ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

Development of Microkernel for Multitasking with ARM11

C. Naga Srikanth[1], M. Veda Chary[2], M. Sudhakar[3]
[1] M. Tech (ES) Final Year, 2 Assoc. Professor, [3] Professor & Vice Principal,
Dept. of ECE, CMR College of Engineering & Technology, Hyderabad, Andhra Pradesh

Abstract: - A 32-task Real Time Microkernel is designed for multi tasking on the targeted processor ARM1176JZFS. The Micro kernel developed includes a preemptive priority scheduler and context switching modules for carrying out multi-tasking. Tasks which are created will be scheduled by the priority scheduler automatically. Subsequently, inter task communication mechanism is added to this scheduler, to make it a small real-time kernel. Routines to create and manage tasks are developed and tested. The results are reported and discussed.

***Keywords*- Context switching, Microkernel, Scheduler, Semaphore.**

I. INTRODUCTION

A general Operating System occupies large memory space, difficult to customize, UN deterministic, and cannot respond to real time events. The Real Time Microkernel will resolve these difficulties and has deterministic interrupt latency. 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, micro kernel permits typical operating system services, such as device drivers, which are used to access the hardware resources, protocol stack to access the various services to access the network devices, file systems to manage and provide access control on various files and folders, and user interface code to run in user space. A32-task Real Time Microkernel is developed for ARM1176JZFS. The micro kernel includes a preemptive priority scheduler to accommodate 32 tasks and context switching modules for carrying out multi-tasking.

Two sets of functions are developed. First set is Kernel functions which are used for carrying out task creation, multi-tasking and Inter task communication. Second set is application functions which run on top of the kernel functions. Each application function is created as a task by the microkernel and scheduled by the pre-emptive priority scheduler.

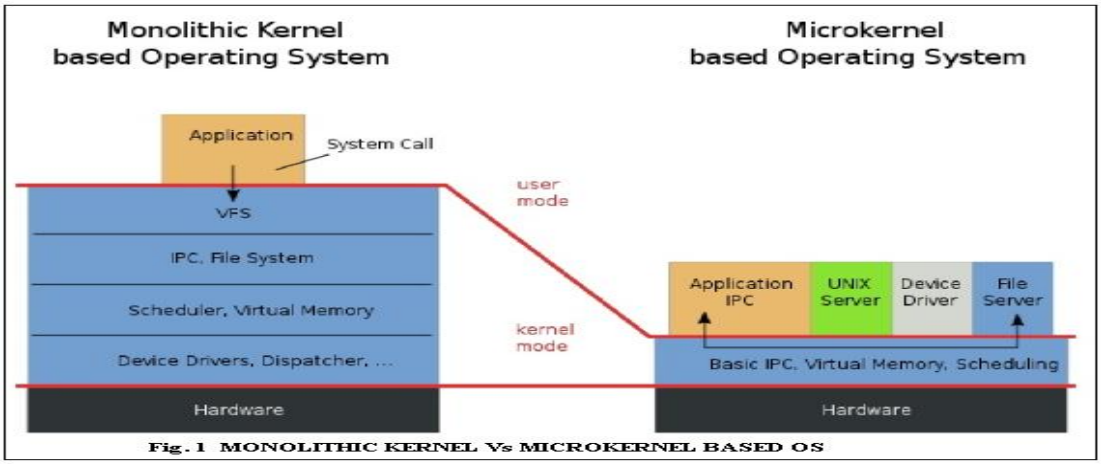

Fig. 1 MONOLITHIC KERNEL Vs MICROKERNEL BASED OS

II. ARM1176JZFS PROCESSOR

The ARM1176JZF-S processor incorporates an integer core that implements the ARM11 ARM architecture v6. It supports the ARM and Thumb™ instruction sets, Jazelle technology to enable direct execution of Java byte codes and a range of SIMDDSP instructions that operate on 16-bit or 8-bit data values in 32-bit registers. Functional block diagram is as shown in fig. 2

