International Core Journal of Engineering Volume 6 Issue 9, 2020
ISSN: 2414-1895 DOI: 10.6919/ICJE.202009_6(9).0025

Design and Development of Real-time Ethernet Communication Driver Software for Trains

Zhang Xiongxiong[1, a], Zhang Chunguang[1, b], Wu Qian[1, c]

[1]School of Dalian, JiaoTong University, Dalian 116021, China.

[a]2641729038@qq.com, [b]1263761813@qq.com, [c]286657945@qq.com

Abstract

With the development of train network control systems towards high speed, intelligence, and digitization, the application of real-time Ethernet for trains to high-speed EMU network technology is becoming increasingly important. Based on the theoretical basis of the train real-time data protocol (TRDP), this paper studies the TRDP communication driver. By designing and configuring the TRDP protocol stack, starting and stopping TRDP communication, and sending and receiving TRDP messages, the process data communication driver is packaged. The use of Swing components, action monitoring processing mechanism, multi-threading technology to achieve driver call, load TRDP configuration database, publish and subscribe message information data real-time display. Finally, use a PC and wireshark packet capture software to communicate and send and receive data and other functions to verify and analyze the captured TRDP messages.

Keywords

TRDP, Process data, Java, Swing, Wireshark.

1. Introduction

At this stage, MVB, LonWorks, ARCNET, CAN bus and other train communication network technologies have been maturely applied to existing high-speed trains, but as the equipment in the train tends to be diversified and intelligent, the service information for passengers tends to be comprehensive The traditional bus technology can no longer meet the increasing transmission bandwidth requirements of the train communication network. Relatively speaking, the mature application of industrial Ethernet and other technologies in their respective fields has led to the application of Ethernet technology to high-speed trains. The vehicle-level network ECN and the train-level network ETB will become the next-generation TCN. This article aims to study a TRDP communication driver software. The Ethernet card can carry TRDP communication with standard Ethernet trains and send and receive TRDP messages in real time.

2. Train Real-time Data Protocol

2.1 Overview

Train Real-Time Data Protocol (TRDP), which shall be used for the exchange of TCN process data and TCN message data over ETB. This layer may optionally be extended by a safety layer which provides safe end-to-end data transmission of safety critical process data and message data between any two end devices in the network, as shown in Figure 1.

The TRDP layer is above the data link layer, network layer and transport layer. At the data link layer TRDP uses ETB for in-train data communication. At the network layer TRDP completes the data exchange between the functions of different train groups based on IP, and the IP address allocation may be changed after the train is initially running. In the transport layer TRDP uses UDP and TCP transport layer protocol services for data communication. TRDP process data should be sent using UDP. The packet length of the TRDP process should be limited to one Ethernet frame length. TRDP message data can be sent using UDP or TCP.

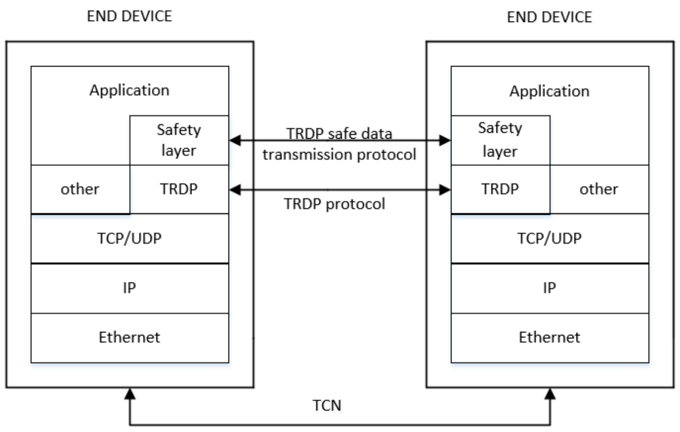


Fig. 1 Overview of the protocol stack

2.2 Process Data

The TRDP process data protocol defines the exchange of PD-PDUs for the transfer of process data. PD-PDUs shall be cyclically transmitted or transmitted on request between a publisher and one or many subscribers using a connectionless and unconfirmed TRDP service. The communication partners in a process data transfer can have different roles:

(1) publisher: the source of process data;

(2) subscriber: the sink of process data;

(3) requester: requesting the publisher(s) to publish its process data.

Process data exchange shall support the following three push communication modes:

(1) point to point, cyclic without acknowledge, source knows the sink;

(2) point to multipoint, cyclic without acknowledge, source knows the sink;

(3) point to multipoint, cyclic without acknowledge, source does not know the sink.

