International Journal of Instrumentation Control and Automation

Volume 2 | Issue 1 Article 14

April 2012

# DESIGN & DEVELOPMENT OF REAL-TIME MULTITASKING MICROKERNEL BASED ON ARM7TDMI FOR INDUSTRIAL AUTOMATION.

SHAKTIRAJ KUMAR CHAGANTY
*Department of Electrical and Electronics Engineering, J.B. Institute of Engineering and Technology (Autonomous), Hyderabad, India.*, shakti456@gmail.com

B. LAVAN
*Department of Electrical and Electronics Engineering, J.B. Institute of Engineering and Technology (Autonomous), Hyderabad, India.*, lavan.bachu569@gmail.com

S.SIVA PRASAD
*Department of Electrical and Electronics Engineering, J.B. Institute of Engineering and Technology (Autonomous), Hyderabad, India.*, sshivprasadeee@gmail.com

Recommended Citation
CHAGANTY, SHAKTIRAJ KUMAR; LAVAN, B.; and PRASAD, S.SIVA (2012) "DESIGN & DEVELOPMENT OF REAL-TIME MULTITASKING MICROKERNEL BASED ON ARM7TDMI FOR INDUSTRIAL AUTOMATION.," *International Journal of Instrumentation Control and Automation*: Vol. 2 : Iss. 1 , Article 14.
DOI: 10.47893/IJICA.2012.1071
Available at: https://www.interscience.in/ijica/vol2/iss1/14

# DESIGN & DEVELOPMENT OF REAL-TIME MULTITASKING MICROKERNEL BASED ON ARM7TDMI FOR INDUSTRIAL AUTOMATION.

**SHAKTIRAJ KUMAR CHAGANTY[1], B.LAVAN[2], DR.S.SIVA PRASAD[3]**

[1,2,3]Department of Electrical and Electronics Engineering, J.B. Institute of Engineering and Technology (Autonomous), Hyderabad, India.
E-mail- [1]shakti456@gmail.com, [2]lavan.bachu569@gmail.com, [3]sshivprasadeee@gmail.com

**Abstract**- A real-time microkernel is the near-minimum amount of software that can provide the mechanisms needed to implement a real-time operating system. Real-time systems are those systems whose response is deterministic in time. In our research a 32-task Real Time Microkernel is designed using which multi tasking can be done on the targeted processor ARM7TDMI. Two sets of functions are developed in this research work. First one is Operating System functions and second is application functions. Operating System functions are mainly for carrying out task creation, multi-tasking, scheduling, context switching and Inter task communication. The process of scheduling and switching the CPU (Central Processing Unit) between several tasks is illustrated in this paper. The number of application functions can vary between 1 to 32. Each of these application functions is created as a task by the microkernel and scheduled by the pre-emptive priority scheduler. Multi tasking of these application tasks is demonstrated in this paper.

***Keywords***- *ARM7, Multi-tasking, kernel, semaphore, rtos.*

## I. INTRODUCTION

A real-time microkernel is the near-minimum amount of software that can provide the mechanisms needed to implement a real-time operating system. These mechanisms include low-level address space management, thread management, and inter-process communication (I.P.C). As an operating system design approach, microkernels permit typical operating system services, such as device drivers, protocol stacks, file systems and user interface code to run in user space.

A 32-task Real Time Microkernel is developed for ARM7TDMI processor which is part of the LPC2138 microcontroller. The micro kernel includes a preemptive priority scheduler and context switching modules for carrying out multi-tasking. Routines to create and manage tasks are developed. Once created, the tasks are scheduled by this scheduler automatically. Subsequently, inter task communication mechanism is added to this scheduler, to make it a small real-time kernel.

Two sets of functions are developed in this paper. First one is Kernel functions and second is application functions. Kernel functions are mainly for carrying out task creation, multi-tasking and Inter task communication. The number of application functions can be from 1 to 32. Each of these application functions is created as a task by the microkernel and scheduled by the pre-emptive priority scheduler. Following section explains the responsibilities of the various functions implemented in the Kernel along with the function prototypes of the Kernel functions.

## II. EMBEDDED SYSTEM

An Embedded System is a special-purpose system in which the computer is completely dedicated to the device or system it controls. Unlike a general-purpose computer, such as a personal computer, an embedded system performs one or a few predefined tasks, usually with very specific requirements. Since the system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product. Embedded systems are often mass-produced, benefiting from economies of scale.

Physically, embedded systems ranges from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants[1-2].

