Implementation of a Scalable μC/OS-II Based Multitasking Monitoring System

CH. S. L. Prasanna, M. Venkateswara Rao
1,2Dept. of ECM, KL University, AP, India

Abstract
Currently embedded monitoring devices have been widely used in many sectors especially in industrial area. However a large number of devices may not have a real time environment. This paper describes an embedded monitoring system based on μC/OS II RTOS operating system using ARM7. It deals with the porting of Micro C/OS-II kernel in ARM powered microcontroller for the implementation of multitasking and time scheduling. Here a real time kernel is the software that manages the time of a micro controller to ensure that all time critical events are processed as efficiently as possible. Different interface modules of ARM7 microcontroller like UART, ADC, LCD are used and data acquired from these interfaces is tested using μC/OS-II based real time operating system. This paper acts as a gateway to implement RTOS for high end applications.

Keywords
Embedded System, μC/OS-II, RTOS

I. Introduction
Sometimes devices may malfunction or totally fail due to long duration of usage or any technical problem which give fatal results.

An embedded monitoring system is necessary for continuously collecting data from site and later analyzing that and eventually taking proper measures to solve the problem. The systems that are in use today use non real time operating systems based on monostack mechanism that hardly satisfies the current requirements. This paper will focus on porting of μC/OS II in ARM7 controller that performs multitasking and time scheduling. The μC/OS II features and its porting to ARM7 are discussed. Finally it provides an overview for design of embedded monitoring system using μC/OS II as application software that helps in building the high end applications.

II. System Design

![System Structure Diagram](image)

The system structure is illustrated in the fig.1. It consists of two ARM7 based LPC2148 kits. One ARM7 based system with the sensors deployed is to be employed in the site of monitoring area. Temperature, IR, Humidity and Gas sensors are deployed. Based on the application requirement the sensors can be used. Another ARM based processor is interfaced with the user system. Zigbee module is inserted on both sides. It is used to transfer the data from remote location to the user system. A graphical user interface is used to visualize the incoming data.

The core model of the system is the ARM7 based LPC2148 processor. The board is made from double sided PTH PCB board to provide extra strength to the connector joints for increased reliability. It supports the operating supply voltage between 7V to 14V and has built-in reverse polarity protection. The ARM is operated in 32 bit mode. It has two 10 bit ADC’s and single 10 bit DAC. It has 2.4GHz zigbee wireless module adapter. It has an inbuilt LCD screen. Thanks to the diversity of the integrated resources, the design and debug of the core module become easy, also the reliability of the system rises.

LM35 temperature sensor is used. Its supply voltage is 4 volts. The hardware module of IR sensor contains a TX-RX pair. The IR sensor acts as an object counter here. SY-HS-220 is the humidity sensor used. MQ5 gas sensor is used. Its detection range is 100 ppm to 3000 ppm. The sensors sensed are transferred to the user system interfaced with ARM through zigbee. μC/OS II RTOS is ported for multitasking since we are measuring different parameters at a time. To measure and control all the parameters at once we use this. Moreover it provides the most critical data at first.

III. MICRO C/OS II RTOS
μC/OS II (pronounced “Micro C O S 2”) stands for Micro-Controller Operating System Version 2 and can be termed as μC/OS-II or uC/OS-II). It is a very small real-time kernel with memory footprint is about 20KB for a fully functional kernel and source code is about 5,500 lines, mostly in ANSI C. It’s source is open but not free for commercial usages. μC/OS-II is upward compatible with μC/OS V1.11 but provides many improvements, such as the addition of a fixed-sized memory manager; user-definable callouts on task creation, task deletion, task switch, and system tick; TCB extensions support; stack checking; and much more.

A. μC/OS II using ARM
μC/OS-II, The Real-Time Kernel is a highly portable, ROMable, scalable, preemptive real-time, multitasking kernel (RTOS) for microprocessors and microcontrollers. μC/OS-II can manage up to 250 application tasks. μC/OS-II runs on a large number of processor architectures and ports. The vast number of ports should convince that μC/OS-II is truly very portable and thus will most likely be ported to new processors as they become available. μC/OS-II can be scaled to only contain the features you need for your application and thus provide a small footprint. Depending on the processor, on an ARM (Thumb mode) μC/OS-II can be reduced to as little as 6K bytes of code space and 500 bytes of data space (excluding stacks). The execution time for most of the services provided by μC/OS-II is both constant and deterministic. This means that the execution times do not depend on the number of tasks running in the application. μC/OS-II is based on μC/OS which has been used in hundreds of commercial applications. μC/OS-II uses the same core and most of the same features as μC/OS yet offers more features.
**B. Starting µC/OS-II**

In any application µC/OS-II is started as shown in the fig. 2. Initially the hardware and software are initialized. The hardware is the ARM core and software is the µC/OS-II. The resources are allocated for the tasks defined in the application. The scheduler is started then. It schedules the tasks in preemptive manner. All these are carried out using specified functions defined in µC/OS-II.

**C. Initializing µC/OS-II**

µC/OS-II can be initialized as shown in the fig. 3. The detailed steps are shown in the figure. Below shows the sample program for the steps shown in the fig.

```c
void main (main)
{
    /* user initialization */
    OSlnit(); /* kernel initialization */
    /* Install interrupt vectors */
    /* Create at least 1 task (start task) */
    /* Additional User code */
    OSStart(); /* start multitasking */
}
```

**D. Task Creation**

To make it ready for multitasking, the kernel needs to have information about the task: its starting address, Top-f-stack (TOS), priority, arguments passed to the task, other information about the task. You create a task by calling a service provider by µC/OS-II:

You can create tasks:
- before you start multitasking (at initialization time)
- during run time.