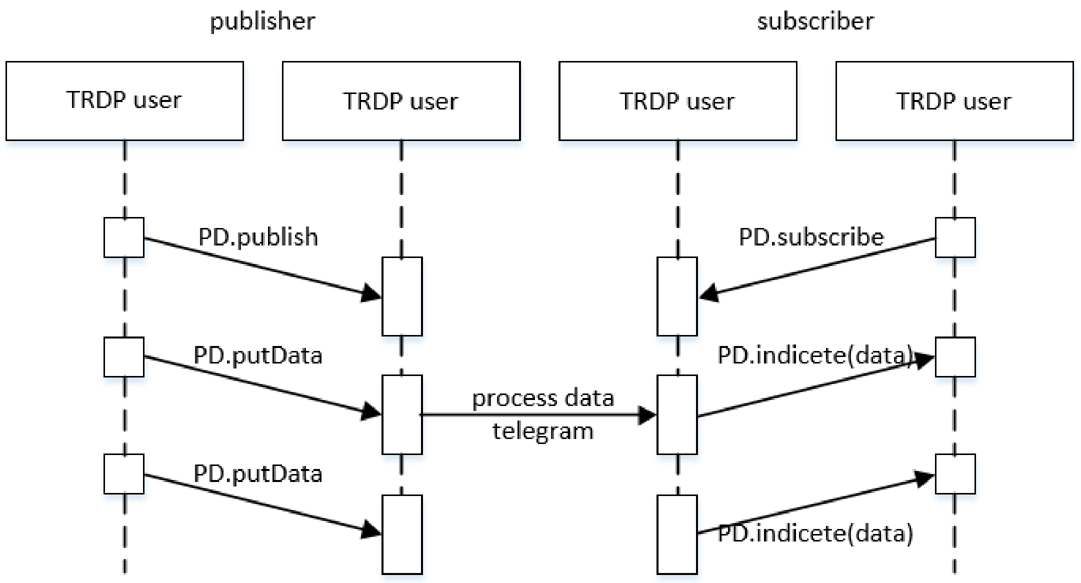


Fig.2 Interaction sequence chart for PD push pattern

A publisher or subscriber shall use an IP unicast address for addressing a known subscriber or publisher. A publisher or subscriber shall use an IP multicast address for addressing groups of known subscribers or publishers. A publisher or subscriber shall use an IP multicast address for addressing unknown subscribers or publishers.

The sequence chart in Figure 2 defines the allowed sequence of the service primitives for the PD push pattern.

Process data are prepared cyclically by the publisher and shall be given to the TRDP layer calling the PD.putData primitive. The publisher TRDP layer shall send the data in the configured cycle to the configured address. The publisher TRDP layer shall send out the data only after checking the locally stored topography counters submitted with the publish against the actual topography counters. Any subscriber can subscribe for the process telegram using ComID, destination IP address and source IP address of the process telegram as possible filter criteria. Timeout supervision at subscriber TRDP layer shall be started after subscription. Timeout supervision at subscriber TRDP layer shall be restarted after receiving the related PD PDU. The subscriber TRDP layer shall check the topography counters of the received telegram against the actual topography counters and against the topography counters submitted with thesubscription.

3. Design and implementation of TRDP driver software

3.1 Driven Design

JNI (JavaNativeInterface) is a Java native program interface, which realizes the interaction between Java and other languages (such as C/C++). Java code can operate applications and libraries written in C/C++ language, and local applications and libraries can also operate Java objects through JNI. In order to solve the problem of low running efficiency of Java, tasks with high real-time requirements can be implemented in C/C++ language, and JNI is used to realize communication between Java and C/C++.

In order to realize the modular design of the driver, the three-layer architecture of Java interface calling layer, local interface calling layer and local implementation layer is adopted.

In the Java interface call layer, the interface for obtaining the version of the dynamic link library, the number of published and subscribed messages, the comid, status, data set size, data set type and data of all messages, and starting and stopping communication are defined. function. On the one hand, these interfaces are provided to the application layer code call, which is used to drive the software to establish communication connections, send and receive TRDP message data, and other functions. On the other hand, these interfaces generate local call interfaces through JNI technology to implement Java call interfaces and local call interfaces. One-to-one correspondence.

