International Journal of Hybrid Information Technology
Vol.8, No.9 (2015), pp.25-32
http://dx.doi.org/10.14257/ijhit.2015.8.9.03

Design and Implementation of Embedded Serial Communication Based on Finite State Machine

Jian Feng and Yuanyuan Ding

College of Computer Science & Technology, Xi'an University of Science and Technology, Xi'an 710054, China
fengjian@xust.edu.cn

Abstract

In order to solve problems of poor real-time capability, poor reliability and low fault tolerance for data transmission in embedded serial communication, a serial communication protocol based on finite state machine (FSM) was designed and implemented, and was applied in communications of network management. The packet format and the communication control mechanism of the serial communication protocol are described, and complete framework for the application is given. Process of receiving and transmitting FSM models in serial communication are abstracted, combined with the interrupt service routine. In practical application, the serial communication program can meet the functional and performance requirements of serial communication between embedded devices and network management system (NMS).

Keywords: *FSM, Serial Communication, Protocol, Real-Time, NMS*

1. Introduction

Serial communication is a very important and widely used means for communication among embedded systems, such as MCU and DSP, and embedded systems and PC.

CPUs of embedded systems not only need to complete the work of main flow, but also need to deal with interrupts occurred at any time, and their processing power are lower than CPUs of PC, which makes the program design for embedded systems is very different from that of PC. The program design of serial communication in embedded system is seemingly simple, but if not carefully designed, the problems of poor real-time capability, poor reliability and low fault tolerance will appear.

Finite state machine (FSM) model is a viable solution for embedded software design and is used frequently [1]. In this paper, a software system based on FSM is designed for serial communication between embedded devices and network management system (NMS). It aims to ensure the accuracy of data transmission by setting a reasonable packet format, while to simplify the protocol implementation and improve the reliability of communication, and to make the communication process has strong fault tolerant property by introducing state machine approaches.

2. The Overall Design

2.1. Serial Communication Protocol

The smallest information unit in serial communication is data frame. A data frame typically is made up of start bits, data bits and end bits, also used to have parity bits for detecting transmission errors. Each data frame can have 5, 7, 8, or 9 data bits [2]. In

ISSN: 1738-9968 IJHIT

actual communication, the data transmission is performed frame by frame according to communication protocol.

The communication protocol is an agreement reached by the two sides needing to communicate; it gives the definitions of data format, synchronous mode, transfer procedure, correction, and control characters and so on. The two sides must abide the communication protocol in communication [3].

The packet format of serial communication protocol defined in designed software system is shown in Table 1.

Table 1. Packet Format of Serial Communication Protocol

Message Type	Bit7-4	Bit3-0	Byte
packet header	0xFF		0
packet header	0xFF		1
control word	S_NO	R_NO	2
control word	packet type	packet subtype	3
length	length/256		4
length	length%256		5
message	uncertain		6
			…
			n-3
check word	checksum of control word, length and message		n-2
packet tail	0xFF		n-1
packet tail	0xFE		n

Where:

- Packet header is used for synchronization, indicates start of receiving a new data packet;
- Control word contains the type of the packet, the number of current received packet (S_NO) and the number of which packet sender wishes to receive (R_NO);
- The length indicates length of message;
- Message is the actual data sent or received;
- Check word uses single-byte XOR to check control field, length field and message field;
- Packet tail indicates the end of data package.

The main communication control mechanisms in the protocol include:

- Scans serial port every 50ms. If the serial port is idle and there has current buffered data needed to be transmitted, then send the data packet;
- After sending data packets, the sender needs to wait for the response;
- Resends the packet if its response has not been received after 2000ms; if the continuous retransmission reaches to 10 times and without response, then serial communication is considered to be lost;

- If packet number of received packet is inconsistent with the desired packet number, carries on synchronization according to current packet number of received packet. If it happens more than 10 times that packet number of received packet is inconsistent with the desired packet number, sends a management packet to make the number of the transmitted packet and that of the receipted packet synchronization.

