© 2017 IJSRST | Volume 3 | Issue 7 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Science and Technology

Porting Modbus Stack to FreeRTOS on LPC1768

P. Azaruddin[1], Arunakar. B[2]
[1]PG Scholar, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India
[2]Solution Architect, TCS, Hyderabad, Telangana, India

ABSTRACT

The key idea behind this paper is to implement Modbus RTU protocol on top of FreeRTOS using LPC1768 microcontroller and DragonBoard 410C. Modbus RTU protocol is serial communication protocol which is mainly for industrial applications like Building Management System. This protocol is implemented on top of FreeRTOS, which is a real-time kernel for small embedded systems. A small application is created by using temperature sensor whose output is sent to IoT gateway device by implementing Modbus Serial protocol. DragonBoard 410C is used as gateway device. The temperature data from gateway device is sent to the cloud using ThingsBoard IoT platform.

Keywords: Modbus Serial Protocol, FreeRTOS, LPC1768, DragonBoard 410C, IoT Gateway, ThingsBoard

I. INTRODUCTION

Embedded Systems plays a most significant role in the field of electronics. Most of the computing systems today utilize embedded systems. As the technology getting advanced embedded systems are programmed with real-time operating system which maximizes the efficiency of the system by proper handling of multiple tasking, reducing latencies and many other challenges. Many things like architecture design, timing analysis, multitasking and specifications are needed to keep in mind while designing as software for real-time embedded system.

FreeRTOS is basically an open source and real-time kernel which runs on the microcontroller to develop real-time embedded system applications. FreeRTOS can be efficiently used for complex embedded system applications because it provides time-related application programming [1].

Modbus protocol basically a serial communication protocol to transmit data between the devices like master/slave communication [6]. This paper discuss about how Modbus protocol is used to send the temperature data from LPC1768 to IoT gateway device. Here Dragon Board 410C is used as IoT gateway device. The microcontroller LPC1768 acts as slave device and Dragon Board 410C as master device. The data from IoT gateway device is sent to PC and from PC to the cloud using Thingsboard.io IoT platform which uses MQTT protocol [4].

The paper is structured as following: section II and section III discuss about basic details of FreeRTOS and Modbus Protocol. The details of IoT gateway and architecture of the system is discussed in section IV and section V including implementation. In these sections, the implementation of Modbus protocol and how data is sent to the cloud by using Dragon Board 410C as gateway device is discussed. The results and conclusion are given in section VI and VII.

II. FreeRTOS

FreeRTOS is a Real Time Kernel which is mostly applied for simple and also for complex embedded system applications. It was officially declared by FreeRTOS Engineers Limited that FreeRTOS supports more than 30 microcontroller architectures and can be built on 20 different compilers. By this porting capability, FreeRTOS has leading scope in real-time embedded applications. Hence FreeRTOS provides the core real-time scheduling functionality, inter-task communication, timing and synchronisation primitives only [1]. Any application in FreeRTOS is written in the form of tasks. If more applications are there then more tasks are created and all tasks are executed one by one by giving priority to each task. The main features of FreeRTOS include pre-emptive scheduling, task prioritization, implementation of queues, semaphores and mutexes. All the applications in FreeRTOS are written in C language. The core FreeRTOS code is contained in three source files with necessary header files as mentioned below:

- task.c
- queue.c
- list.c

The project files for FreeRTOS demo applications can be downloaded from FreeRTOS official website. The only header file which must be included to any FreeRTOS project is FreeRTOSConfig.h because it contains the configuration items of the demo project. Each demo project has its own FreeRTOSConfig.h header file [1]. The FreeRTOS code is managed by the heap memory. Whenever a new task, queue, semaphore or mute is created RTOS needs RAM. The RTOS heap memory dynamically allocates the RAM within the RTOS application program interface object creation functions. Due to the high advantage of FreeRTOS in complex and real-time embedded systems, it is mostly used for commercial applications and it is also freely available.