In the local interface call layer, JNI technology automatically generates a local code (C/C++) interface corresponding to the Java interface, and implements the call with the local implementation code. In order to maintain the relative independence of the module, the local call layer does not contain specific communication driver code. Its main function is to achieve proper extraction and format conversion of the data passed from JNI, and pass the processed data to the local implementation layer. function.

In the local implementation layer, the TRDP protocol stack is initialized using the well-defined API at the bottom, starting and stopping TRDP communication, processing TRDP message data and other functions, and encapsulating the local call interface.

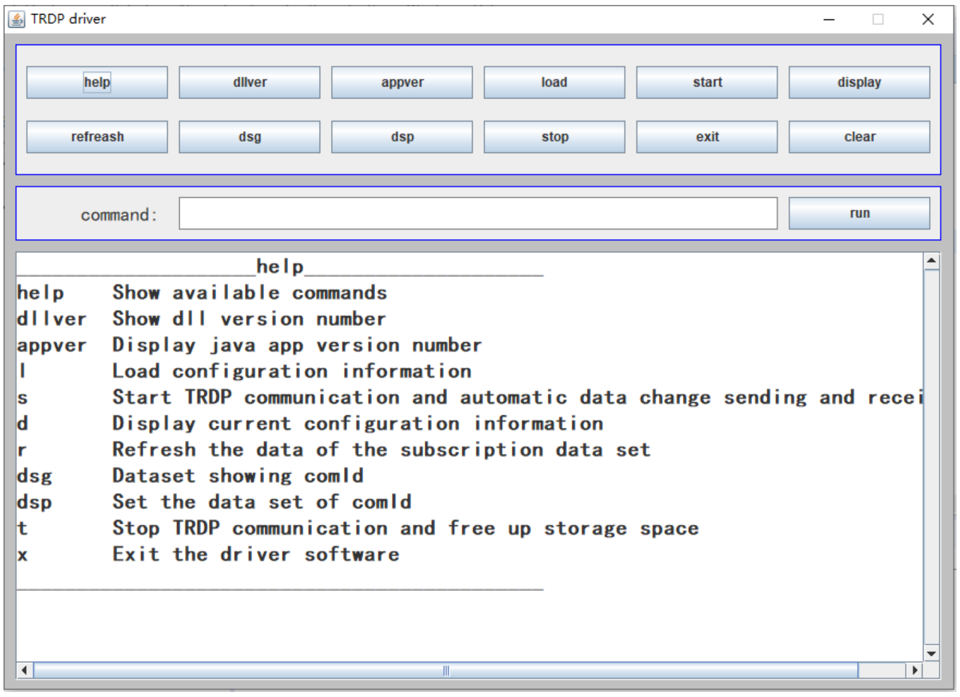
3.2 View Design

3.2.1 Java Swing technology

Java Swing technology is mainly used for graphical user interface development. It is improved on the basis of AWT (Abstract Window Toolkit) and provides rich components and functions. The developed window program has good cross-platformity and can run on the Windows platform. Can also run on the Linux platform, there will be no problems with different interface display. Swing components can be divided into top-level container, middle container and basic components according to different functions. The top-level container is a window-like component, inherited from java.awt.Window, and can be displayed independently. A graphical interface requires at least a window. The intermediate container acts as a carrier of basic components, inherited from javax.swing.Jcomponent, and cannot be displayed independently. Several basic components can be added to the middle container to group and manage the components in the container. The topmost middle container must be supported in the top container. Common containers include Jframe, Jwindow, Jdialog, JPanel, etc. Basic components are components that directly implement human-computer interaction, inherited from javax.swing.Jcomponent. Common basic components include Jbutton, JcheckBox, JtextField, JLabel, etc. To add various components to the panel container, you need to specify a layout manager for the panel container, and clarify the arrangement and layout of the components in the container.