2.2. System Design

According to real-time needs and reliability needs for communication between embedded systems and NMS, the task of serial receiver in the embedded device is defined as a strong real-time task, achieved by interrupt routine; the task of serial transmitter is defined as a weak real-time task, achieved by ways of polling, and every 10ms to send a byte.

When designing receiving interrupt in the serial port, in order to avoid consuming a large number of machine cycles to deal with protocol data resulting in that processing time of interrupt routine is too long to make other real-time tasks on the embedded device be delayed, copies the data received in the serial cache directly into the default data buffer, instead of processes the data directly and waits for next data arrived. Later a task invoked by the main program outside the interrupt routine will process the data in the default data buffer according to the serial protocol. In order to receive and identify a complete packet, as well as tolerant faults, FSMs and message mechanism are used in communication program design. System control process is shown in Figure 1, and the key technologies are described in detail in Part 3.

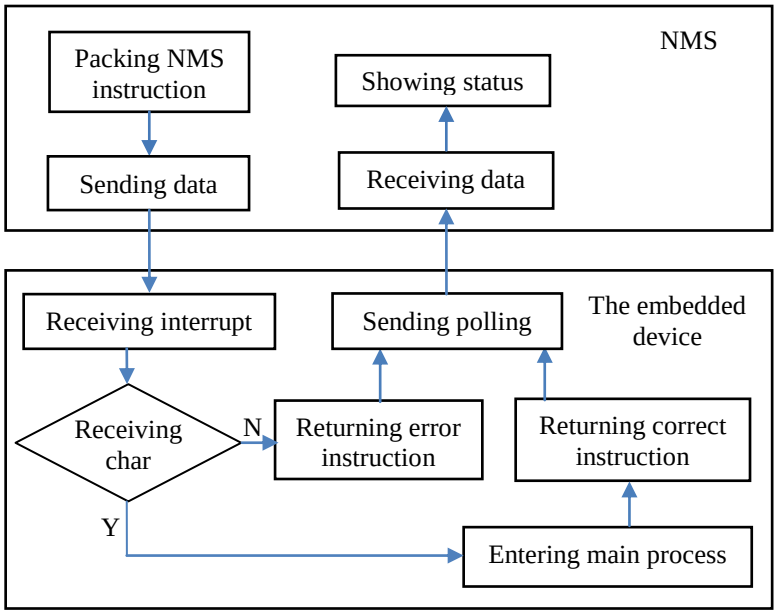


Figure 1. System Control Process between NMS and the Embedded Device

3. Key Technologies

3.1. Interrupt Routine Design

To make embedded systems more robust, one of the basic principles of program design should be taken is to make interrupt handler simple. In the serial interrupt

service routine, after one byte of data is received, copies it into the specified circulating public buffer. The data location in the buffer is pointed by read pointer and write pointer, and the buffer is set appropriate flags to indicate its contents. The main program reads the contents of the buffer in accordance with flags, and analyses and processes according to serial communication protocol.

Circulating public buffer is set to 8192 bytes. If communication rate is set to 9600bps and each byte occupies 10 bits, the buffer can store continuous serial data for about 8.53s.

3.2. Fsms of Protocol Processing

As the reaction system, a communication system is highly dependent on the states occurred in communication processing, so using FSM to model dynamic behavior in a communication system is a generally used programming method.

A FSM is used to describe systems with a finite number of states; at any given time it can operate on input to either make transitions from one state to another or cause an output or action to take place. A FSM can only be in one state at any moment. As a basic formal method, FSM has been widely applied in computer related field, such as definition of formal languages and description of network protocols. In serial communication, each processing procedure has several states transition driven by external events, so FSM can be used to describe these transition procedures.

A model of FSM can be defined as a five-tuple array $M = (Q, \sum, \delta, q_0, F)$[4], including:

- $Q=\{q_0, q_1,\ldots,q_n\}$ consists of a set of states (including the initial state). At a determined time, FSM will be in a determined state q_i. In serial communication protocol, there are 6 states, shown in Table 2.