In terms of complexity embedded systems can range from very simple with a single microcontroller chip, to very complex with multiple units, peripherals and networks mounted inside a large chassis or enclosure [3].

## II. ARM7TDMI

ARM (Advanced RISC Machine) processors are meant for particularly high end applications which involve more complex computations. ARM has good speed/power consumption ratio and high code density as needed by an embedded system.

The ARM7TDMI processor employs a unique architectural strategy known as THUMB, which makes it ideally suited to high-volume applications with memory restrictions, or applications where code density is an issue. The key idea behind THUMB is that of a super-reduced instruction set [4-5]. The ARM core uses RISC architecture. RISC is a design philosophy aimed at delivering simple but powerful instructions that execute within a single cycle at a high clock speed.

The RISC philosophy concentrates on reducing the complexity of instructions performed by the hardware because it is easier to provide greater flexibility and intelligence in software rather than hardware. As a result, a RISC design places greater demands on the compiler.

In contrast, the traditional complex instruction set computer (CISC) relies more on the hardware for instruction functionality, and consequently the CISC instructions are more complicated.

## III. MULTI TASKING

Multitasking system executes multiple tasks on only one CPU. The CPU switches from one task to other so quickly that an illusion of simultaneous execution is created. Multitasking is the process of scheduling and switching the CPU (Central Processing Unit) between several tasks; a single CPU switches its attention between several sequential tasks. Multitasking maximizes the utilization of the CPU and also provides for modular construction of applications. One of the most important aspects of multitasking is that it allows the application programmer to manage complexity inherent in real-time applications. Application programs are typically easier to design and maintain if multitasking is used.

## IV. MICROKERNEL

A microkernel is the near-minimum amount of software that can provide the mechanisms needed to implement an operating system. These mechanisms include low-level address space management, thread management and Inter Process Communication (IPC) [6].

In terms of source code size, microkernels (as a rule of thumb) tend to be under 10,000 lines of code. MINIX3 for example has around 4,000 lines of code. Kernels larger than 20,000 lines are generally not considered microkernels. As an operating system design approach, microkernels permit typical operating system services, such as device drivers, protocol stacks, file systems code, to run in user space. If the hardware provides multiple rings or CPU modes, the microkernel is the only software executing at the most privileged level generally referred to as supervisor or kernel model [7-8].
Microkernel functions Implemented -
All the functions implemented in this research work are classified as follows

- Task related functions for task creation, priority changing, deletion etc.
- Scheduler for scheduling using priority pre-emptive feature
- Context switch to swap two tasks
- Semaphore related functions for intertask communication

Scheduling is a key concept in multi processing operating system and real time operating system designs. Scheduling refers to the way processes are assigned to run on the available CPUs, since there are typically many more processes running than there are available CPUs.

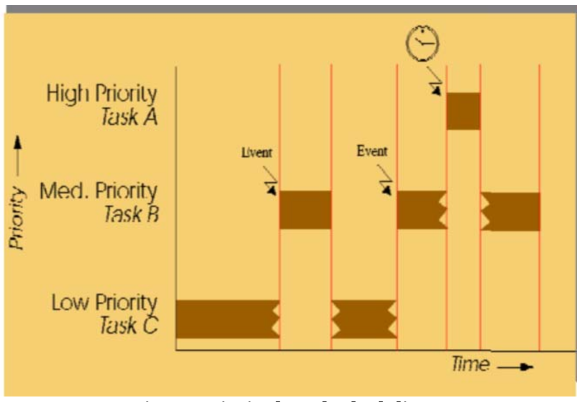


**Fig.1 – Priority based scheduling.**

A. Scheduler void thread_schedule (void) Runs the thread scheduler.

The job of the scheduler is: Scan entire TCB and compare the status and priority of all the tasks with the current task running. A task must be created by the OS before it can be scheduled for execution. Task will be always in one the three states namely READY, WAIT, RUN. User program can change priority of task. Always highest priority ready task runs.

If there is any other task which is ready to run and has higher priority than the current task, make the current task ready and new task as running. Call the context switch routine to swap the register contents. In summary the job of the scheduler is to run the highest priority ready task. Context switching is the computing process of storing and restoring of a CPU so that execution can be resumed from the same point at a later time. This enables multiple processes to share a single CPU. The context switch is an essential feature of a multitasking operating system.

B. Semaphore-

Semaphore is an object used for intertask communication in operating system [9]. It is for informing the kernel about the status of the task for its waiting for a resource or releasing a resource.

