LR5-LAN Socket Communication

Sample Program

(Windows Java)

content

1. Overview

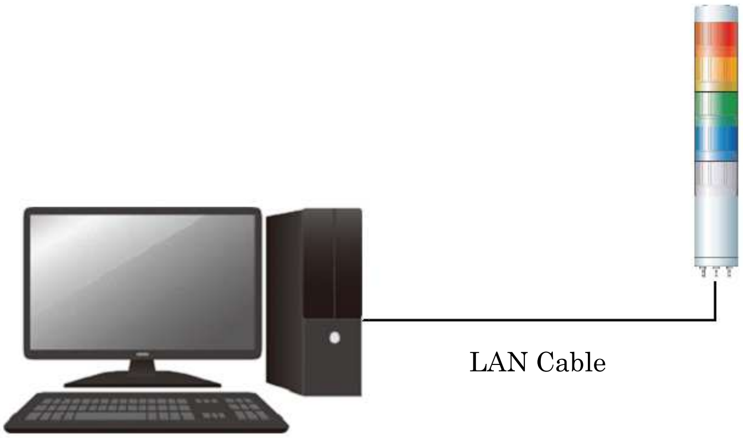
This is an outline of sample programming to control LR5-LAN via socket communication.

The programs are intended to control the unit using JAVA control without using the DLLs provided by PATLITE.

1.1. System Overview

The system configuration diagram of this program is as follows.。

The sample program controls one LA6 POE by socket communication.。



2. Development Environment

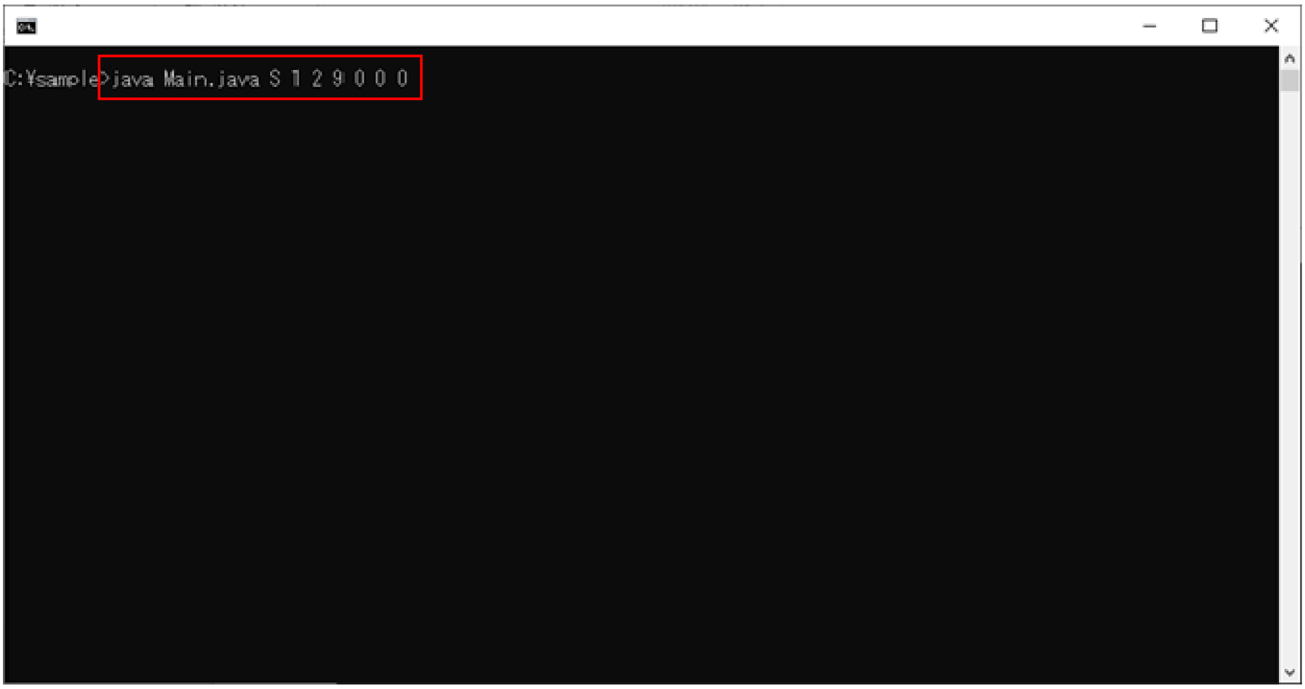
The development environment of the sample program is shown below.。