Table 2. States of Serial Communication Protocol

State	Type	Definition
IDLE	Idle state	The initial/ terminal state
FIRSTFF	Receive state	Receive the first 0xff
SECONDFF	Receive state	Receive the second 0xff
THIRDFF	Receive state	Receive the third 0xff
WAITACK	Send state	Send data and wait for ack
NOACK	Send state	Timeout and no ack

- $\Sigma=\{\sigma_1,\sigma_2,\ldots,\sigma_m\}$ is a set of input events. At a determined time, FSM can only receive one determined event σ_j. In serial communication protocol, there are events including receive 0xff/0xfe, timer timeout, and so on, shown in Table 3.

Table 3. Input Events of Serial Communication Protocol

Event	Type	Definition
rff	Receiving FSM	Receive 0xff
rfe	Receiving FSM	Receive 0xfe
roth	Receiving FSM	Receive other char
em	Receiving	Packet error

sg	FSM	
en	Sending FSM	Seqno error for 10 times
o		
tto	Sending FSM	Heartbeat timer timeout
h		
ttoa	Sending FSM	Ack timer timeout
ttoat	Sending FSM	Ack timer timeout for 10 times
ack	Sending FSM	Ack packet needs to be sent
data	Sending FSM	Data packet needs to be sent
rack	Sending FSM	Receive ack packet
rmn	Sending FSM	Receive management packet
rhb	Send FSM	Receive heartbeat packet

- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function that maps input events and current states to a next state. The function takes the current state $q_i \in Q$ and an input event $\sigma_j \in \Sigma$ and introduces a set of actions and the next state $q' = \delta(q_i, \sigma_j) \in Q$. Table 4 shows the actions caused by an input event.

Table 4. Actions of Serial Communication Protocol

Action	Type	Definition
rnx	Receiving FSM	Receive next char
dem	Receiving FSM	Deal with data packet
smn	Receiving FSM	Send management packet
shb	Sending FSM	Send heartbeat packet
sack	Sending FSM	Send ack packet
sda	Sending FSM	Send data packet

- $q_0 \in Q$ is the initial state, computation begins in the start state with an input event. In serial communication, it is state IDLE.

- $F \subseteq Q$ is the terminal state, computation ends in this state. In serial communication, it is state IDLE too.

The state machine can be viewed as a function which maps an ordered sequence of input events into a corresponding sequence of (sets of) output events.

Figure 2 shows receiving event/state transitions in serial communication protocol.

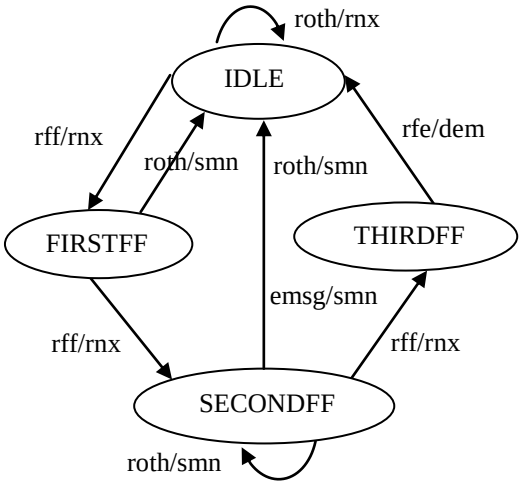


Figure 2. Receiving FSM

In receiving FSM, each stage of receiving process is defined to different state, the new received character or the results of the data processing are used as external trigger conditions to achieve the purpose of the state changes, and finally to complete receiving and checking of a data packet. In the FSM, after a valid data packet received, its processing needs to be completed with other task module on embedded devices, this is beyond the scope of this paper, so do not go into details here. This process may lead to generate some data packets, such as acknowledgement packet, new data packet, management packet, and so on. These created packets are sent into the transmit buffer and waiting to be transmitted, as while become events to impact sending FSM, as shown below.

