Analysis Framework of Freemodbus

Zhang Wenxia[1,a] ， Wang Yan[2,b]， Li Shuhua[3, c]

Department of Electronic Information Engineering, Ordos College of Inner Mongolia University, Ordos, Inner Mongolia

[a]zhangwenxia100@163.com， [b]nmsnake@126.com ， [c]lsh8120@imu.edu.cn

Keywords: modbus protocol， freemodbus,framework，HAL

Abstract. Modbus is a popular network protocol in the industrial manufacturing environment. A modbus communication stack requires two layers:the modbus application protocol which defines the data model and functions and a Network layer. The FreeMODBUS provides an implementation of the modbus application protocol and supports RTU/ASCII transmission models. The paper introduces the analysis of FreeMODBUS's whole framework and the serial links implementation. According to the analysis, it is easily to realize the modbus protocol stack on the chip which FreeMODBUS hasn't realized and so quickly to finish the product design.

Introduction of modbus

Modbus is the first filed bus that used in the industrial filed around the world. In China, modbus has become the national standard GB/T19582-2008. modbus is the transport protocol on application layer of the seventh level of OSI model which provides client/server communications among devices that connected by varies bus and network.

Introduction of FreeMODBUS[1]

FreeMODBUS is a free implementation of the popular modbus protocol specially targeted for embedded systems. The FreeMODBUS current version provides an implementation of the modbus application protocol v1.1a and supports RTU/ASCII transmission models defined in the modbus over serial line specification 1.0. The implementation is based on the most recent standards and fully compliant with it. Receiving and transmitting of modbus RTU/ASCII frames is implemented as a state machine which is driven by callbacks from the hardware abstraction layer. This makes porting to new platforms easily. After completed received, the frame is passed to the modbus application layer where HAL(hardware abstraction layer) is inspected. Hooks are available in the application layer to add new modbus functions.

Analysis the framework of FreeMODBUS and the realization of serial links

Modbus serial link protocol is a master/slave protocol[2], which layered on the second level of the OSI model.(shown in Fig.1).

Layer	ISO/OSI MODEL	
7	Application	Modbus Protocol
6	Presentation	Null
5	Session	Null
4	Transport	Null
3	Network	Null
2	Data Link	Modbus Serial Link Protocol
1	Physical	EIA/TIA-485(or EIA/TIA-232)

Fig 1. modbus protocol layer

Published by Atlantis Press, Paris, France.

In physical layer, modbus serial link can use different physical interfaces(such as RS485,RS232) . The serial port drivers are different in various chips. So the physical layer must be abstracted to a HAL (Hardware Abstraction Layer)which intents to hide the hardware differences among various chips.

As a universal model, the realization of the modbus application layer and the protocol stack is independent of the chips. The protocol stack implementation diagram is shown in Fig.2.

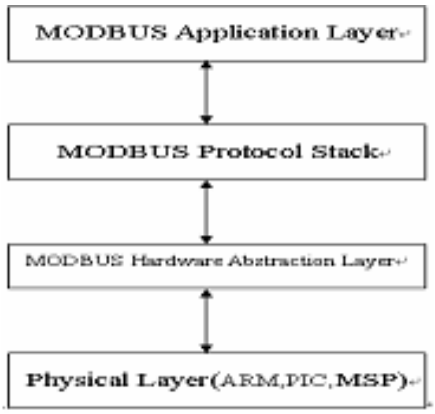

Fig 2. protocol stack implementation

Analysis of the application layer

The application layer interface file is mb.c, which is mainly realized the initialization and close of the protocol stack. The core part of the protocol stack is the event-driven message management which is responsible for processing corresponding events. In the high efficient network transmissions, such as middleware technology of the wireless sensor network , the technology based on event-driven is very common[3].The event-driven management function is eMBPoll, (shown Fig.3.)

