

RTOS Programming



RTOS Programming

Key concepts of RTOS

Agenda

- ❖ RTOS concept
- ❖ Soft Real Time and Hard Real Time OS
- ❖ RTOS market
- ❖ Example of Real Time System Programming
- ❖ Live Demo on Embedded Linux



2

RTOS Programming

What is RTOS

- ❖ Real Time operating System
 - These operating system are used where task has to be completed in defined time constraints
- ❖ RTOS are used in Real Time Embedded System (RTES)
- ❖ In RTES if task is not completed in defined time then there is no use of it

EmbeddedCraft

IMBUENT

3

Consider an example surface to air missile (SAM)

- ❖ Purpose was To destroy aircraft or missile
- ❖ SAM is equipped with target via missile (TVM) guidance system
- ❖ TVM continuously monitor target
- ❖ TVM send signal to control unit and control unit do the exact (remember exact) timing calculations
- ❖ and fire antimissile to target, calculation is highly accurate and antimissile system can hit target even at 70km far away



EmbeddedCraft

4

RTOS Programming

Patriot Missile Failure

- ❖ It happen on February 25, 1991, during the Gulf War
- ❖ American Patriot missile fail to intercept Iraq's Scud missile, and
- ❖ Scud struck at American army barracks and kill 28 soldiers
- ❖ Actually it was due to software bug



EmbeddedCraft

5

What was software bug 1/2

- ❖ Inaccurate calculation of time since boot
- ❖ System was measuring time in 100 ms, by internal clock
- ❖ Value of time was stored in 24 bit fixed point register
- ❖ Obtained value was multiplied by 10 to get 1 sec
- ❖ Binary representation of $1/10 = 0.0001,1001,1001,1001,1001,1001,1001,100...$
- ❖ Storing above value in 24 bits will result in $0.00000000000000000000000011001100...$ Error = 0.000000095 in decimal



6

RTOS Programming

What was software bug 2/2

- ❖ This number was multiplied by 100 hours
 $0.000000095 \times 100 \times 60 \times 60 \times 10 = 0.34$
- ❖ So mistake was 0.34 second in 100 hour
- ❖ Scud missile speed was 1,676 meters per second
- ❖ In 0.34 second it traveled 500 meter, which was sufficient time for scud to struck on American barracks

Reference

<http://www.howstuffworks.com/patriot-missile.htm>

<http://www.ima.umn.edu/~arnold/disasters/patriot.html>

EmbeddedCraft

IMBUENT

7

Conclusion 1/2

- ❖ Surface to air missile system is the example of **Safety Critical Real Time System**
- ❖ System is useless if
 - Give incorrect result
 - Misses time deadline
- ❖ **Real Time System** is meant for
 - Giving correct result
 - Never ever miss time deadline

EmbeddedCraft

IMBUENT

8

RTOS Programming

Conclusion 2/2

- ❖ *You are free to miss deadline but it could be a threat for life*
- ❖ **Real Time Embedded System** are always important part of Safety critical system

EmbeddedCraft

IMBUENT

9

Soft Real Time and Hard Real Time System

- ❖ In **hard real time system**, deadlines are never ever be crossed,
 - All critical tasks will be completed in defined time limits
- ❖ **Example:**
 - Missiles, Aircraft Flight Control, Special Robots, Medical devices like ventilator etc



EmbeddedCraft

IMBUENT

10

RTOS Programming

Soft Real Time and Hard Real Time System

- ❖ But in **soft real time system**, some critical tasks may be delayed, but due to this there is no any threat to life
- ❖ **Example:**
 - Video Players (24 frames/sec), Internet Telephony etc



EmbeddedCraft

IMBUENT

11

Hard Real Time System ! *Difficult story*

- ❖ Should we use Hard Real Time or Soft Real Time System ?
- ❖ This all depends on requirement
- ❖ Hard real time system are very difficult to handle
- ❖ So, if there is not such need, then **avoid**

EmbeddedCraft

IMBUENT

12

RTOS Programming

RTOS Available in Market

- ❖ Different RTOS are available in market
 - **Proprietary**
 - Property of a organization
 - You have to pay for it
 - Mostly closed source
 - But for sometime Source code can also be available for extra payment
 - **Freeware**
 - Developed by either community or by organization
 - Most of the time open source
 - Most of the open source code software are free