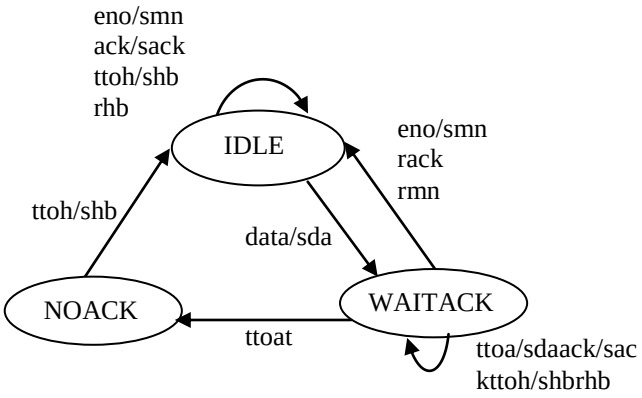


Figure 3. Sending FSM

The receiving FSM mainly concerns about receiving and checking of received packets, but the sending FSM focuses on dealing with events including sending data, monitoring timer expiration and checking sequence number of received packets to realize communication control of serial communication protocol.

3.3. Timers

In order to ensure the reliability of message transmission, serial communication process uses message retransmission mechanism. State machine designed above uses multiple timers, these timers and their meanings are summarized in Table 5.

Table 5. Timers

Timer	Meaning	Event by timeout
Timer A	ACK retransmission interval	ttoa
Timer AT	10 times of ACK retransmission interval	ttoat
Timer H	Heartbeat retransmission interval	ttoh

4. Testing

Serial program is tested by serial debugging software installed on the PC. Table 6 shows the basic parameters of the software. Connects the PC with the tool to the embedded device, and then sets the serial port parameters of the embedded device.

Table 7. Parameters of Serial Communication

Items	Description	Example
communication rate	Communication rate between PC and the embedded device	9600bps
data frame length	Maximum length of data frame received from serial port	200bytes
wait time	Expiration time span of serial port	50ms
parity bit	Number of parity bits	0bit

After extensive testing, the serial communication program can resolve protocol packets by 100%, it never shows any lost. The dispatching cycle of main program is 10ms, and the serial port interrupt needs 60μs by average, task which runs serial protocol needs 1.5ms by average and it does not affect the execution of other tasks.

In actual use, the embedded device which realized the serial program hosts a large number of sending and receiving job for network management, and can work properly and keep stable performance.

5. Conclusions

The serial communication protocol designed in the paper is semantically rich and signaling integrity. Its implementation uses two FSMs - receiving FSM and sending FSM. It has been applied to a NMS, running in good condition. As a result of using FSMs and message mechanism, the design of interrupt service routine is simple and quick, program structure is clear and easy to be maintained and transplanted, while the stability and reliability of the program is increased. The designed method has some practical value, and can be widely used in the field of industrial control and data communication.

Acknowledgements

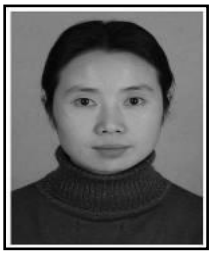
This work was supported in part by Shaanxi Provincial Natural Science Foundation Project (No. 2012JQ8030).

References

[1] ZH. H. Sheng, J. Y. Hong, L. Jie and G. J. Ning. Design of satellite management software based on FSM and event-driven, Computer Engineering. 35(21),280-282 **(2009)**.
[2] X. C. Sheng, S. T. Lu and SH. Cheng. The methods of formulating serial port communication protocol. Heilongjiang Science and Technology Information. 33, 83 **(2009)**.
[3] Ying, J. Bin. Design and implementation of serial port communication protocol based on state machine, Electronic Design Engineering. 20(7), 100-103 **(2012)**.

[4] ZH. W. Bo, ZH Hai, W. X. Ying, G. Mo and R. G. Chun. Study of the CFSM in an embedded device access server, Computer Engineering. 31(11), 103-104 **(2005)**.

Author



Jian Feng's date of birth: August, 1973. She received her doctoral degree in Computer Software and Theory from Northwest University, Xi'an, China, in 2008. And research interests on computer network and communication, network security, distributed computing.

She is currently an Associate Professor in College of Computer Science & Technology, Xi'an University of Science and Technology, Xi'an, China.