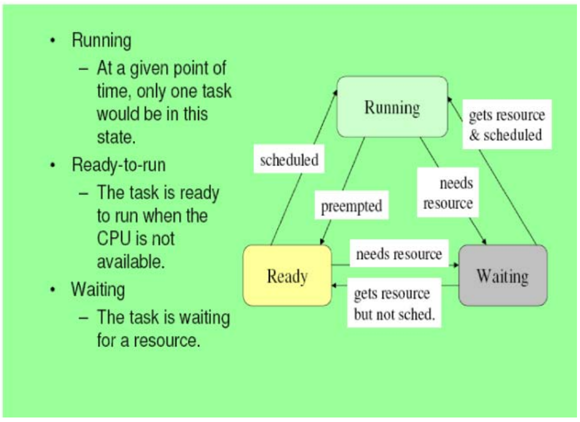

Running
– At a given point of time, only one task would be in this state.
Ready-to-run
– The task is ready to run when the CPU is not available.
Waiting
– The task is waiting for a resource.
Running
gets resource & scheduled
scheduled
preempted
needs resource
needs resource
Ready
Waiting
gets resource but not sched.

**Fig.2- Task States**

## VI. TOOLS AND HARDWARE BOARD USED FOR DEMONSTRATION-

LPC2138 ARM microcontroller, Keil C compiler, flash magic programmer are used for conducting the experiment[10].
LPC2138

The LPC2138 microcontrollers are based on a 32 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory of 512 kB.

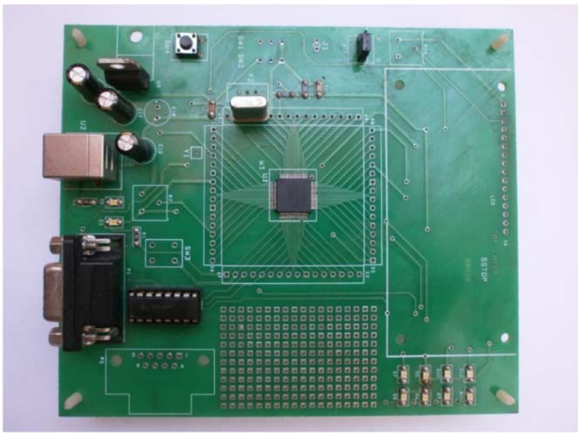


**Fig.3 Embedded board based on LPC2138**

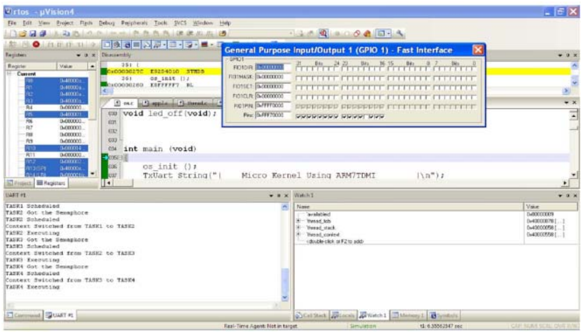
A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at the maximum clock rate [11]. For critical code size applications, the alternative 16-bit Thumb mode reduces the code by more than 30 % with minimal performance penalty.

## VI. EXPERIMENTAL RESULTS

The results of this research work are shown in Fig.4 using the Keil uVision simulator. Total 8 tasks are created and scheduled. Each task sends different message to the UART terminal0 which can be observed in the Debugger environment of the tool. Once a task is completed it posts a semaphore for another task and waits for another semaphore indefinitely. Whenever it gets a semaphore it gets scheduled. The context switching and scheduling functions are implemented in C and ARM assembly language. Along with this function two application tasks are developed.

Task 1: Sends display data related to Taks1 on UART0, blinks LEDs, posts a semaphore for task2
Task 2: Waits for the semaphore, and sends different data related to task2 on UART0 and toggles the LED status.

The simulation output using Keil Microvision tools are as follows-

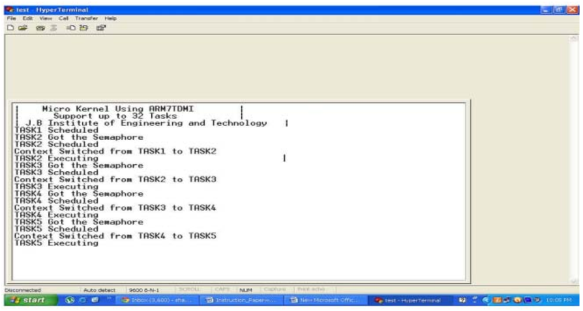


```
void led_off(void);

int main (void)
{
    os_init ();
    TxUart String("|    Micro Kernel Using ARM7TDMI          |\n");
```

```
TASK1 Scheduled
TASK2 Got the Semaphore
TASK2 Scheduled
Context Switched from TASK1 to TASK2
TASK2 Executing
TASK3 Got the Semaphore
TASK3 Scheduled
Context Switched from TASK2 to TASK3
TASK3 Executing
TASK4 Got the Semaphore
TASK4 Scheduled
Context Switched from TASK3 to TASK4
TASK4 Executing
```