EmbeddedCraft

IMBUENT

13

Proprietary RTOS

- ❖ **Montavista Linux**
<http://www.mvista.com/>
- ❖ **VxWorks**
<http://www.windriver.com/>
- ❖ **Threadx**
<http://www.rtos.com/>
- ❖ **QNX**
<http://www.qnx.com/>
- ❖ **uC/OS – II**
<http://www.micrium.com>

montavista™

VxWorks

THREADX™

QNX

µC/OS-II™
The Real-Time Kernel

EmbeddedCraft

IMBUENT

14

RTOS Programming

The screenshot shows the MontaVista website with a navigation bar including COMPANY, SOLUTIONS, PRODUCTS & SERVICES, SUPPORT, PARTNERS, and EDUCATION. The main content area features a 'Products & Services' section with the headline 'We help you get the most out of open source™'. Below this, there are five product categories: Professional Edition, Carrier Grade Edition, Mobile Linux, Professional Services, and Developer Tools. A sidebar on the left contains links to resources and a quote. A featured webinar by Corey Minyard is also highlighted.

Products & Services
We help you get the most out of open source™

MontaVista Software provides a family of commercial-grade open source Linux software development platforms. As integrated, tested environments, these platforms provide faster time-to-market than in-house roll-your-own Linux deployments. And being based on Linux, they provide developers with greater control than proprietary solutions.

MontaVista Linux is now in its 6th generation, and provides a new approach to embedded Linux development containing some of the latest tools and technologies available in the Linux community.

6 MontaVista Linux

Professional Edition
For general embedded projects, Professional Edition is a comprehensive commercial-grade Linux development platform providing a complete run-time operating system.

Carrier Grade Edition
Designed for telecommunications & data communications, CCE is the industry-standard foundation of a COTS Carrier Grade Linux platform, certified to meet this market's needs.

Mobile Linux
Designed for the needs of wireless handsets and mobile devices, Moblinux is an optimized Linux operating system and development platform with features addressing power management, boot...

Professional Services
With unmatched experience in engineering, QA, and project management, MontaVista's Professional Services team offers both on-site and off-site support.

Developer Tools
With a comprehensive suite of Linux-specific developer tools, MontaVista provides you with direct control over the platform and all the tools you need to get to market quickly and...

“By selecting MontaVista Linux we are able to achieve all these objectives, an open source operating system of the highest quality, rapid...”

Adapting to the Changing Communications Infrastructure with Linux and Open Source

Corey Minyard
Carrier Grade Edition Architect
Recorded Webinar
View »

15

The screenshot shows the Wind River website with a navigation bar including Solutions, Products, Services, Partners, Customers, Education, Support, and Company. A search bar is located at the top right. The main content area features a 'Wind River VxWorks' section with a featured article titled 'Introducing New Solutions for Critical Business Needs from the World's Most Widely Adopted RTOS'. The article includes a 'SHARE' button and a list of key features: Proven, Optimized, and Innovative. A 'Featured' sidebar on the right lists 'Wind River VxWorks Overview', 'Wind River VxWorks Product Note', and 'VxWorks Customer Success Snapshots'.

WIND RIVER

Solutions | Products | Services | Partners | Customers | Education | Support | Company

United States [Search]

Wind River VxWorks

VxWorks | Benefits | Solutions | Technology | Contact Us

Introducing New Solutions for Critical Business Needs from the World's Most Widely Adopted RTOS [SHARE]

Wind River's VxWorks is the world's leading commercial real-time operating system (RTOS) and has been serving the needs of embedded systems of all shapes and sizes for more than 25 years. VxWorks maintains this leadership because it is:

- Proven:** VxWorks has been used in more than 500 million deployed devices, from small consumer products to commercial airliners. The systems that VxWorks gets built into require complete reliability, and the consequences for failure are expensive or, worse, life threatening. Why take chances with such a critical piece of your system?
- Optimized:** VxWorks has been optimized for performance, determinism, and code footprint on each processor platform it runs on. VxWorks is also optimized for specialized hardware support for such features as network acceleration and graphics. Why waste processing power on nonoptimized solutions?
- Innovative:** VxWorks is leading the market in RTOS innovation. These innovations, such as multicore and multi-OS support, provide our customers with the leading-edge solutions they require to stay competitive. Why go with an RTOS that doesn't provide the solutions you need to take advantage of the latest technology?