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

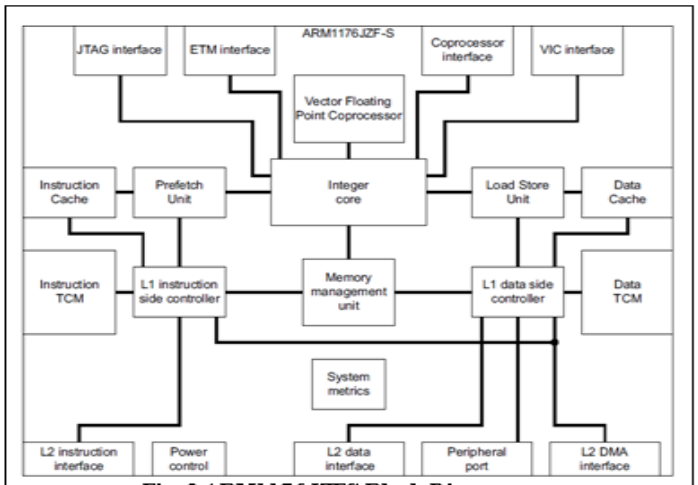

Fig. 2 ARM1176JZFS Block Diagram

III. MICROKERNEL DEVELOPMENT SCHEME

The components namely task management & scheduling and Inter task communication synchronization of microkernel are developed in this paper. The Context Switching services available with the operating system are extracted and employed with the Microkernel.

A. Task Management and Scheduling

Task (or "process", or "thread") management is a primary job of the operating system: tasks must be created and deleted while the system is running; tasks can change their priority levels, their timing constraints, their memory needs; etc. Task management for an RT Kernel is a bit more critical than for a general purpose OS: if a real-time task is created, it has to get the memory it needs without delay, and that memory has to be locked in main memory in order to avoid access latencies due to swapping; changing run-time priorities influences the run-time behavior of the whole system and hence also the predictability which is so important for an RTOS. So, dynamic process management is a potential problem for an RTOS, therefore not recommended, though it is implemented in this paper.

In general, multiple tasks will be active at the same time, and the OS is responsible for sharing the available resources (CPU time, memory, etc.) over the tasks. The CPU is one of the important resources, and deciding how to share the CPU over the tasks is called "Scheduling". The general trade-off made in scheduling algorithms is between, on the one hand, the simplicity (and hence efficiency) of the algorithm and on the other hand, its optimality. Real-time operating systems favor simple scheduling algorithms, because these take a small and deterministic amount of computing time, and require little memory footprint for their code.

General purpose and real-time operating systems differ considerably in their scheduling algorithms. They use the same basic principles, but apply them differently because they have to satisfy different performance criterions A general purpose OS aims at maximum average throughput, a real-time OS aims at deterministic behavior. In this paper a dynamic priority pre-emptive scheduling algorithm is implemented. The prototypes of the Task Management and scheduling related functions implemented are described here.

osrc_t thread_create (int 32*ptid, int32 pri, void (*pfunc)(void))

Creates a thread with the entry point pointed to by pfunc, and makes it ready to run. If the priority is higher than the current thread's priority, it schedules the created thread. Returns OK if successful, or ERR on error.

void thread_suspend (int32 tid)

Suspends the specified task (puts it in the wait state).

void thread_resume (int32 tid)

Resumes the specified task (puts it in the ready/running state depending on the priority).

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

int32 thread self (void)
Returns the id of the current task.

int32 thread_getpri (int32 tid)
Returns the priority of the specified task.

myrc_t thread_setpri (int32 tid, int32 pri)
Sets the priority of the specified task. Additionally, performs scheduling if required due to the change in the task priority.

void os_init (void)
Initializes the common OS data structures. Calls thread_init for thread specific initializations. Later, this function will call other object initialization functions.

void thread_init (void)
Initializes thread specific data structures
void thread_schedule (void)
Runs the thread scheduler.

void thread_switch_context (int32 *pcurcxt, int32 *pnxtcxt) (Assembly code)
Internal function to do the context switch. Takes in the pointers to the arrays containing contexts of the current and new task. Swaps these contexts and returns to new task.

B. Communication and Synchronization
Second major responsibility of an OS is Inter-Process Communication (IPC). "Process", in this context, is a "task". The IPC collects a set of programming primitives that the operating system makes available to tasks that need to exchange information with other tasks, or synchronize their actions. In this paper Inter task communication is carried out using semaphores. Semaphore is an object used for Inter Task Communication in operating system. It is for informing the kernel about the status of the task for its waiting for a resource or releasing a resource. Every semaphore must be created before it is used. Below are the prototypes of the Inter Task Communication Synchronization related functions implemented in the paper.

osrc_t ossem_create (int32 count, int32 *psid)
Creates a semaphore with the specified initial count and returns the id in *psid.
Returns OK if successful, or ERR on error.

void sem_post (int32 semid)
Posts the specified semaphore. Reschedules the tasks if posting the semaphore could cause a task switch.

void sem_wait (int32 sid)
Waits for a semaphore. If the semaphore is unavailable, puts the current task in waiting state and schedules another task

C. Context Switching
Context switching is done when a processor switches from one task to another task. Context is mainly the Register snapshot of the processor when a task is under execution. This function is called after the scheduler if task switching is required. In this paper this function is implemented using ARM assembly language programming. This routine saves all the registers of the first task on the stack of the task or context area and restores all the registers from the stack or context area of the next task. At the end of this task the control switches to second task

void thread_switch_context (int32 *pcurcxt, int32 *pnxtcxt) (Assembly code)
This function Takes in the pointers to the arrays containing contexts of the current and new task.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

Swaps these contexts and returns to new task.

