void cleanup module(void)
```

```
reset_rt_fun_ext_index(rtai_getunixtime_fun, RTAI_GETUNIXTIME_IDX);
printk("%s: unloaded.\n", MODULE_NAME);
}
```

The include file "rtai_unixtime.h":

```
/*
       RTAI kernel module include file rtai unixtime.h:
       extending the API with rt getunixtime, which will provide
       the current unixtime in the RTIME format
       Derived from the showroom example "signal" for RTAI3.2
       http://www.captain.at/programming/
 * /
#define RTAI GETUNIXTIME IDX 1
#define GETUNIXTIME 0
#ifdef KERNEL
#else /* ! KERNEL */
#include <rtai lxrt.h>
static inline RTIME rt getunixtime()
       struct { unsigned int dummy; } arg;
       return rtai lxrt(RTAI GETUNIXTIME IDX, SIZARG, GETUNIXTIME, &arg).rt;
#endif /* KERNEL */
```

The "Makefile": (tested only with kernel 2.6!)

```
prefix := $(shell rtai-config --prefix)
ifeq ($(prefix),)
$(error Please add <rtai-install>/bin to your PATH variable)
endif

CC = $(shell rtai-config --cc)

LXRT_CFLAGS = $(shell rtai-config --lxrt-cflags)

LXRT_LDFLAGS = $(shell rtai-config --lxrt-ldflags)
all:: parlxrtmagma
ifneq ($(findstring 2.6.,$(shell rtai-config --linux-version 2>/dev/null)),)
obj-m := rt_getunixtime.o

KDIR := /lib/modules/$(shell uname -r)/build
```

```
PWD
    := $(shell pwd)
EXTRA CFLAGS := -I/usr/realtime/include -I/usr/include/
all::
       $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
else
TARGET := rt getunixtime
INCLUDE:= -I/lib/modules/`uname -r`/build/include -I/usr/realtime/include
CFLAGS := -02 -Wall -DMODULE -DUSEFIFO -D KERNEL -DLINUX
     := acc
all:: rt getunixtime.o
${TARGET}.o: ${TARGET}.c
      $(CC) $(CFLAGS) ${INCLUDE} -c ${TARGET}.c
endif
parlxrtmagma: parlxrtmagma.c
       $(CC) $(LXRT CFLAGS) -o $@ $< $(LXRT LDFLAGS)
clean::
       $(RM) parlxrtmagma
.PHONY: clean
```

Parallel to USB Adapter

Connect USB Printer to PC Parallel Professional Gate, Ball, Globe, Port with LPT2USB Adapter www.lpt2usb.com

api ball valve

Check Valves, High Quality & Low price www.TF-valve.com

Api Globe Valves CN

Manufacturer of Butterfly Valves, Fluorinated API Intermediates, US Excellent Service Competitive Price manuf. TFA, TFE, ETFA... www.ChinaBioKo.com

Fluorinated Chemicals

www.halocarbon.com



Ads by Google

The modified example "parlxrtmagma.c" (RTAI LXRT hard real time): see Parallel Port Interrupt LXRT Example for RTAI 3.2 - MAGMA