Core Areas of Innovation

Featured

- Wind River VxWorks Overview
- Wind River VxWorks Product Note
- VxWorks Customer Success Snapshots

16

RTOS Programming

Open Source and Freeware RTOS

❖ Embedded Linux

- **Linux with RTAI Library**
<https://www.rtai.org/>
- **OSADL Realtime Linux**
<http://www.osadl.org>
- **uClinux**
<http://www.uclinux.org/>



❖ FreeRTOS

<http://www.freertos.org>



❖ ECOS

<http://ecos.sourceware.org/>




EmbeddedCraft

IMBUENT

17

New FreeRTOS eBook! "Using the FreeRTOS Real Time Kernel - a Practical Guide"
Includes source code for all examples



Front Page
Download
[OpenRTOS and SafeRTOS](#)
[About FreeRTOS](#)
[Getting Started...](#)
[More Advanced...](#)
[Supported Devices](#)
[API Reference](#)
[Contact and Support](#)
[FreeRTOS Interactive](#)

FreeRTOS™ is a portable, open source, royalty free, *mini* Real Time Kernel - a free to download and free to deploy RTOS that can be used in commercial applications without any requirement to expose your proprietary source code. Downloaded more than 77,500 times during 2008, *FreeRTOS is the cross platform de facto standard for embedded microcontrollers.*

Each official port includes a pre-configured example application demonstrating the kernel features, expediting learning, and permitting 'out of the box' development. Support is provided by an active user community.

OpenRTOS™ is a commercially licensed and supported version of FreeRTOS that includes fully featured professional grade USB, file system and TCP/IP components.

SafeRTOS™ is a SIL3 RTOS version that has been certified for use in safety critical applications. It is a functionally similar product for which complete IEC 61508 compliant development/safety lifecycle documentation is available (conformance certified by TÜV SÜD, including compiler verification evidence).

"The new book is great, it provides very helpful easy to understand examples of FreeRTOS concepts. The book is also an easy read free of unexplained arcane language and concepts... Excellent!!" - Darrell F.

"It's probably safe to say at this point that FreeRTOS goes through more 'peer-review' than any other RTOS available on the planet. I have used it in several projects - one of which was a multiprocessor environment that used more than 64 processors and needed to run for months reliably. The FreeRTOS core performed well. Take FreeRTOS for a spin!" - John Westmoreland

Using the FreeRTOS Real Time Kernel - a Practical Guide" eBook
Including full source code!

NEW! Cortex M3 / LPC17xx edition now available

Latest news snippets:

- o FreeRTOS Interactive has been launched. The first part of its roll out allows users to both upload and download individual user contributed projects. User contributed projects have therefore been removed from the main FreeRTOS download, which is now at version 6.0.4. This permits the official (and supported) FreeRTOS download to significantly decrease in size. V6.0.4 also contains the first Energy Micro EFM32 demo application.
- o FreeRTOS now supports the SuperH from Renesas - complete with embedded web server demo application for the SH7216.
- o A new Cortex M3 / LPC17xx edition of the FreeRTOS eBook "Using The FreeRTOS Real Time Kernel - A Practical Guide" has been released.
- o FreeRTOS V6.0.0, the first version to integrate memory protection support, is now available for free download. See the press release for an overview.
- o Microchip publish an application note showing how to integrate their libraries and stacks with FreeRTOS.
- o Fujitsu launch new FreeRTOS development kits.
- o WITTENSTEIN publish free kernel aware 'State Viewer' plug-in for IAR Embedded Workbench.

site update: March 15 2010

18

RTOS Programming

OSADL Open Source Automation Development Lab

Home | **Projects** | Downloads | Search

You are here: Home / Projects /

2010-03-18 - 11:43

OSADL Project: Realtime Linux

Is realtime possible with Linux? Yes, It is. But which Linux derivate should I use? This is the key question to be answered when Linux is going to be used in the automation Industry.

The OSADL criteria for realtime Linux are:

- Strong and agile community
- Long term availability
- Ease of use (user space realtime)
- POSIX API for real time (standard API)
- Wide architecture support
- Mainstream support by the Vanilla Kernel

This project supports the evolution of the realtime preempt patch maintained by Ingo Molnar, and Thomas Gleixner. We believe that long term stability can only be achieved when the realtime aspects are as close to the Vanilla Kernel as possible.