IV. FLOW CHART

The flow chart to create and manage the several tasks by Micro kernel is shown in fig. 3

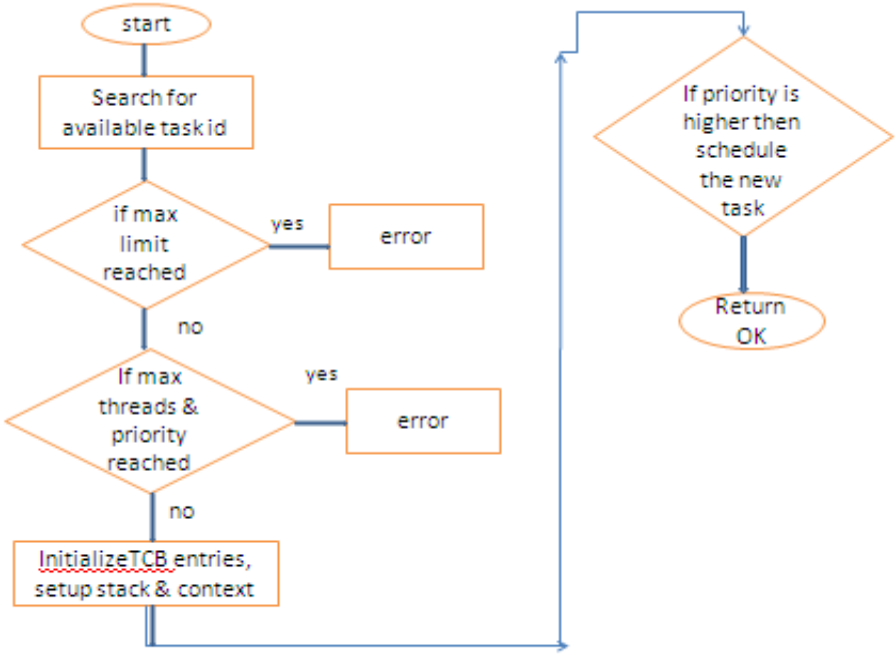

Fig. 3 FLOW CHART FOR TASK CREATION

V. RESULTS

Performance results of 8 tasks considered with the scheme are shown is figure 4, 5 and 6