III. MODBUS PROTOCOL

Modbus is an open protocol developed by Modicon in 1979 which is mainly used by many manufacturers for industrial applications. It is basically a serial communication protocol to transmit data between the devices like master/slave communication. To keep it simple, it is a protocol that determines the process of how data is packaged and sent from slave device to master device [2]. The basic block diagram of Modbus serial communication is shown in below Fig.1:

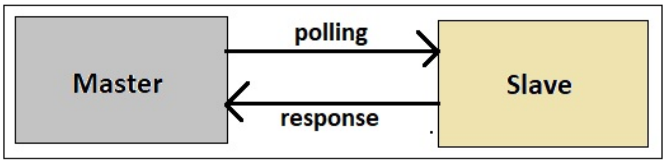

Figure 1 Basic Modbus Communication

Modbus can be communicated in three ways as mentioned below:

- Modbus ASCII
- Modbus RTU
- Modbus TCP/IP

Modbus ASCII, all the messages in this communication are in the form of ASCII characters and the messages can be readable while monitoring. It is the slowest protocol among above three protocols which is suitable for telephone modems.

Modbus RTU, data is sent in the form of binary and uses hexadecimal representation of data. As the messages are in binary coding and cannot be read while monitoring. Modbus RTU communication is done using RS-232, RS-485, and UART serial cables by mentioning slave address. It is mostly used by the manufactures in the industry.

Modbus TCP/IP protocol is like Modbus RTU but is built on top of TCP/IP and commutation is established using IP address rather than slave address. The Ethernet network which supports TCP/IP must support for Modbus/TCP protocol.

To establish communication with the slave device, the master should send the slave address, function code, data and error check in the Modbus RTU frame. Modbus message format contains Slave Address, Function Code, Data that needs to be transfer and Check bit which is called as CRC (Cyclic Redundancy Check). The following Fig.2 gives the detail message format of Modbus message.

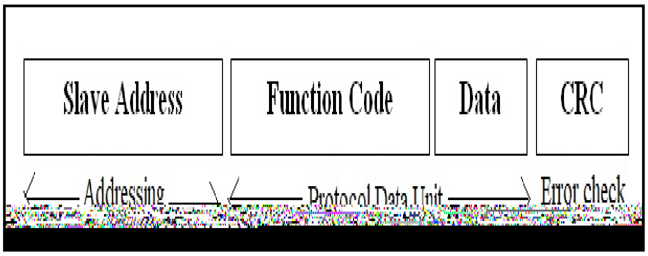

Figure 2 Modbus Message Format

Slave Address: Modus can have up to 247 slaves from 0 to 247 (decimal number). If only one device is using as a slave then '0' is given as slave address in the message format. To have successful transmission of messages between master and slave, slave address must be called in the Modbus initialization function.

Function Code: There are nearly 255 function codes which give commands to the slave device to read or write the data. Depending on the device necessary

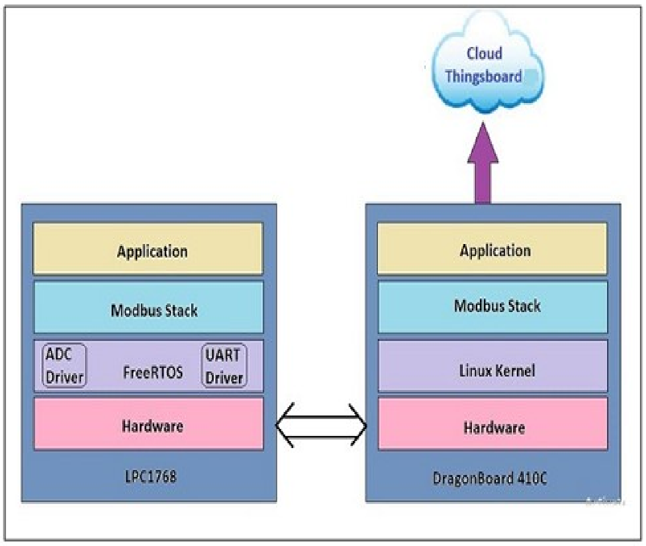

Figure 4 Architecture of the System

1) Hardware:

The hardware contains the peripherals like CPU, RAM and I/O. The LPC1768 microcontroller operates at 100MHz and contains the 512KB of flash memory. The communication between LPC1768 microcontroller and Dragon Board 410C is Modbus Serial Communication.