Realtime Linux Road Map

This is the current status of Realtime Linux using the Realtime-Preempt patches:

Architecture	x86	x86/64	powerpc	arm	mips	68knommu
Feature						

Next OSADL Events:
Hannover Fair 2010
19.04 - 23.04

Breaking News:
2010-03-10 12:00
Parallel real-time on multi-core systems with mainline Linux
Several tasks simultaneously running at real-time priority no longer interfere to each other! [more]

2010-02-22 12:00
"Latest Stable" Linux mainline real-time 2.6.31 is out!
Kernel 2.6.31.12-r21 is our latest and greatest! [more]

19

RTOS Market Survey

Type of Operating System Used for the Current 32/64 Bit Project

Operating System	Percentage
Linux	25%
VxWorks	12%
Windows XPE	8%
Windows CE	5%
Windows CE .NET	4%
Windows NTE	4%
uC/OS	3%
Neutrino	3%
Nucleus	3%
Other	17%
Proprietary OS	8%
No formal OS	8%

Note: Percents sum to over 100% due to multiple responses.

- Developers surveyed by VDC indicate that Linux is the top OS for their projects
- Consistent with other surveys of embedded developers
- 29% of developers survey that Linux will be the embedded OS in their "next" project - growth continues

EmbeddedCraft

IMBUEMENT

20

RTOS Programming

When to use RTOS

- ❖ **Depends on application requirement**
 - Need of multitasking
 - Need to decide priority of task

EmbeddedCraft

IMBUENT

21

Example of a Real Time System

- ❖ Consider a real-time system comprised of three motors and three switches.
- ❖ The switches have two positions, ON and OFF.
- ❖ The switches must be scanned at about 10 times per second,
- ❖ and the motors turned on or off as appropriate.

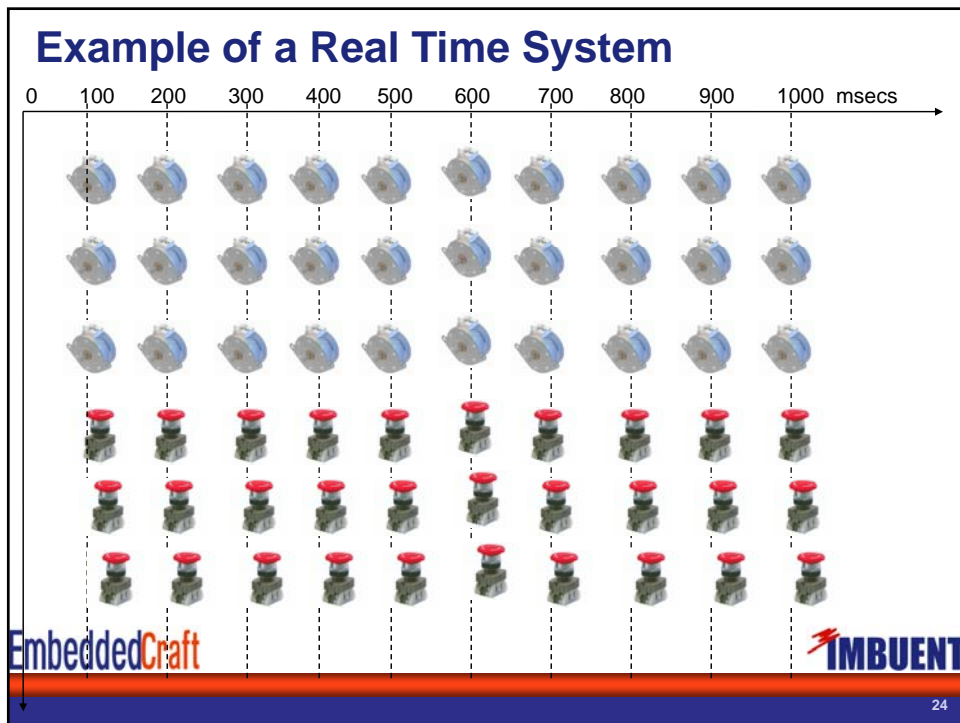
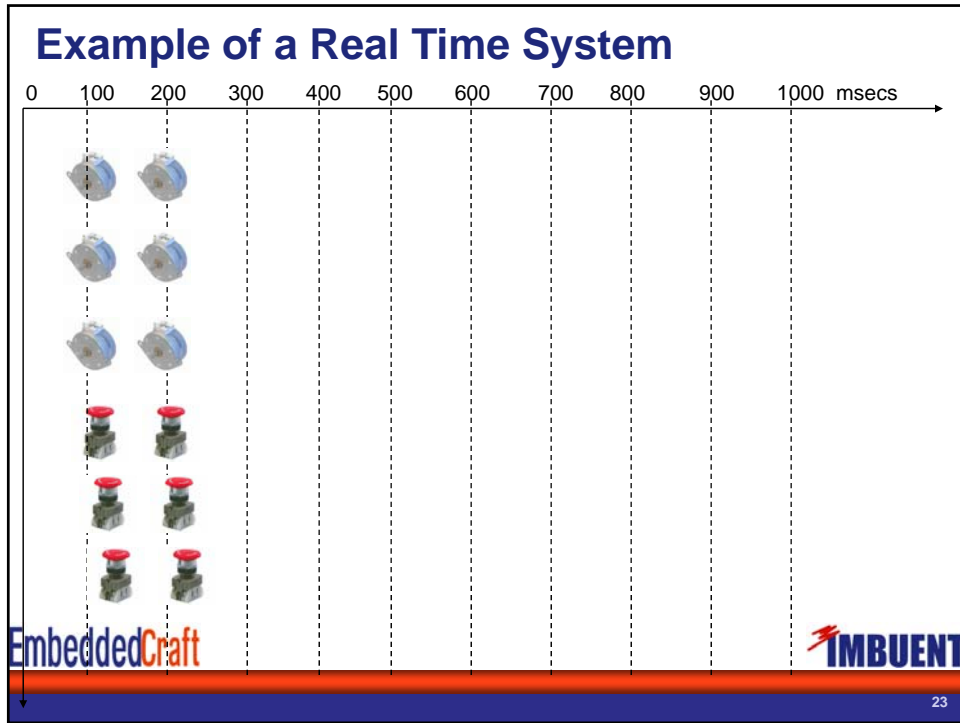