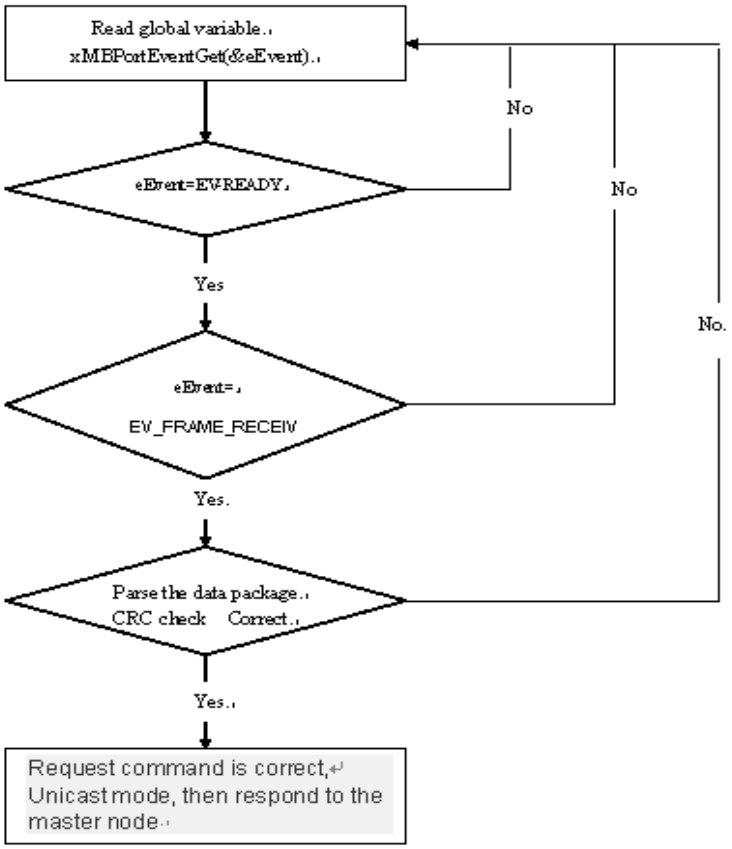

Fig 3. event driven management

Published by Atlantis Press, Paris, France.

Analyze the setting of the event (Msp430 RTU as an example

- EV_FRAME_RECEIVED

In the xMBRTUTimerT35Expired function, when eRcvState is at STATE_RX_RCV state, which indicates that /* A frame is received and t35 expired. Notify the listener that a new frame was received. * /, it is set to EV_FRAME_RECEIVED, then in the event handle processing poll, data analyze can be started.

- STATE_RX_RCV

In the xMBRTUReceiveFSM function, when eRcvState is modified from STATE_RX_IDLE state to STATE_RX_RCV state, it indicates the beginning of receiving data.

- STATE_RX_IDLE

In the xMBRTUTimerT35Expired function, eRcvState is set to STATE_RX_IDLE at last.

- pxMBFrameCBByteReceived

For the callback, the function pointer of the pxMBFrameCBByteReceived is set to xMBRTUReceiveFSM in the eMBInit. And the function pointer pxMBFrameCBByteReceived is called in the function prvvMBSerialRXIRQHandler (void) __interrupt [USART0RX_VECTOR]. So while data is receiving, read interruption is generated by the chip that lead to a series of actions and complete the data reading, parsing and response with the event management poll. This flow chart is shown in Fig.4.