**E. Implementation Through µC/OS-II**

In embedded systems, a Board Support Package (BSP) is implementation specific support code for a given (device motherboard) board that conforms to a given operating system. It is commonly built with a boot loader that contains the minimal device support to load the operating system and device drivers for all the devices on the board. Some suppliers also provide a root file system, a tool chain for making programs to run on the embedded system (which would be part of the architecture support package), and configurations for the devices (while running). A Board Support Package (BSP) is the common name for all board hardware-specific code. It typically consists of the following: The boot loader, The OEM Adaptation Layer (OAL), Board-specific device drivers. The BSP creation process involves the following tasks:
- Developing a boot loader,
- Developing an OAL,
- Creating device drivers,
- Modifying run-time image configuration files.


IV. System Architecture

The heart of the system is a real-time kernel that uses preemptive scheduling to achieve multitasking on hardware platform. The previous sections dealt with µCOS_II porting to the application desired. This section deals with the implementation of both hardware and software.

In Micro C/OS-II maximum number of tasks is 64. In the figure shown above the application has four tasks. Depending on the required application the number of tasks may vary. Here temperature, humidity, gas and IR sensors are the four tasks. Temperature is considered the most critical even and has given the highest priority. Humidity, gas and IR sensor follow in order. The following is the code presented for implementing the four tasks using the µC/OS-II RTOS. The four tasks are initialized. We can allocate the stack for the the tasks as follows:

```
OS_STK TestTask1Stack[TaskStkLength];
void TestTask1(void *pdata);
OS_STK TestTask2Stack[TaskStkLength];
void TestTask2(void *pdata);
OS_STK TestTask3Stack[TaskStkLength];
void TestTask3(void *pdata);
OS_STK TestTask4Stack[TaskStkLength];
void TestTask4(void *pdata);
```

OSInit() is used to the µC/OS-II.

OSTaskCreate(TestTask1, (void *)0, &TestTask1Stack[TaskStkLength-1], 0);

The above function is used to create task.

OSStart() is used to perform multitasking.

```
AD1CR = 0x00200400 | (1 << 3);
//Choosen ADC1.3(P0.12)
```

OSTimeDlyHMSM() is used to provide the delay in hours, minutes, seconds and milliseconds.

The ARM runs the Real time operating system to collect information from the external world. Here RTOS is used to achieve real time data acquisitions. Micro C/OS-II kernel is ported in ARM powered microcontroller.

Keil IDE is used for implementation. Keil IDE is a windows operating system software program that runs on a PC to develop applications for ARM microcontroller and digital signal controller. It is also called Integrated Development Environment or IDE because it provides a single integrated environment to develop code for embedded microcontroller. The following shows the snapshots of the results using the µC/OS-II RTOS and ARM7 LPC2148 measuring the temperature, humidity and percentage of gas content and the number of obstacles in the path.

---

**Fig. 4:** The Architecture of Hardware and Software When Using µCOS_II

**Fig. 5:** Block Diagram

**Fig. 6:** LPC2148 With Sensors, 1. Gas Sensor, 2. Humidity Sensor, 3. Infrared Sensor, 4. Temperature Sensor
The humidity sensor used is SY-HS-220. It converts the relative humidity to output voltage. The gas sensor used is MQ5. It can detect town gases, hydrogen, 100ppm to 3,000ppm. They are used in gas leakage detecting equipments in family and industry and suitable for detecting of LPG, natural gas, town gas. LM35 temperature sensor is used to measure the temperature. The hardware module of IR sensor contains a TX-RX pair. The IR sensor acts as an object counter here. Zigbee is used to transfer information from the remote location to the user place. The ARM7 sensor module is placed at the remote location. It acts as the transmitter. The receiver module is at the user location. Zigbee module is incorporated in both the modules for transmitting and acquiring the information. At the user’s location the information will be viewed in the Matlab GUI. It is shown below.

![Matlab GUI](image)

**Fig. 7: Matlab GUI**

The COM port is selected. Then on pressing the capture button the measured values will be displayed in the respective places.

**V. Conclusion**

In this paper the development of a monitoring system with porting of μC/OS-II in ARM 7 is presented. It mainly focus on designing an embedded monitoring system using ARM 7 and μC/OS-II. The steps involved in porting the RTOS and final implementation details are provided. This paper provides an detailed overview for developing a embedded monitoring system using ARM and μC/OS-II.

**References**

[1] Liu Zhongyuan, Cui Lili, Ding Hong,“Design of Monitors Based on ARM7 and Micro C/OS-II”, College of Computer and Information, Shanghai Second Polytechnic University, Shanghai, China, IEEE 2010.


Venkateswara Rao received his M. Tech degree in VLSI design from Satyabhamma University, Chennai in 2008. He is currently working as an assistant professor in the Department of ECM in K.L University since 2009. His research interests include vlsi design and embedded systems.

Prasanna pursing her Master’s degree 2nd year in Embedded Systems from KL University. Her research interests include sensor networks and embedded design.