3.2.2 Software layout and functions

Use the related components of Java Swing technology to lay out the software as shown in Figure 3. The interface is divided into three parts: button, command execution, and information display. The button part is a JPanel object set to a 2 row 6 column GridLayout layout, and 12 JButton objects are placed on it. The command execution part is a JPanel object set to a FlowLayout layout. Place three objects: JLabel, JtextField, and Jbutton. The information display part is to place a JTextArea object on a JScrollPane object. These three parts are placed in the NORTH, CENTER, and SOUTH areas of the JFrame object with the default layout of BorderLayout.



```
______________________help______________________
help     Show available commands
dllver   Show dll version number
appver   Display java app version number
l        Load configuration information
s        Start TRDP communication and automatic data change sending and recei
d        Display current configuration information
r        Refresh the data of the subscription data set
dsg      Dataset showing comId
dsp      Set the data set of comId
t        Stop TRDP communication and free up storage space
x        Exit the driver software
________________________________________________
```

Figure 3 Driver software interface

The driver software should be able to achieve the following functions:

(1) Display the available commands and detailed usage of the driver software;

(2) Display the driver library and software version number;

(3) Load the TRDP configuration database and initialize the TRDP protocol stack;

(4) Display all current configuration information;

(5) Start TRDP communication and allocate storage space;

(6) Send release message data, display all release messages and the specified comid release message;

(7) Receive subscription message data, display all subscription messages and subscription messages with specified comid;

(8) Stop TRDP communication and free up storage space;

(9) Clear all display information;

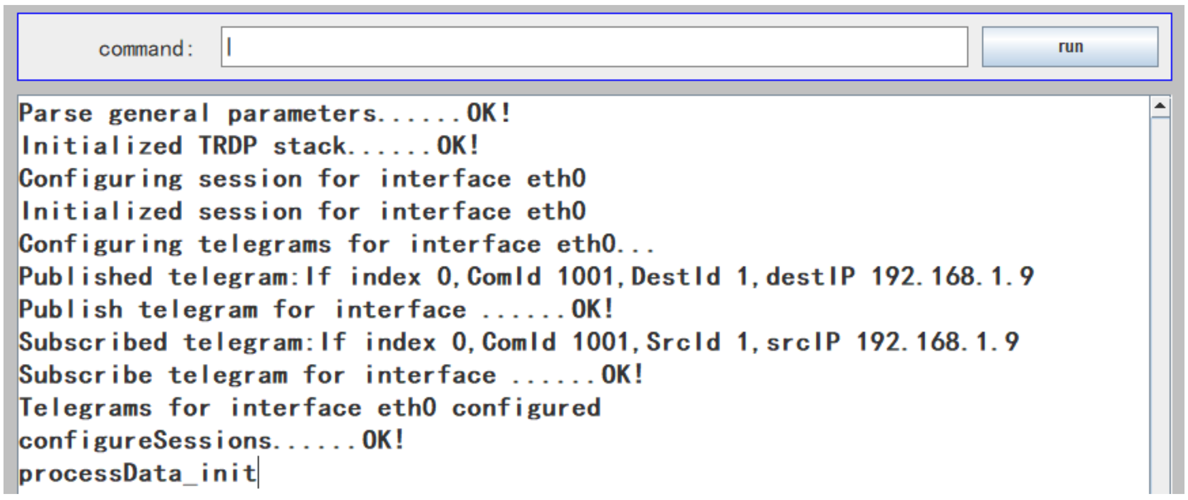
(10) Exit the driver software;

3.3 Function realization

Create a Java project TrdpDriver, create a class TrdpDriver.java in the project that inherits from JFrame and contains the main method. Define a no-argument constructor within the class, and add code to set the title, initial position, adaptive window size, window size cannot be changed, window operation is visible, and window closing method is added to the form. Create an object of the TrdpDriver class in the main method to enable the program to run normally.

In the Swing event model, three separate objects, event source, event and listener, handle the event. The event source triggers an event, which is received by one or more listeners, which are responsible for processing the event. The realization of the driver software function is mainly through the user input commands, click the execute button, and display the output after logical judgment. In order to achieve quick operation of some functions, the action event is triggered by clicking the corresponding button on the interface, and then the action event listener added to it is responsible for processing, and the action event listener needs to be added to each button in the program.

After starting the software, when you enter an invalid command or an empty command in the command box, there is an error prompt and the correct available command is displayed. Enter dllver and appver to get the version number of the dynamic link library and driver software. Enter l to parse the parameters by loading the configuration information database, configure the TRDP message, read the data set, and initialize the TRDP session. After inputting s, TRDP communication is started by opening up multiple threads. As shown in Figure 4, for TRDP publish or subscribe messages, the comid, data set id, destination ip, source ip, and ip address can be initialized with unicast or multicast addresses. Enter d to display all the configuration information of the current message, such as comid, data set id, data set size, data set name, data type and other parameters. Enter d 1 to start TRDP to send and receive data. Enter dsp to display the data of all published messages, enter dsp comid to display the data of the specified comid published message, and enter dsp comid zipian weipian data to send data to a certain word of the published message. Enter dsg to display the data of all subscription messages, enter dsg comid to display the data of the specified comid subscription message, and enter refresh to refresh the received change data. Enter stop to stop TRDP communication, enter exit to exit the driver software, and enter clear to clear all information on the display interface.