EmbeddedCraft

IMBUENT

22

RTOS Programming



RTOS Programming

Example of a Real Time System

```
void main(void)
{
  int i;
  while(1)
  {
    for (i= 0; i < 3; i++)
    {
      if (switchChanged(i))
        changeMotor(i);
    }
  }
}
```



EmbeddedCraft

..IMBUENT

25

Example of a Real Time System

```
void main(void)
{
  while(1)
  {
    if (OneTenthSecondIsUp)
    {
      for (i= 0; i < 3; i++)
      {
        if (switchChanged(i))
          changeMotor(i);
      }
      OneTenthSecondIsUp = 0;
    }
  }
}
```



EmbeddedCraft

..IMBUENT

26

RTOS Programming

Adding one sensor

- ❖ Let a pressure gage must be checked every 50 milliseconds
- ❖ A valve opened if the pressure is greater than 100 psi.
- ❖ Once opened, the valve must be closed after the pressure drops below 90 psi.

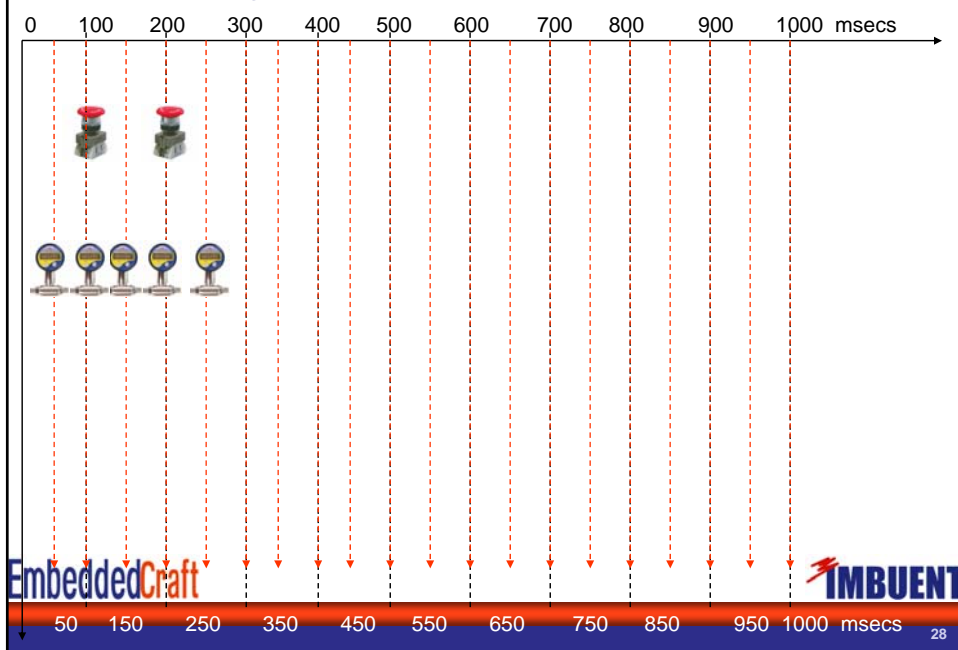


EmbeddedCraft

IMBUENT

27

Example System

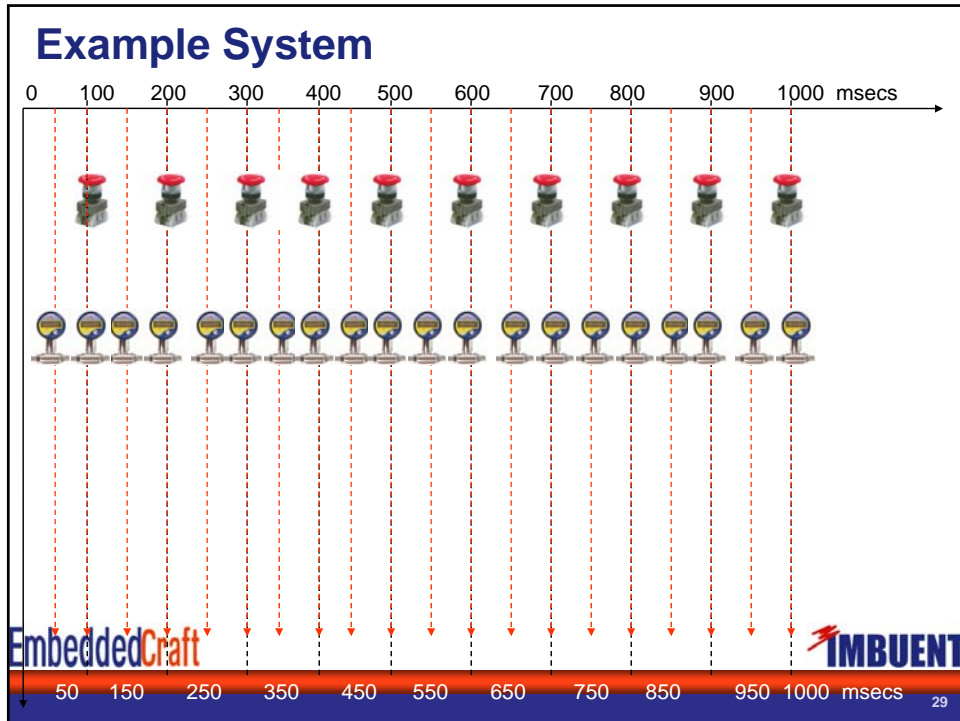


EmbeddedCraft

IMBUENT

28


RTOS Programming




Example

```


if (FiftyMslsUp)
{
    switch (valveState)
    {
        case CLOSED:
            if (pressure() > 100)
            {
                openValve();
                valveState = OPEN;
            }
            break;
        case OPEN:
            if (pressure() < 90)
            {
                closeValve();
                valveState = CLOSED;
            }
    }
    FiftyMslsUp = 0;
}
    
```




S1




S2




S3




M1



M2



M3



NT

RTOS Programming

Let us add datagrams (data packets)

- ❖ Assume that the system is connected to a network
- ❖ and that incoming datagrams must be processed.
- ❖ This could be handled by adding yet another function call to the loop.
- ❖ `checkDatagrams();`



