Implementation of the Modbus/TCP Protocol on the STM32

Chen Tiejun, Yang Shaoming
College of Electric Engineering, Zhengzhou University, Zhengzhou, 450001, China
Email: sdytysm@163.com

Abstract

One of the key functions in industrial automatic control is communication between the various automation equipments. This paper discusses the requirements and methods about the selection of the TCP/IP protocol stack, it selects the uIP TCP/IP protocol stack, and it studies the transplanting feasibility about the stack on the STM32, it gives the specific transplant program. It provides a specific flow chart for Modbus/TCP communication protocol application layer, and it implements communication of the Modbus/TCP protocol on the STM32. This paper uses the Modscan32 as the upper computer simulation software and communicates with the STM32, through the test, the program is feasible, and can achieve the desired results.

Keywords: STM32, Modbus/TCP, uIP

1. Introduction

Modbus is a more intelligent field bus technology, which is similar to the other field bus standard has the characteristics of openness, intelligence, high environmental adaptability, convenient installation and maintenance, system integration initiative, accurate and reliable operation is very easy understanding of the advantages of a wide range of applications in the control field, and it formed industry standard. Modbus is a serial communications protocol published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Simple and robust, it has since become a de facto standard communication protocol, and it is now amongst the most commonly available means of connecting industrial electronic devices. The main reasons for the use of Modbus in the industrial environment are: it has been developed with industrial applications in mind, it is openly published and royalty-free, it is easy to deploy and maintain, and it moves raw bits or words without placing many restrictions on vendors.

Modbus protocol is implemented in two ways: First, Modbus / Serial Line protocol via serial, follow TIA / EIA 232-F and 485-A standard; Second, by means of Ethernet, Modbus / TCP protocol follows IETF RFC793 and RFC791 standard. For the second communication, this paper selects STM32 as main processor, to identify and select the uIP TCP / IP protocol stack, and it studies the transplanting feasibility of protocol stack in the STM32, gives the specific transplantation program of protocol stack in STM32, it provides specific flow chart for the Modbus / TCP communication protocol application layer, and it ultimately implements the Modbus / TCP protocol to communicate on the STM32.

Modbus protocol contains the Modbus/TCP for network communication [1]. When performing network communication, it need TCP / IP protocol stack, the control chip STM32 in the project doesn’t run the operating system which is integrated TCP / IP protocol stack, so it need to select a viable transplant TCP / IP protocol stack. But generally the TCP / IP protocol stack which is running on the PC is too large, it must simple the handling. uIP TCP / IP protocol stack is developed by the Swedish Institute of Computer Science, Adam Dunkels, it is designed in modular, simple transplantation, accounting for less memory. It is wise to select uIP TCP / IP protocol stack for saving storage space. The implementation of Modbus/TCP communication will become more simplistic because it contains Ethernet peripherals.

2. Modbus/TCP protocol stack Overview

Modbus/TCP protocol contains Modbus, TCP, IP, and physical layer and so on, the comparison with OSI seven layers protocol architecture are as follows:
Table 1. Compare Modbus / TCP communication protocol stack and the OSI architecture

<table>
<thead>
<tr>
<th>OSI architecture</th>
<th>Modbus protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Application layer</td>
<td>Modbus</td>
</tr>
<tr>
<td>6 The presentation layer</td>
<td>TCP</td>
</tr>
<tr>
<td>5 Session Layer</td>
<td>TCP</td>
</tr>
<tr>
<td>4 Transport layer</td>
<td>IP, ARP, RARP</td>
</tr>
<tr>
<td>3 Network layer</td>
<td>Ethernet, CSMA/CD, MAC</td>
</tr>
<tr>
<td>2 Data link layer</td>
<td>Ethernet Physical layer</td>
</tr>
<tr>
<td>1 Physical layer</td>
<td>Ethernet Physical layer</td>
</tr>
</tbody>
</table>

Table 1 show that Modbus/TCP communication protocol is divided into five layers. Because uIP TCP/IP protocol stack consists of two layers of 3 (network layer) and 4 (transport layer), it need transplant uIP TCP/IP protocol stack and implement Modbus/TCP application layer to communication.

3. uIP TCP / IP protocol stack transplantation

uIP TCP / IP protocol to communicate through some interface functions with the underlying system as well as the upper application, it is within the protocol for collection of external system is transparent, thereby enhancing the versatility and independence of the protocol, you can easily transplant to different systems and applications platform [2].

![Figure 1. uIP TCP/IP protocol stack interface](image)

Figure 1 shows the structural relationship between the uIP TCP/IP protocol stack and the underlying system and the upper application, and it gives three interface functions uIP_init (), uIP_input () and uIP_periodic () for communication with the underlying system, it provides uIP_appcall () interface function for communication with the application layer, so the main task of systems transplantation is to design these interface functions.

3.1 uIP TCP/IP protocol stack and network device driver interface

uIP TCP/IP protocol stack through uIP_input () function, and global variables uIP_buf, uIP_len to implement device driver interface. uIP_input () is the ground floor entrance function of uIP TCP/IP protocol, it is processing the received IP data packets; uIP_buf is used to store received or to be sent data packets of uIP TCP/IP protocol stack during system running; uIP_len is used to identify the received or to be transmitted to the data packet length of uIP TCP/IP protocol stack during system running, it can be used to determine whether there is the arrival of the data packet, or whether there is more data to send by determining the values of the variables. When the network device driver received data packet, the first packet is stored in uIP_buf, then call uIP_input () function for processing data packets; When data needs to be sent, also the first stored in uIP_buf in instructions package with uIP_len, when uIP_len is greater than 0, it indicates that data needs to be sent, the network device driver is called to send the data packet, otherwise, no data transmission.
3.2 The interface of uIP TCP/IP protocol stack and the system timer

System timer is used for timing polling system connection, polling to a TCP connection then calling uIP_periodic() function to check the status of the connection, if there is data need to be sent, and put the data into the global variable uIP_buf, and to modify uIP_len value. When uIP_periodic() returns, it checks uIP_len value, If the value of uIP_len is not 0, you need to the uIP_buf buffer packet sent to the network [3].