Development Environment		Remarks
Development OS	Windows11 64bit	
Development Language	Java (Adopt Open JDK)	11 Subsequent
Development tool	Eclipse	2021-06

3. Application Overview

3.1. Command Operation

Open Command Prompt, move to where 、Main.java is located and specify the command line arguments to execute commands for each operation.。



```
C:¥sample>java Main.java S 1 2 9 0 0 0
```

3.1.1. Command list

Command name	Contents
Operation control command	Controls LED unit pattern for each tier and alarm.
Clear Command	Turn off LED unit and stop alarm.
Status Acquisition Command	Used to acquire status of signal lines and the status of the led unit and alarm..

3.1.2. Operation control command

Execute command with the following command line arguments

No.	Command Line Argument	Value
1	Command ID	S
2	LED Unit Red	Off：0 On：1 Flashing(slow)：2 Flashing(medium)：3 Flashing(fast)：4 Single flash：5 Double flash：6 Triple flash：7 No change：9
3	LED Unit Amber	
4	LED Unit Green	
5	LED Unit Blue	
6	LED Unit White	
7	Alarm Pattern	Off：0 On：1 No change：9

e.g.)：java Main.java S 1 2 9 0 0 1

3.1.3. Clear Command

Execute command with the following command line arguments

No.	Command Line Argument	Value
1	Command ID	C

e.g.)：java Main.java C

3.1.4. Status Acquisition Command

Execute command with the following command line arguments

No.	Command Line Argument	Value
1	Command ID	G

e.g.)：java Main.java G

3.2. Method description

3.2.1. Method list

メソッド名	Explanation
Control	Constructor
SocketOpen	Connect to LR5-LAN
SocketClose	close the socket
SendCommand	send command
PNS_RunControlCommand	Send PNS command operation control commands
PNS_ClearCommand	Send clear PNS command
PNS_GetDataCommand	Send Pns Command Status Acquisition Command

3.2.2. Constructor

Function Name	Control(final String ip, final int port)	
Parameters	String ip	LR5-LAN IP address
	int port	LR5-LAN port number
Return Value	Instances of Control	
Explanation	Create an instance of the Control class that operates the LR5-LAN of the specified IP address.	

How to use functions	// Main function
	public static void main(String[] args) {
	    Control ctr = new Control ("192.168.10.1", 10000);
	}	

Remarks	Please refer to 「4.1Connect to LR5-LAN」For The Program Overview.	

3.2.3. Connect to LR5-LAN

Function Name	public int SocketOpen()	
Parameters	None	
Return Value	int	Success：0、Faild：Other than 0
Explanation	Connect to LR5-LAN with specified IP address and port number using socket communication	

How to use functions	// Main function
	public static void main(String[] args) {
	    Control ctr = new Control ("192.168.10.1", 10000);
	
	    // Connect to Server
	    int ret = ctr.SocketOpen();
	    if (ret == -1) {
	        return;
	    }
	}	

Remarks	Please refer to 「4.1Connect to LR5-LAN」For The Program Overview.	

3.2.4.　close socket

<table>
<tr><td>Function Name</td><td>public void SocketClose()</td></tr>
<tr><td>Parameters</td><td>None</td></tr>
<tr><td>Return Value</td><td>None</td></tr>
<tr><td>Explanation</td><td>Close the socket connected to LR5-LAN</td></tr>
<tr><td>How to use functions</td><td><pre><code>// Main function
public static void main(String[] args) {
    Control ctr = new Control (″192.168.10.1″, 10000);

    // Connect to Server
    int ret = ctr.SocketOpen();
    if (ret == -1) {
        return;
    }

    try {
        // Any processing
    } finally {
        // close socket
        ctr.SocketClose();
    }
}</code></pre></td></tr>
<tr><td>Remarks</td><td>Please refer to 「4.2close socket」For The Program Overview.</td></tr>
</table>

3.2.5.　Send Command

Function Name	private byte[] SendCommand(final byte[] sendData)	
Parameters	byte[] sendData	Transmission Data
Return Value	byte[]	Received Data
Explanation	Send data to the connected LR5-LAN and return response data	
How to use functions	※Only available in the Control Class	
Remarks	Please refer to 「4.3Send Command」For The Program Overview.	