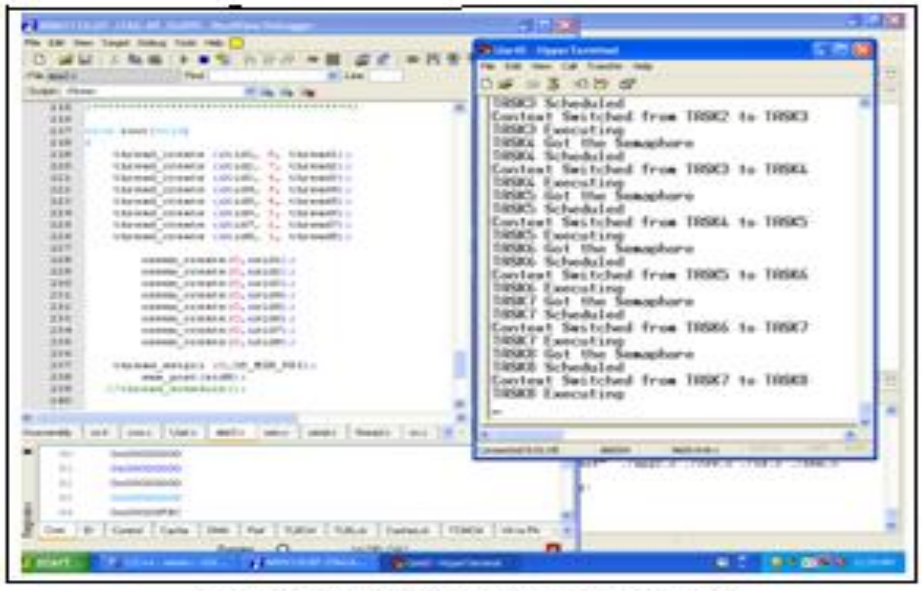

Fig. 4. UART INITIALISATION

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

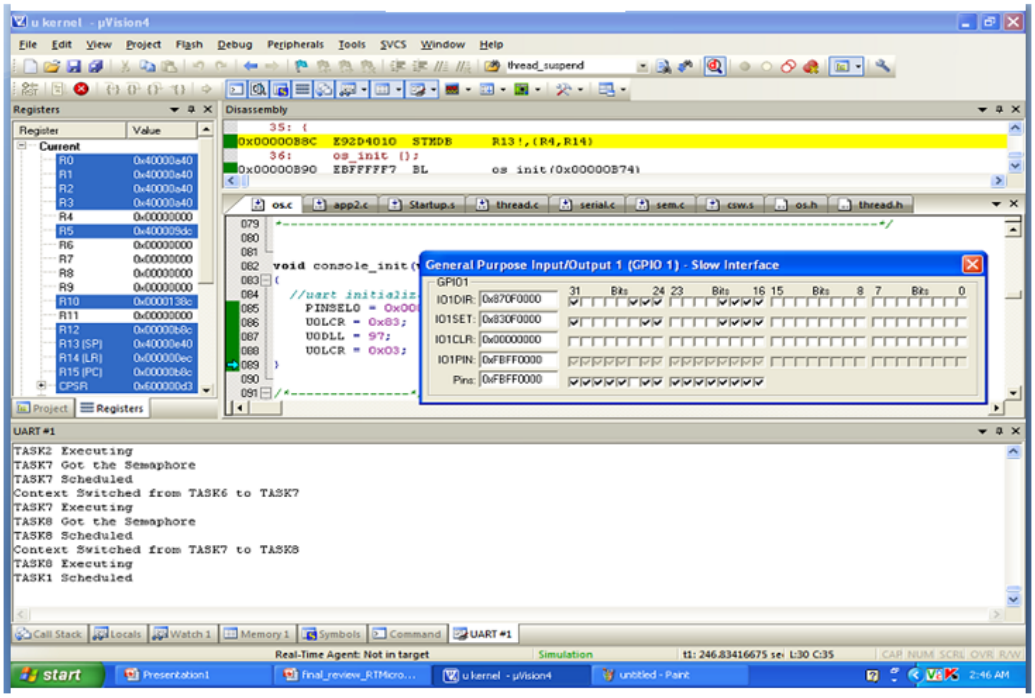

Fig. 5 LED BLINKING

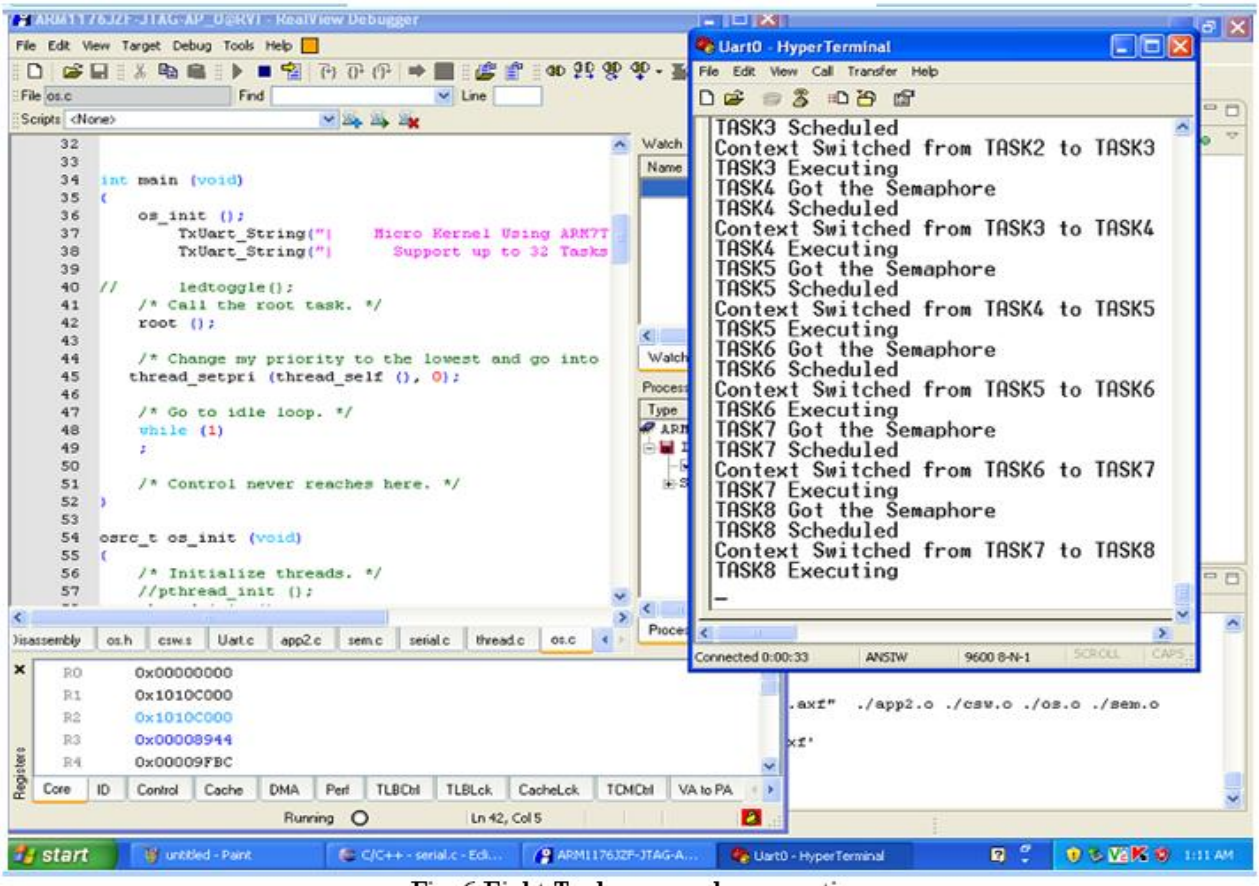

Fig. 6 Eight Tasks are under execution

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

VI. CONCLUSION

A real time microkernel has been developed for ARM11with the **Real view** tool, which is used for software development and debugging. Flash Magic is used for flash loading. The developed micro kernel can used for multi-tasking up to 32 tasks in the areas of avionics, on board computers industry, process automation and auto mobiles. These application functions can be replaced with any applications as per the requirements.

REFERENCES

[1] http://infocenter.arm.com/help/topic/com.arm.doc.ddi0183g/DDI0183G_uart_pl011_r1p5_trm.pdf.

[2] http://infocenter.arm.com/help/topic/com.arm.doc.ddi0301h/DDI0301H_arm1176jzfs_r0p7_trm.pdf.

[3] http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0190b/I1002697.html.

[4] http://infocenter.arm.com/help/topic/com.arm.doc.ddi0190b/DDI0190.pdf.

[5] http://infocenter.arm.com/help/topic/com.arm.doc.DDI0190B_gpio_PL061_trm.

[6] http://infocenter.arm.com/help/topic/com.arm.doc.DUI0425F_realview_platform_baseboard_for_arm1176jzf_ug.

[7] Real-time systems by Jane W.S.Liu.

[8] **www.csie.ntu.edu.tw/~ktw/rts/uCOSII- prn.pdf.**

[9] **www.ucos-ii.com.**