2) OS Layer:

This layer provides the communication between hardware layer which contains all the necessary peripherals and application layer. In which the driver codes like ADC and UART codes are written to communicate with the other devices. The operating system running in Dragon Board 410C is Android.

3) Application Layer:

On top of FreeRTOS, Modbus stack is written which send the data to the application layer. Then data is sent to the cloud by using open source IoT platform called ThingsBoard.

B. Implementation of Modbus Serial Protocol

A small application is done by implementing Modbus Serial Protocol to send the temperature data from LPC1768 to IoT gateway device and from gateway device to cloud. Dragon Board 410C is used an IoT gate device and the communication between LPC1768 and Dragon Board is through Modbus Serial UART port.

The complete block diagram of the application is shown in the following Fig.5:

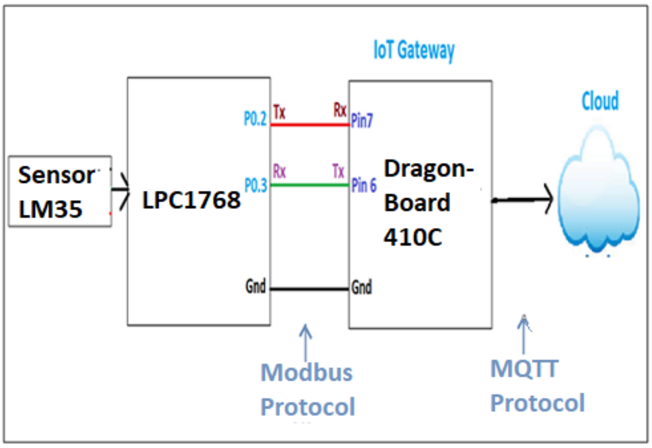

Figure 5 Block Diagram of the System

1) Integration of Temperature Sensor:

The first step is to integrate temperature sensor with LPC1768 microcontroller. The temperature sensor used is LM35 which senses surrounding temperature and gives its output in the form of analog voltage [7]. The output of the temperature sensor is given to ADC of LPC1768 to convert the data into digital and to process it to IoT gateway. A task is created by using FreeRTOS to read the temperature data from sensor. The input voltage applied to LPC1768 is 3.3Volts. Registers of ADC are configured properly to get the temperature data from the sensor.

2) Modbus Communication:

Modbus protocol is used to send the temperature data from LPC1768 to IoT gateway device. Modbus RTU is a serial communication which uses RS-232, RS485 or UART. Here UART serial communication is used to transfer temperature data. UART0 is used to communicate with IoT gateway device. To send the data from LPC1768 to IoT gateway, UART registers need to be configured. There are 6 UART registers in LPC1768: RBR, THR, FCR, LCR, DLL and DLM [5].

The communication is done through Modbus protocol, hence a function called eMBInit () is initialised in the code which slave address, port number and baud rate as its arguments. The slave address and port number are initialised as ‘0’, baud rate as 115200. To have proper UART communication between devices the baud rate of two devices should be same. In the initialisation of eMBInit (), another Modbus function is included to send data to Dragon Board 410 C. The function used to send data is eMBRTUSend () which has the same slave address as mentioned in the function eMBInit (). All these functions are included in the FreeRTOS task which is created initially to get the temperature data.

3) Sending Data to Cloud:

The temperature data which is obtained from IoT gateway device is now sent to the cloud using ThingsBoard IoT platform. ThingsBoard is an open source IoT platform which is used to collect, analyze and deliver device data to other systems. Data from IoT gateway device is sent to the cloud using ThingsBoard platform which uses MQTT protocol. Installation of MQTT libraries and java script files needed to send data to the cloud is done in PC (Personal Computer) which runs on Ubuntu 14.04 version. A device is created in ThingsBoard IoT platform by using tenant account.