S1



S2



S3



M1



M2



M3

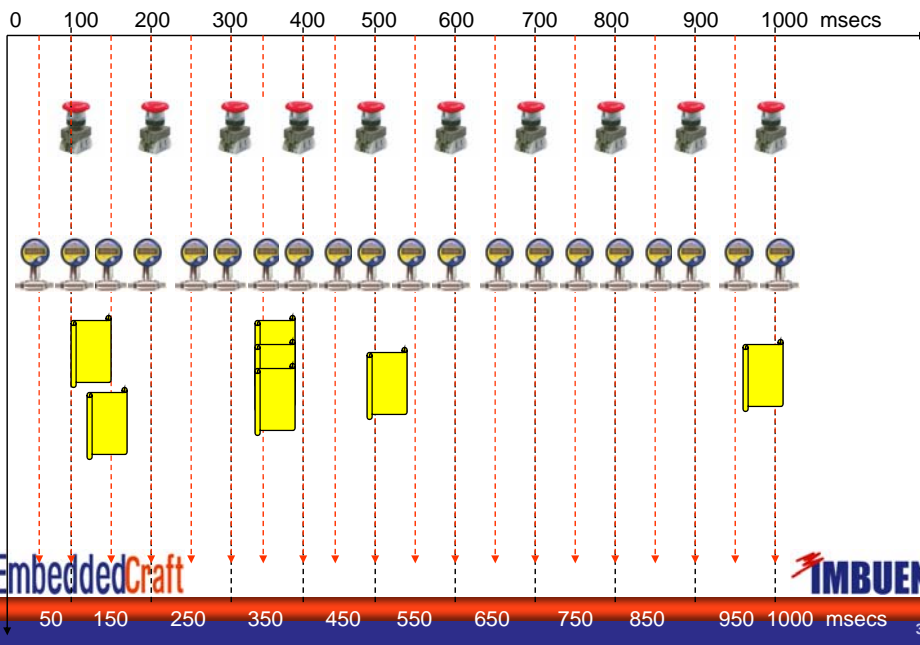


EmbeddedCraft

..IMBUENT

31

Example System



EmbeddedCraft

..IMBUENT

32

RTOS Programming

Let us handle datagrams (data packets)

if the function *checkDataGrams* is not called at a sufficient rate,
datagrams can be lost.
In order to avoid this,
a queue must be created so that when the interrupt service routine for
incoming datagrams is entered
, the datagram is placed into a queue for processing by the function
checkDatagrams. .



EmbeddedCraft

IMBUENT

33

Final code for our system...

```
Void main()  
{  
While(1)  
{  
checkMotorSwitches();  
checkPressure();  
checkDatagrams();  
}  
}
```



EmbeddedCraft

IMBUENT

34

RTOS Programming

Final code for our system: Drawback...

- ❖ There are no priorities
- ❖ Busy waiting should not there
- ❖ And more responsibility (e.g. timer management) is put on the programmer.
- ❖ User has to manage queue
- ❖ Priority to task is not given

EmbeddedCraft

IMBUENT

35

RTOS Solution...

Three tasks

First Task -----

```
void checkMotorSwitches(void)
{
    while (TRUE) {
        pause(100L);
        for (i = 0; i < 3; i++) {
            if (switchChanged(i)) changeMotor(i);
        }
    }
}
```

Second Task -----

```
void checkPressure(void)
{
    while (TRUE) {
        pause(50L);
        if (pressure() > 100) {
            closeValve();
            while (TRUE) {
                pause(50L);
                if (pressure < 90) {
                    openValve();
                    break;
                }
            }
        }
    }
}
```

Third Task

```
void checkDatagrams(void)
{
    typeMsg * msg;
    while (TRUE)
    {
        msg = waitMsg(DATA_GRAM);
        processDataGram(msg);
        freeMsg(msg);
    }
}
```

EmbeddedCraft

IMBUENT

36

RTOS Programming

RTOS Solution (Ex: Embedded Linux with RTAI Support)

```
#include <rtai_sched.h>

Void main()
{

    RT_TASK *motor; /*declare motor task*/
    RT_TASK *pressure; /*declare motor pressure*/
    RT_TASK *datagram; /*declare motor datagram*/

    rt_task_init ( &motor,checkMotorSwitches , 0, 512/*stacksize*/, 1/*priority*/, 0, 0);
    rt_task_init ( &pressure, checkPressure, 0, 512, 2, 0, 0);
    rt_task_init ( &datagram, checkPressure, 0, 512, 3, 0, 0);

}
```

EmbeddedCraft

IMBUENT

37

RTOS Solution (Ex: Embedded Linux with RTAI and POSIX)

```
#include <pthread.h>

struct sched_param { int sched_priority; /* Scheduling priority */ };
void main()
{
    pthread_t motor, pressure, datagram;
    param.sched_priority = 1;
    pthread_setschedparam(motor, SCHED_FIFO, param);
    param.sched_priority = 2;
    pthread_setschedparam (pressure, SCHED_FIFO, param);
    param.sched_priority = 3;
    pthread_setschedparam (datagram, SCHED_FIFO, param);

    pthread_create(&motor, NULL, checkMotorSwitches, 0);
    pthread_create(&pressure, NULL, checkPressure, 0);
    pthread_create(&datagram, NULL, checkPressure, 0);
}
```

EmbeddedCraft

IMBUENT

38

RTOS Programming

Advantages

1. Busy waiting is eliminated.
2. Timer management is no longer a concern of the programmer.
3. *checkPressure* does not have to retain a state variable.
4. Queue development and management is no longer a concern of the programmer.