3.3 The interface of uIP TCP/IP protocol stack and application program

Because of the diversity of the application program, in order to make the transplant convenience, the protocol stack only provides an application program interface uIP_appcall(), the application is defined by the user. Finally, the user simply defined user program macros as UIP_APPCALL(), it can be connected with uIP TCP/IP protocol stack.

3.4 The transplantation on the STM32 of uIP TCP/IP protocol stack

Because uIP TCP/IP protocol stack for 8/16 bit microprocessor, STM32F107 is 32-bit microprocessor, in order to make the uIP TCP/IP protocol stack on a 32-bit microprocessor, so in the transplant, it add a uIP_add32() function. The function is as follows:

```c
void uip_add32(u8_t *op32, u16_t op16)
{
    uip_acc32[3] = op32[3] + (op16 & 0xff);
    uip_acc32[2] = op32[2] + (op16 >> 8);
    uip_acc32[1] = op32[1];
    uip_acc32[0] = op32[0];

    if(uip_acc32[2] < (op16 >> 8))
    {
        ++uip_acc32[1];
        if(uip_acc32[1] == 0)
        {
            ++uip_acc32[0];
        }
    }

    if(uip_acc32[3] < (op16 & 0xff))
    {
        ++uip_acc32[2];
        if(uip_acc32[2] == 0)
        {
            ++uip_acc32[1];
            if(uip_acc32[1] == 0)
            {
                ++uip_acc32[0];
            }
        }
    }
}
```
4. The implementation of the Modbus/TCP communication application layer

TCP protocol is a reliable connection-oriented protocol, so no additional provisions parity in Modbus/TCP protocol, and use a dedicated packet header (MBAP) to identify Modbus [4-8] application data unit, replace the serial link slave address on the commonly used, its structure is as follows:

![Figure 2. TCP / IP Modbus request/response][3]

MBAP packet header contains the following byte, as shown in Table 2.

<table>
<thead>
<tr>
<th>field</th>
<th>length</th>
<th>description</th>
<th>client</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction identifier</td>
<td>2 Byte</td>
<td>Modbus request/response transaction identification</td>
<td>Set by client</td>
<td>The server re-copied from the received request</td>
</tr>
<tr>
<td>protocol identifier</td>
<td>2 Byte</td>
<td>80=Modbus protocol</td>
<td>Set by client</td>
<td>The server re-copied from the received request</td>
</tr>
<tr>
<td>length</td>
<td>2 Byte</td>
<td>the number of bytes Subsequently</td>
<td>Set by client (request)</td>
<td>Set by server (response)</td>
</tr>
<tr>
<td>Unit identifier</td>
<td>1 Byte</td>
<td>Walk through link or other remote connection bus from the station identification</td>
<td>Set by client</td>
<td>The server re-copied from the received request</td>
</tr>
</tbody>
</table>

As child nodes in the communication, when receiving the data need to check whether the data is found in Table 2 and the structure shown in Figure 2, if mistaken, then skip or returns an exception code information. Specific flowchart as shown below:
When the system in communication, it uses detection function which is provided by uIP TCP/IP protocol stack to detect command byte sent by host, if detected, it will process data as shown in Figure 3, then it uses application program interface function which is provided by protocol stack and named uip_send() sent the processed result.

5. System experiments

This system uses STM32F107RC as a control sub-node, uses the TCP/IP protocol stack transplant method to transplant, according to above method to design the application layer. To let Modscan32 simulation software as the master node and control sub-node communication, and use function code 03 read multiple registers operating, control chip sampling data is uploaded to the host computer for processing.

In experiment, the control sub-node is set to the node 1, and the operation of reading a plurality of registers, the register is set the starting position is 0001, the number of read is 10, the experiment when the first stored in the register to be read 0-9 data. The experimental results shown below:
Host computer display map

Gray part in above picture is the request bytes of read register, the byte is analyzed as follows:

<table>
<thead>
<tr>
<th>01</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>01</th>
<th>03</th>
<th>00</th>
<th>00</th>
<th>0a</th>
</tr>
</thead>
</table>
A   | B   | C   | D   | E   | F   | G   |

Note:
A: transaction identifier
B: protocol identifier
C: length
D: unit identifier
E: function code
F: register start address (in the software, address is 0001 when the starting addresses of the register to 0, and so on)
G: The number of registers

Black part in above picture is the response bytes of read register; the byte is analyzed as follows:

<table>
<thead>
<tr>
<th>01</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>17</th>
<th>03</th>
<th>14</th>
<th>00</th>
<th>00</th>
<th>01</th>
<th>00</th>
<th>09</th>
</tr>
</thead>
</table>
A   | B   | C   | D   | E   | F   | G   | H   | I   |

Note:
A to E meanings of the above
F: the number of bytes when read register
G: the values of the first register
H: the values of the second register
I: tenth register values

By experimental analysis, it showed: The system runs effective; it can read required registers in accordance with the design requirements.

6. Conclusions

uIP TCP/IP protocol stack is designed in modular, easy to implement the transplant on STM32 control chip, on this basis, Modbus/TCP will become simplistic. The program for point-to-point communication, point-to-multipoint communication has certain reference value.
7. References