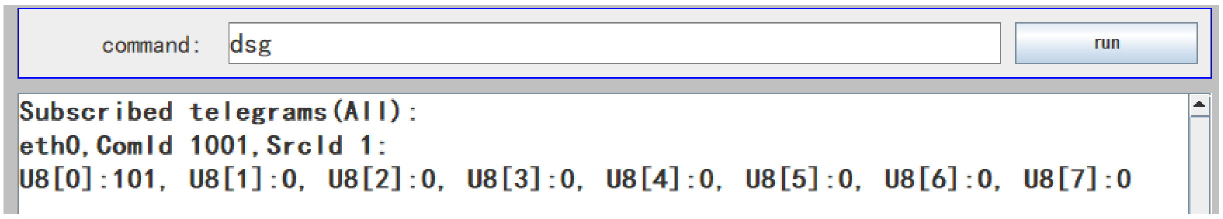
```
Parse general parameters......OK!
Initialized TRDP stack......OK!
Configuring session for interface eth0
Initialized session for interface eth0
Configuring telegrams for interface eth0...
Published telegram:If index 0,ComId 1001,DestId 1,destIP 192.168.1.9
Publish telegram for interface ......OK!
Subscribed telegram:If index 0,ComId 1001,SrcId 1,srcIP 192.168.1.9
Subscribe telegram for interface ......OK!
Telegrams for interface eth0 configured
configureSessions......OK!
processData_init
```

Figure 4 Initial configuration display

4. Software functional test

Through two PCs and a router, the communication function test is performed. The two PCs are connected to the router and automatically obtain the assigned IP addresses of 192.168.1.8 and 192.168.1.9, respectively, and set the comid and data set id to 1001 , The destination and source ip of the host PC are configured to 192.168.1.8, the destination and source ip of the sink PC are configured to 192.168.1.9, and the data set size is 1. Before running the driver software, use the DOS command PING to ensure normal communication between the two PCs.

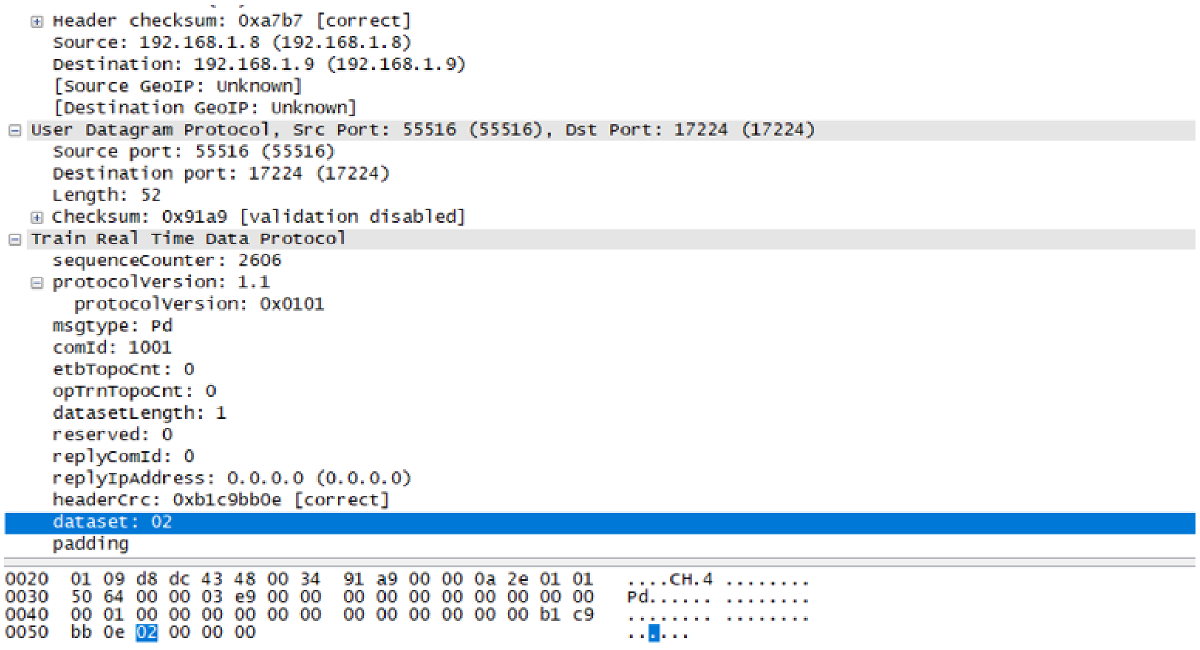
Click the dsg button in the main PC driver software to receive the data sent by the sink PC. As shown in Figure 5, you can see that the first data is 101, which verifies that the received data was successful. Enter the dsp 1001 0 0 66 command in the command box to send data 66 to the sink PC. Similarly, you can see that the data becomes 66 in the display area, and verify that the data is successfully sent.



```
Subscribed telegrams(All):
eth0,ComId 1001,SrcId 1:
U8[0]:101, U8[1]:0, U8[2]:0, U8[3]:0, U8[4]:0, U8[5]:0, U8[6]:0, U8[7]:0
```