5. Kernel API

waitMSG()	Pause()
freeMSG()	rt_task_init()

EmbeddedCraft

IMBUENT

39

RTOS Important Features

- ❖ Task scheduling is priority based preemptive
- ❖ Interrupt latency is almost zero
- ❖ Kernel should robust (stable)
- ❖ Tasks should be protected
- ❖ High resolution timer (1 microsecond)
- ❖ Intertask communication
 - Semaphore
 - Mail box
 - Shared Memory

EmbeddedCraft

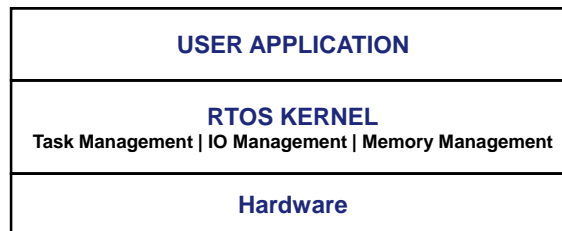
IMBUENT

40

RTOS Programming

Where is kernel placed

- ❖ Kernel resides between user's application and hardware
- ❖ Application will communicate with kernel through API



Conclusion 1/2

- ❖ Any application can also be designed without RTOS
 - But development time will be unpredictable
- ❖ Using RTOS means we are focusing on our application code, rest of things like task management, scheduling will be managed by RTOS
- ❖ Several middleware's are available with RTOS like
 - TCP/IP stack
 - USB Stack
 - Graphic Stack
 - Bluetooth Stack

RTOS Programming

Conclusion 2/2

- ❖ Companies also offer certifiable kernel !!!
 - you need to certify your code if application of critical domain
 - Avionics, Medical, Industrial etc

- ❖ RTOS study and learning is a matter of
 - Devoting more time
 - Having patience
 - Getting experience while working

EmbeddedCraft

IMBUENT

43

Our offering

- ❖ **Embedded Linux Training Program**
 - ARM9 Based Development Board (CS9302)
 - Linux kernel 2.6.x
 - POSIX Threads and Process
 - Accessing Peripherals
 - GPIO, USB, Ethernet etc
 - Eclipse, GCC ARM tools are used
 - Real Time API

Visit to us at

<http://www.imbuent.com/traininglinux.html>

EmbeddedCraft

IMBUENT

44

RTOS Programming

Live Demo

Demonstration of Embedded Linux

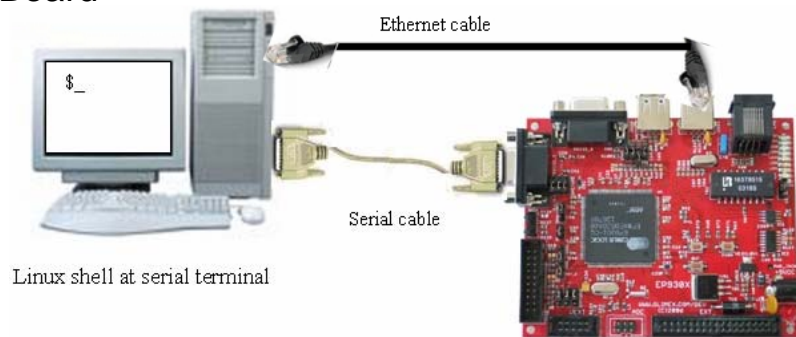
EmbeddedCraft

IMBUENT

45

Setup

- ❖ Demonstration of Embedded Linux on SPJ9302 Board



Linux shell at serial terminal

EmbeddedCraft

IMBUENT

46

RTOS Programming

Reference

❖ **Embedded Linux**

[.http://www.embeddedcraft.org/elinixtutorials.html#top](http://www.embeddedcraft.org/elinixtutorials.html#top)

❖ **Minicom: Serial Terminal In Linux**

[.http://www.embeddedcraft.org/mincominlinux.html#top](http://www.embeddedcraft.org/mincominlinux.html#top)

❖ **Difference Between Embedded Linux and Desktop linux**

[.http://www.embeddedcraft.org/embedlinuxdesktoplinux.html](http://www.embeddedcraft.org/embedlinuxdesktoplinux.html)

❖ **RTOS Presentation Part 1**

[.http://www.embeddedcraft.org/RTOS%20Part1.pdf](http://www.embeddedcraft.org/RTOS%20Part1.pdf)

❖ **RTOS Presentation Part 2**

<http://www.embeddedcraft.org/RTOS%20Part2.pdf>

EmbeddedCraft

IMBUENT

47


IMBUENT
Today's Solutions
Tomorrow's Technologies

EmbeddedCraft
crafting of intelligent systems

Thank You

Logos and brand names used in this presentation are belonging to their respected owners. We have used them here only for the purpose of information