ACKNOWLEDGEMENT

The authors thank Dr. MBR Murthy, Professor, Department of ECE, and CMR College of Engg. & Technology, Hyderabad, AP. for his valuable suggestion

AUTHOR BIOGRAPHY

C Naga Srikanth: He is graduated (B.Tech) from Narasaraopeta Engineering college, Narasaraopet in the year 2010 with the specialization of ECE. Later He is doing (M.Tech) from JNTU, Hyderabad in the year 2012.

M Veda Chary: He is a graduated (B. Tech) from Kakatiya Institute of Technology and Sciences, Warangal in the year 2004 with the specialization of EIE. Later completed Post Graduation (M.Tech) from JNTU, Hyderabad in the year 2011 with the specialization of Embedded Systems.

M Sudhakr: He is graduated (B.Tech) from JNTU College of Engineering, Hyderabad in the year 1979, with the specialization of ECE. Later completed his post graduation (M.Tech) from Indian Institute of Technology, Madras in the year 1986 with the specialization of Instrumentation, Control & Guidance. He also did his PG Degree in Aeronautical Engineering (Electronics) from Air Force Technical College, Bangalore in the year 1981. Presently pursuing his research in "Intelligent and Adaptive Control Systems, in JNTU Hyderabad. Completed R&D Project assigned by IAF on 'Mathematical Modeling & Simulation of Aero Engine Control System' at Aeronautical Development Establishment, Bangalore and Gas Turbine Research Establishment, Bangalore for a period of 2 years.