**Fig.4 Simulation results in Keil window.**



```
|    Micro Kernel Using ARM7TDMI        |
|      Support up to 32 Tasks           |
| J.B Institute of Engineering and Technology   |
TASK1 Scheduled
TASK2 Got the Semaphore
TASK2 Scheduled
Context Switched from TASK1 to TASK2
TASK2 Executing
TASK3 Got the Semaphore
TASK3 Scheduled
Context Switched from TASK2 to TASK3
TASK3 Executing
TASK4 Got the Semaphore
TASK4 Scheduled
Context Switched from TASK3 to TASK4
TASK4 Executing
TASK5 Got the Semaphore
TASK5 Scheduled
Context Switched from TASK4 to TASK5
TASK5 Executing
```

**Fig.5 Results demonstrated on the test board based on LPC2138**

The same results are demonstrated on the test board based on LPC2138 are show in Fig.5. Flash Magic tool used for loading the software in the microcontroller board.

## VII. CONCLUSION

A real time microkernel for ARM7TDMI is successfully developed and demonstrated in this research work. The tools used are Keil uVision for Software development and debugging, Flash Magic is used for flash loading. Microcontroller LPC2138 is used. Microkernel developed in this research work can be used for multi-tasking. ARM architecture is a very promising venture for the present and the future. ARM has found wide acceptance among the mobile device manufacturers with more that 98% devices being shipped having at least one ARM core. At least 90% of the embedded 32 bit processors are based on ARM. The application areas include Industrial controllers for remote machine operation, cooling mechanism monitoring systems in industries, security monitoring systems, medical equipment, home automation products, flight instrument panel and industry process automation.

## ACKNOWLEDGMENT

Authors are deeply indebted to the Management & Principal of J.B.Institute of Engineering and Technology (Autonomous) and also Mr. C. Ramesh Reddy, Scientist –D, RCI, DRDL, Ministry of Defense, Hyderabad for their cooperation during this research work.

## REFERENCES

[1] Edwards S, Lavagno L, Lee E.A. "Design of Embedded systems: Formal modules, validation and synthesis" published in volume 85, issue 3 of IEEE transactions of Computer design,Pg. 366-390.

[2] Raghunathan V, "Advanced techniques for programming networked Embedded Systems" published in IEEE conference on 7-11 Jan. 2012. Pg.36-37.

[3] Wilhelm,R, Grund D, Reineke J, "Memory hierarchies, pipelines and buses for future architectures in time critical embedded systems" appears in IEEE transactions on integrated circuits and systems. Date of Publication: July 2009.

[4] http://www.arm.com/ products/ processors/ classic/ arm7/ index.php

[5] http://www.engineersgarage.com/articles/arm-advanced-risc-machines-processors

[6] Nanjing Xiaozhuang Coll., Nanjing "Design of Embedded OS Micro-kernel Experiment Series" presented in International conference of Electrical and Computer Engineering, Date of Conference: 11-13 Aug. 2012.

[7] Jianjun Shen Sihan Qing; Qingni Shen "Design Of A Micro-Kernel Based Secure System Architecture" Presented In Information Assurance Workshop, 2006 Ieee Date Of Conference: 21-23 June. 2006.

[8] Haiying Zhou , Kun Mean Hou, Christophe De Vaulx "A Super Small Distributed REAL-Time Microkernel Dedicated to Wireless Sensors" published in International Journal of Pervasive Computing and Communications in Vol.2 Issue 4

[9] Chiao-Jang Wu; Hsiou-Mien Lien; I-Neng Chen "Design And Implementation Of A Distributed Semaphore Facility" –Presented In Distributed Computing Systems, 2009., Proceedings Of The Third Workshop On Future Trends Date Of Conference : 14-16 April 2009

[10] Puaut, I. Puaut, I. "Low-complexity algorithms for static cache locking in multitasking hard real-time systems" Published in Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE Page(s):114 – 123 ISSN :1052-8725

[11] http://www.nxp.com/ documents/ data_sheet/ LPC2131_32_34_36_38.pdf (data sheet)

◈◈◈