```
/* RTAI LXRT Parallel Port Interrupt - www.captain.at
       This examples is for RTAI3.2 (MAGMA)
       based on usi process.c from showroom
       Needs a "null-modem" for the parallel port:
               connect any output pin (pin 2-9) with the IRQ pin (pin 10 = ACK).
* /
#include <stdio.h>
#include <errno.h>
#include <sys/mman.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <rtai lxrt.h>
#include <rtai sem.h>
#include <rtai usi.h>
#include \langle sys/io.h \rangle
#include "rtai unixtime.h"
#define PARPORT IRQ 7
#define BASEPORT 0x378
static SEM *dspsem;
static volatile int end = 1;
static volatile int ovr, intcnt, retval, maxcnt;
static RTIME unixtime;
static void *timer handler(void *args)
       RT TASK *handler;
       if (!(handler = rt task init schmod(nam2num("HANDLR"), 0, 0, 0, SCHED FIFO, 0xF))) {
               printf("CANNOT INIT HANDLER TASK > HANDLR <\n");</pre>
               exit(1);
       rt allow nonroot hrt();
       mlockall(MCL CURRENT | MCL FUTURE);
       rt make hard real time();
       end = 0;
       rt request irq task(PARPORT IRQ, handler, RT IRQ TASK, 1);
       rt startup irq(PARPORT IRQ);
       rt enable irg(PARPORT IRQ);
       while ( !end && (ovr != RT IRQ TASK ERR) ) {
               do {
                       ovr = rt irq wait(PARPORT IRQ);
                       unixtime = rt getunixtime(); // our new function
                       if (ovr == RT IRQ TASK ERR) break;
                       if (end) brea\overline{k};
                       if (ovr > 0) {
                              // overrun processing, if any, goes here
                               rt sem signal(dspsem);
                       /* normal processing goes here */
                       intcnt++;
                       rt ack irg(PARPORT IRQ);
               } while (ovr > 0);
               rt pend linux irq(PARPORT IRQ);
       rt release irq task(PARPORT IRQ);
```

```
rt make soft real time();
       rt task delete (handler);
       intcnt = maxcnt;
       return 0;
int main(void)
       RT TASK *maint;
       int thread;
       printf("GIVE THE NUMBER OF INTERRUPTS YOU WANT TO COUNT: ");
       scanf("%d", &maxcnt);
       if (!(maint = rt task init(nam2num("MAIN"), 1, 0, 0))) {
               printf("CANNOT INIT MAIN TASK > MAIN <\n");</pre>
               exit(1);
       // create semaphore to notify main() when interrupt occurs
       if (!(dspsem = rt sem init(nam2num("DSPSEM"), 0))) {
               printf("CANNOT INIT SEMAPHORE > DSPSEM <\n");</pre>
               exit(1);
       // ask for permission to access the parallel port from user-space
       if (iopl(3)) {
               printf("iopl err\n");
               rt task delete (maint);
               rt sem delete(dspsem);
               exit(1);
       outb p(0x10, BASEPORT + 2); //set port to interrupt mode; pins are output
       thread = rt thread create(timer handler, NULL, 10000); // create thread
       while (end) { // wait until thread went to hard real time
               usleep(100000);
       while (intcnt < maxcnt) {</pre>
               rt sem wait (dspsem);
               printf("unixtime = %lld\n", unixtime);
               printf("RETVAL %d, OVERRUNS %d, INTERRUPT COUNT %d\n", retval, ovr, intcnt);
       end = 1;
       printf("TEST ENDS\n");
       outb p(0, BASEPORT);
       outb p(255, BASEPORT); // generate final interrupt to unblock rt irq wait
       outb p(0, BASEPORT);
       rt release irq task(PARPORT IRQ);
       rt thread join(thread);
       rt task delete(maint);
```

```
rt_sem_delete(dspsem);
return 0;
}
```

Load the RTAI modules, aswell as our extension-module with "loadmods.sh":

```
#!/bin/sh
TMP=$PWD
cd /usr/realtime/modules/
insmod ./rtai_hal.ko
insmod ./rtai_lxrt.ko
insmod ./rtai_sem.ko
insmod ./rtai_usi.ko

cd $TMP
insmod ./rt_getunixtime.ko
```

Remove the modules with "rmmods.sh":

```
#!/bin/sh
rmmod rt_getunixtime
rmmod rtai_usi
rmmod rtai_sem
rmmod rtai_lxrt
rmmod rtai_lxrt
rmmod rtai_hal
```

Last-Modified: Sat, 04 Feb 2006 16:13:51 GMT



© 1996-2008 captain@captain.at . All rights reserved.

go to top