Figure 5 Subscription message data

Use wireshark packet capture software to perform packet capture test and analyze the subscription message. As shown in Figure 6, you can see that the source IP of the subscription message is 192.168.1.8, the destination IP is 192.168.1.9, the comid is 1001, and the data set length is 1. The received data is 02.



```
  ⊞ Header checksum: 0xa7b7 [correct]
    Source: 192.168.1.8 (192.168.1.8)
    Destination: 192.168.1.9 (192.168.1.9)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊟ User Datagram Protocol, Src Port: 55516 (55516), Dst Port: 17224 (17224)
    Source port: 55516 (55516)
    Destination port: 17224 (17224)
    Length: 52
  ⊞ Checksum: 0x91a9 [validation disabled]
⊟ Train Real Time Data Protocol
    sequenceCounter: 2606
  ⊟ protocolVersion: 1.1
      protocolVersion: 0x0101
    msgtype: Pd
    comId: 1001
    etbTopoCnt: 0
    opTrnTopoCnt: 0
    datasetLength: 1
    reserved: 0
    replyComId: 0
    replyIpAddress: 0.0.0.0 (0.0.0.0)
    headerCrc: 0xb1c9bb0e [correct]
    dataset: 02
    padding

0020  01 09 d8 dc 43 48 00 34  91 a9 00 00 0a 2e 01 01   ....CH.4 ........
0030  50 64 00 00 03 e9 00 00  00 00 00 00 00 00 00 00   Pd...... ........
0040  00 01 00 00 00 00 00 00  00 00 00 00 00 00 b1 c9   ........ ........
0050  bb 0e 02 00 00 00                                   ......
```

Figure 6 Subscription packet capture analysis

5. Conclusion

This paper designs and implements the driver software based on Java Swing, multi-threading, JNI package dynamic link library and other technologies to solve the problem of the lack of TRDP communication driver in the train Ethernet card, and realizes the functions of real-time communication and data transmission and reception. In the future research, we will continue to improve the function of this driver software so that it can be easily applied to actual equipment.

References

[1] IEC 61375-2-3: 2015-07, Electronic railway equipment - Train Communication network (TCN) – Part 2-3: TCN communication profiles [S].

[2] Zhai Yameng, Li Chao, Tian Li: Design of Train Process Data Communication Resaersh Based on Real-time Ethernet [J]. Industrial Control Computer, Vol. 32 (2017) No. 2, p. 8-10.

[3] Li Tingxu, Li Changxian: Design of Train Real-time Ethernet Communication and Monitoring Software [J]. Industrial Control Computer, Vol. 30 (2019) No. 4, p. 77-79.

[4] Luo Yinqi, Liu Liyin: Design and Implementation of Java Serial Communication System Based on JNI Taking Windows Platform as an Example [J]. Computer Knowledge and Technology, Vol. 13 (2017) No. 34, p. 51-56.

[5] Tang Jianqiang, Li Miaozai: Design and Implementation of Calculator Based on Java Swing [J]. Computer and Telecom, Vol. 21 (2016) No. 10, p. 61-63.

[6] Yang Yang, Tang Guangheng, Hu Mingliang, Xiao Junchun: Design and Implementation of Student Status Management System Based on Java Swing Technology [J]. Fujian Computer, Vol. 65 (2018) No. 5, p. 127-128.