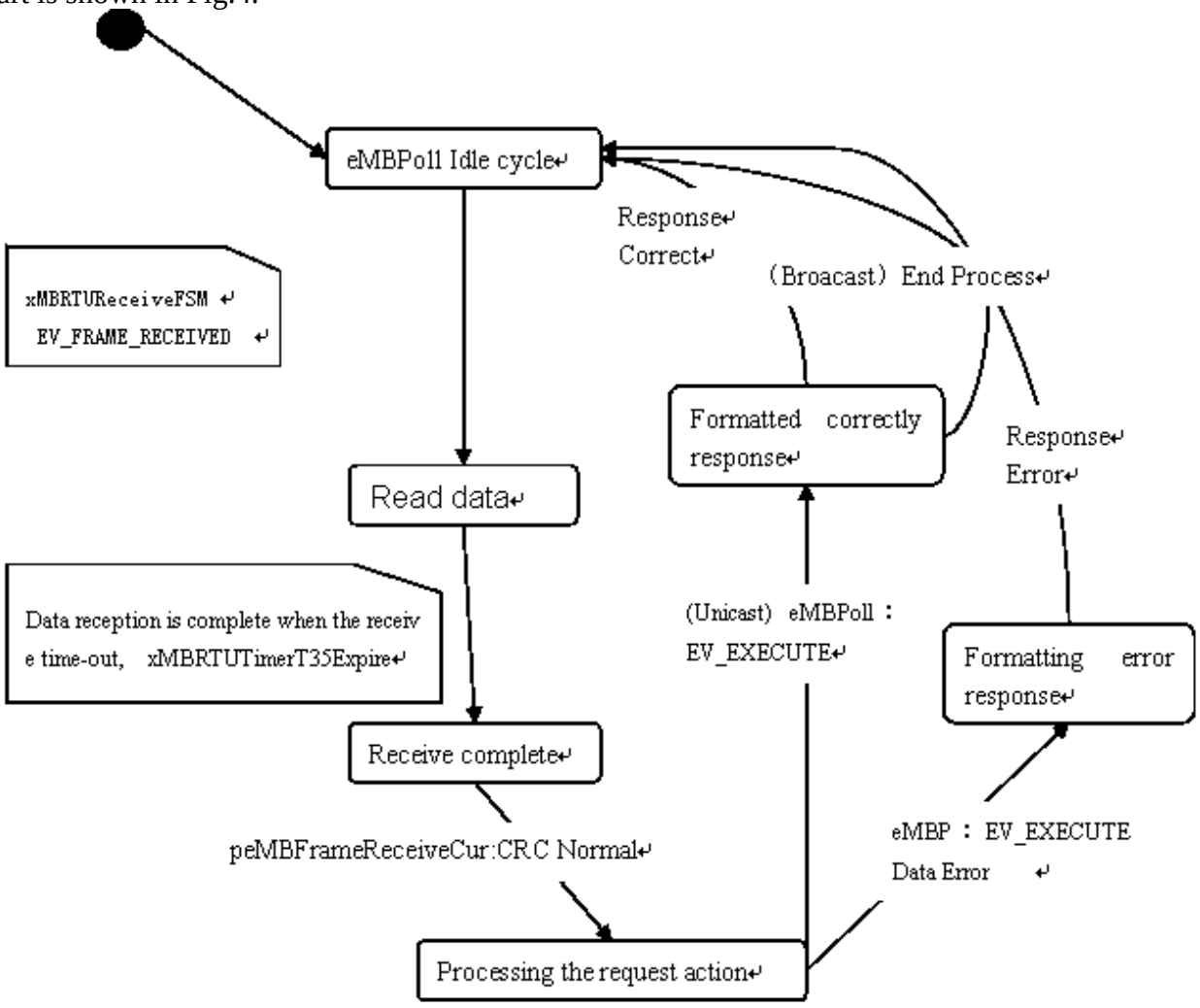

Fig 4. event management poll

HAL（Hardware Abstraction Layer) implementation

Realization of the HAL locates in the directory “port” of FreeMODBUS, the concrete implementation of the serial link is coded in port.h、 portevent.c、 portserial.c and porttimer.c.

port.h

The file defines data types and critical code section(ENTER_CRITICAL_SECTION / EXIT_CRITICAL_SECTION) which has to be simulate MCU, it has been realized in the windows system.

portevent.c

This file includes event management: get、post and init.

portserial.c

This file includes read interruption and the corresponding buffer read and write, this is the kernal part to hidden hardware differences among various chips’ read/write operation.

porttimer.c

This file includes the concrete realization of the timer’s initialization, enable and close on different MCU.

Summary

Through the analysis, it can concluded that FreeMODBUS has completely realized modbus protocol in which message response based on event can response for various interrupts, so it has good expandability. With the implementation of HAL, the hardware differences among various chips are hidden, so the entire protocol stack is independent of the hardware. The engineers need only to re-write the HAL layer so that modbus protocol can be realized on their chips. Our labs have completed the implementation of modbus protocol on the PIC30F6014 chip[4] [5]. The results of the software to test the protocol is shown as in Fig.5.

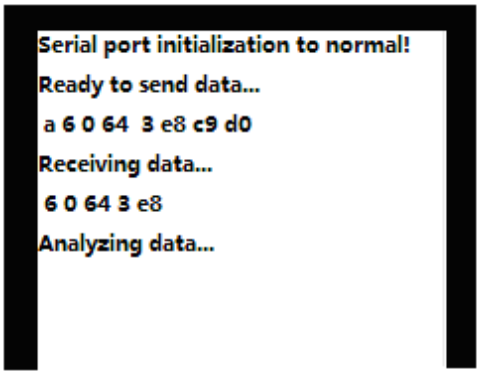

Fig 5. modbus realization on pic6014

References

[1] Http://FreeMODBUS.berlios.de/References.

[2] modebus protocol Chinese version, pp 0-126

[3] Wireless sensor network middleware technology, Ru-chuan, Science Press, 2011

[4] dsPIC30F Programmer’s Reference Manual，2003 Microchip Technology Inc. pp 5-84 5-102

[5] dsPIC30F Family Overview,2004 Microchip Technology Inc. pp 8-16

Published by Atlantis Press, Paris, France.