Real Time Operating System based on AVR Microcontroller

Srishti Dubey
Electronics and Communication
Medi-Caps Institute of Engineering & Technology
Indore, India

Devendra Singh Bais
Assistant Professor
Electronics and Communication
Medi-Caps Institute of Engineering & Technology
Indore, India

Ankit Chouhan
Electronics and Communication
Medi-Caps Institute of Engineering & Technology
Indore, India

Abstract

This paper describes about a compact and efficient Real Time Operating System (RTOS) based on AVR microcontrollers. By using RTOS, it can result to eliminate processor waiting time, without doing any applicable work. Lots of tasks can be run independently and simultaneously and due to this CPU's efficiency will be more than conventional systems. RTOS is pre-emptive and multitasking. The design has a small code size, good performance and low memory usage, as the design was implemented for AVR devices. Finally, practical algorithm with suitable circuit and ATmega32 is presented to test this information about the designed RTOS.

Keywords- *RTOS, ATmega32, Pre-emptive.*

1. Introduction

An operating system (OS) is software that uses all the hardware such as RAM, ROM, CPU, and so on for better management and controlling them. Generally, based on the running programs, there are two operating systems: single-task operating system and multi-task operating systems. In multi-task OS there are too many tasks in the system simultaneously; the OS divides processor among all of them. That kind of processor that divides times among them is called scheduler. There are two kinds of scheduling algorithms: pre-emptive algorithm and non pre-emptive algorithm.

Later, each task can be run after finishing another one, but in the former, scheduler can transform programming control from a running program to another one. Real time operating system (RTOS) is a kind of OS that each task must run in a certain times and if not, OS fails. So, based on the above mentioned points RTOS should be multi-task and pre-emptive for performing the scheduling mechanism. The main purpose in these kinds of OS is to be performing in a time limitation. There are many kinds of RTOS such as; µCOS, ECOS, QNX, Vxworks, Free RTOS, and so on. In this paper, RTOS for AVR microcontrollers has been designed by Atmel Company. It has been designed by C language and also it has been performed on ATmega32 microcontroller. For compiling its code, Code vision AVR has been used.

2. Introducing AVRs

Before First time, AVR technology has been introduced in 1997. Its design is based on RISC technology. It is having 32 general purpose registers: R0 to R31.

There are many kinds of AVR including LCD AVR, Mega AVR, Xmega AVR, Tiny AVR, and AT90S AVR. Like other microcontrollers, there are different languages for programming such as C and Assembly. There is a favourite compiler and language for programming is C and Code vision. For more information about AVRs, refer to [8].

2.1. [Real Time] Operating Systems Basics

An operating system is a system program which provides an interface between application programs and the computer hardware. There are two primary functions:

- Making the computer system convenient to use.
- To organize correct and efficient use of the computer Resources.

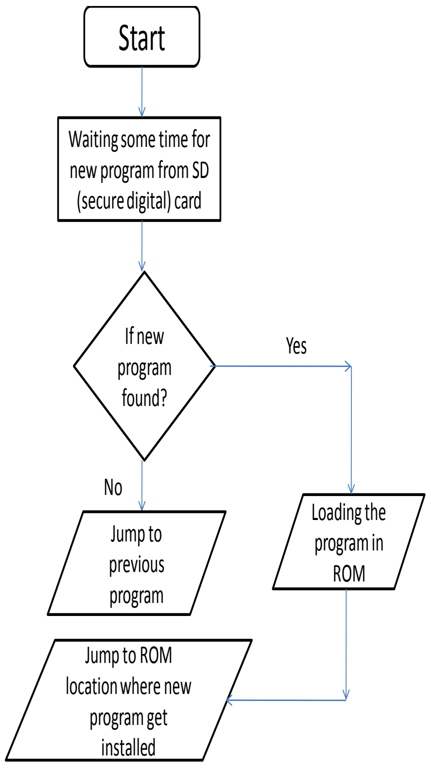
There are four main tasks of operating system:
Process management, inter-process communication and synchronization, memory management, and input/ output management. The process management component is also responsible for process loading, process creation and execution control, the interaction of the process with signal events, process monitoring, CPU allocation and process termination. Inter-process communication covers issues such as synchronization and coordination, process protection, deadlock and live-lock detection and handling and data exchange mechanisms. Memory management includes services for file creation, deletion, reposition, and protection. I/O management handles request and release the subroutines for a variety of peripherals, write, read, and reposition programs.
Real-time systems are those systems where the proper functionality assumes both the correctness of the output as well as the correct timing behaviour of the system.