The temperature data from DragonBoard 410C is pulled into PC (Personal Computer) by using ttyUSB. A file is created to read the data from ttyUSB. Now the data from PC to cloud is pushed by running mosquito.sh file in the terminal which contains the java script to publish the data. After running the shell script file, the connection status is visible in the Ubuntu terminal. If the connection is established then the temperature data is visible on the selected widget. There are many widgets available and any one widget can be selected to visualize temperature data.

VI.EXPERIMENTAL SETUP AND RESULT

The MCUXpresso is the IDE used to compile the FreeRTOS code which is written to read temperature data. The code is written in Embedded C. To flash the program into LPC1768 microcontroller Segger J-Link debugger is used. Segger J-Link supports many microcontrollers and downloads the code directly to RAM and flash memory. Setting breakpoints in external memory of Cortex-M systems is possible only with J-Link's Unlimited Flash Breakpoints technology. The hardware setup that is done to get the temperature data is shown in the below Fig.6:

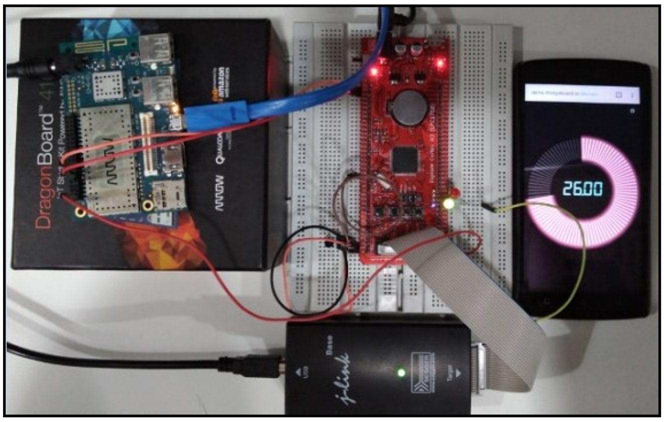

Figure 6 Experimental Setup

It contains LPC1768 microcontroller, Segger J-Link debugger, Dragon Board 410 C, LM-35 temperature sensor, and basic electronics kit like breadboard, connecting wires. In the above figure, it is observed that the temperature reading 26 degree Celsius is shown in one of the widgets of the ThingsBoard IoT platform.

VII. CONCLUSION

Modbus protocol stack has been successfully ported to FreeRTOS on LPC1768 microcontroller by utilising temperature sensor as an application part. By this implementation of Modbus protocol on top of FreeRTOS, it is easy to use in industrial applications to communicate and control the devices. As the world is transforming towards IoT, this Modbus implementation using FreeRTOS is an added advantage for industrial applications like Building Management Systems. The project can be extended further by adding more sensors like humidity sensor, ultrasonic sensors to the IoT gateway device by creating few more tasks in FreeRTOS. If the tasks are more in FreeRTOS then all those tasks need to be scheduled in time which is one of the challenging tasks to do.

VIII. REFERENCES

[1] FreeRTOS; http://freertos.org.
[2] Kelong Wang, Daogang peng, lei Song and Hao Zhang "Implementation of Modbus Communication Protocol based on ARM Cortex M0", IEEE Conference on System Science and Engineering (ICSSE), pp.69-73, July 2014.

[3] DragonBoard 410C Development Board: https://www.96boards.org/documentation/ConsumerEdition/DragonBoard-410C/
[4] ThingsBoard Open Source IoT platform: https://thingsboard.io/docs/getting-started-guides/heloworld/
[5] LPC1768 ARM Cortex-M3 based Microcontroller:: http://akizukidenshi.com/download/lpc1768.pdf
[6] Modbus Protocol: http://modbus.org/faq.php
[7] LM35 Temperature Sensor: http://www.ti.com/lit/ds/symlink/lm35.pdf
[8] IoT: https://en.wikipedia.org/wiki/Internet_of_things
[9] MCUXpresso Integrated Development Environment: https://www.nxp.com/docs/en/user-guide/MCUXpresso_IDE_User_Guide.pdf