3. Proposed plan for RTOS

Most of the designed system will perform the determined tasks after turning their power on. Those systems will use too much memory space and are somehow complex. Regarding low space memory in AVR and determined tasks without any need to change them in most cases, optimum plan will be presented as follow.

First all tasks with their requirement time to perform should be determined. A plan to compile tasks and OS has been designed. In fact, each task in the designed RTOS should use all hardware independently from other tasks. Tasks should be able to run simultaneously without any need to reprogramming for new codes. Users can change scheduling algorithm according to their systems. A timer can be used to generate the least timer tick according to the least time slice and based on it; scheduler can manage and control tasks according to their priority.

4. Flowchart


Start
Waiting some time for new program from SD (secure digital) card
If new program found?
Yes
No
Jump to previous program
Loading the program in ROM
Jump to ROM location where new program get installed

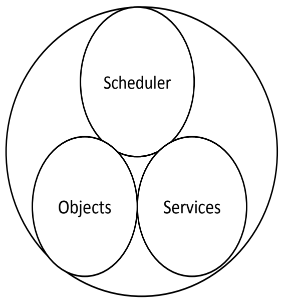
“Figure 1. Flowchart”

5. Real Time Operating System

An operating system is said to be real time, when it schedules the execution of programs in time, it handles system resources and gives a reliable basis for the development of software code.

5.1. Components of RTOS

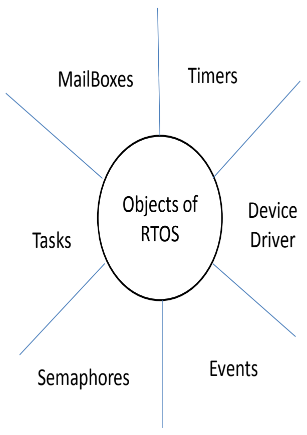
Most RTOS kernels consist of following components such as:-



“Figure 2. Normal Component of RTOS”

- Scheduler
- Objects
- Services

5.1.1. Scheduler. Scheduler is at center of each kernel. A scheduler allows the algorithms that are needed to determine what role do when.
5.1.2 Objects. The most common RTOS kernel objects are as follows:
• **Information ---** it is simultaneous and independent threads of execution that can compete for CPU execution time.
• **Semaphores ---**it is a token-like object that can be raised or charged by information for synchronization or mutual exclusion.
• **Message Queues ---**they are buffers that data structures that can be used, mutual exclusion, synchronization and communication by sending messages between tasks. [3]



.
"Figure 3. Objects of RTOS"

5.1.3 Services. Most of the kernels provide services to assist developers for creation of real-time embedded applications. These services comprise of API calls that are used to perform operations on kernel objects and can be used in general to facilitate the following services:

- Timer Management
- Interrupt Handling
- Device I / O
- Memory Management

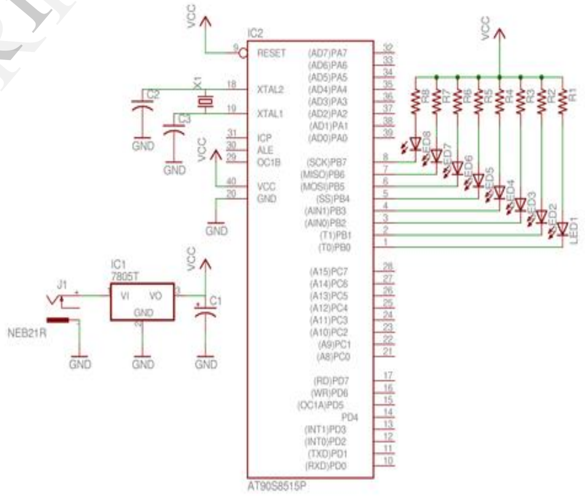
Embedded systems are used for many different applications. These applications can be proactive or reactive, depends on interface requirements, connectivity, scalability, etc. Selecting of OS for an embedded system is based on an analysis of operating system itself and the requirements of the application. [4]

6. How one RTOS differs from the other?

(i) RTOS differ in main architecture.
(ii) Types of scheduling algorithm used in it. (Pre-emptive or co-operative scheduling).
(iii) Number of instructions of kernel without any task written to it formed after compilation of complete code. It ultimately occupies space in ROM and RAM, so memory and execution speed get affected.
(iv) It can run no. of tasks without degrading the performance like response time. [9]
(v) Performance metrics that we have chosen i.e. Context switching Time, Pre-emption time and Interrupt Latency.

7. Hardware Design

ATmega32 is used for testing the designed RTOS. The circuit is as fig. 4. ATmega32 microcontroller is used in 16 MHz clock, Max232 i.e.) for serial peripheral interface with PC, capacitors and Led's.



"Figure 4. Circuit to test RTOS (STK500)"

8. Discussing Results

This paper has a proposed algorithm to design RTOS and it is also a RTOS for AVR devices which has been designed. Customary programs for microcontrollers use infinite loop and each task for running, needs to be run after finishing another one. So in case of such systems, the processor cannot do some tasks simultaneously. Using interrupts may seem like an option, but it also reduces the interrupt latency as the computation has to take place inside the ISR. Thus using RTOS is an applicable engineering solution. The main and important features of such RTOS rather than available

RTOS such as ECOS are its low memory usage and its small size. The results of proposed algorithm on practical circuit show that RTOS works in a proper manner. This RTOS may be used with all kinds of AVR controllers.

10. References

[1] Tran Nguyen,Bao Anh, "REAL-TIME OPERATING SYSTEMS FOR SMALL MICROCONTROLLERS", Published by the IEEE Computer Society, pp.30-46, September 2009.
[2] Baynes, K.; Collins, C.; Fiterman, E.; Brinda Ganesh; Kohout, P.; Smit, C.; Zhang, T.; Jacob, B.; , "The performance and energy consumption of embedded real-time operating systems," Computers, IEEE Transactions on , vol.52, no.11, pp. 1454- 1469, Nov. 2003 doi: 10.1109/TC.2003.1244943.
[3] Hessel, F.; da Rosa, V.M.; Reis, I.M.; Planner, R.; Marcon, C.A.M.; Susin, A.A.; , "Abstract RTOS modeling for embedded systems," Rapid System Prototyping, 2004. Proceedings. 15th IEEE International Workshop, pp. 210-216, 28-30 June 2004 doi:10.1109/IWRSP.2004.1311119
[4] M. L. So_a and M. J. Irwin, "Organizing principle for operating systems,"Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 49-60, March 2009.
[5] Tanenbaum, Andrew S.,Operating Systems: Design and Implementation
[6] G. Lipari and S. K. Baruah, "Efficient scheduling of real-time Multi-task applications in dynamic systems," 6th IEEE Real-Time Technology and Applications Symposium, 2000
[7] Atmel, http://www.atmel.com.
[8] ATmega32 datasheet, www.datasheetcatalog.com
[9] Edwards, Stephen A.; "Real-Time Embedded Software"; John Wiley & Sons, Inc.; Wiley Encyclopaedia of Electrical and Electronics Engineering; SN:9780471346081;2001;doi:10.1002/047134608X.W8113