3.2.6. PNS Command Operation Control Command Transmission

Function Name	public int PNS_RunControlCommand(final PNS_RUN_CONTROL_DATA runControlData)	
Parameters	PNS_RUN_CONTROL_DATA runControlData	TRANSMISSION DATA THAT CONTROLS EACH COLOR PATTERN AND BUZZER OF THE LED UNIT For Details, See 「3.3.1Operation control data」For The Program Overview.
Return Value	Int	Success：0、Faild：Other than 0
Explanation	Send PNS command operation control commands to control each color pattern and buzzer of the led unit	

How to use functions	// Main function
	public static void main(String[] args) {
	  Control ctr = new Control (″192.168.10.1″, 10000);
	  // Connect to Server
	  int ret = ctr.SocketOpen();
	  if (ret == -1) {
	    return;
	  }
	  try {
	   // PNS Command Operation Control Command Transmission
	   // Led pattern0：Off
	   // Led pattern1：On
	   // Led pattern2：Flashing(slow)
	   // Led pattern3：Flashing(medium)
	   // Led pattern4：Flashing(fast)
	   // Led pattern5：Single flash
	   // Led pattern6：Double flash
	   // Led pattern7：Triple flash
	   // Led pattern9：No change
	   // Alarm Pattern0：Off
	   // Alarm Pattern1：On
	   // Alarm Pattern9：No change
	   Control.PNS_RUN_CONTROL_DATA runControlData = ctr.new PNS_RUN_CONTROL_DATA();
	   runControlData.ledRedPattern = Control.PNS_RUN_CONTROL_LED_ON;
	   runControlData.ledAmberPattern = Control.PNS_RUN_CONTROL_LED_BLINKING_SLOW;
	   runControlData.ledGreenPattern = Control.PNS_RUN_CONTROL_LED_NO_CHANGE;
	   runControlData.ledBluePattern = Control.PNS_RUN_CONTROL_LED_OFF;
	   runControlData.ledWhitePattern = Control.PNS_RUN_CONTROL_LED_FLASHING_TRIPLE;
	   runControlData.buzzerPattern = Control.PNS_RUN_CONTROL_BUZZER_RING;
	   ctr.PNS_RunControlComman(runControlData);
	  } finally {
	  // close socket
	  ctr.SocketClose();
	  }	

	}
Remarks	Please refer to「エラー! 参照元が見つかりません。エラー! 参照元が見つかりません。」For The Program Overview.

3.2.7. Send Clear Command For PNS Command

Function Name	public int PNS_ClearCommand()	
Parameters	None	
Return Value	Int	Success：0、Faild：Other than 0
Explanation	Send the PNS clear command to turn off the led unit and stop the buzzer	

How to use functions	// Main function
	public static void main(String[] args) {
	    Control ctr = new Control ("192.168.10.1", 10000);
	
	    // Connect to Server
	    int ret = ctr.SocketOpen();
	    if (ret == -1) {
	        return;
	    }
	
	    try {
	        // Send Clear Command For PNS Command
	        PNS_ClearCommand();
	    } finally {
	        // close socket
	        ctr.SocketClose();
	    }
	}	

Remarks	Please refer to 「4.5Send Clear Command For PNS Command」For The Program Overview.	

3.2.8. Send Pns Command Status Acquisition Command

Function Name	public PNS_STATUS_DATA PNS_GetDataCommand()	
Parameters	None	
Return Value	PNS_STATUS_DATA	Status Acquisition Command の Received Data(LED UNIT AND BUZZER STATUS) For Details, See 「3.3.3Operation control status data」For The Program Overview.
Explanation	Send the status acquisition command of the PNS command to acquire the status of the led unit and buzzer.	

How to use functions	// Main function
	public static void main(String[] args) {
	    Control ctr = new Control ("192.168.10.1", 10000);
	
	    // Connect to Server
	    int ret = ctr.SocketOpen();
	    if (ret == -1) {
	        return;
	    }
	
	    try {
	        // Send Pns Command Status Acquisition Command
	        Control.PNS_STATUS_DATA statusData = ctr.PNS_GetDataCommand();
	    } finally {
	        // close socket
	        ctr.SocketClose();
	    }
	}	

Remarks	Please refer to 「4.6Send Pns Command Status Acquisition Command」For The Program Overview.	

3.3. Constant Description

3.3.1. Product Differentiation

Constant name	Value	Explanation
PNS_PRODUCT_ID	0x4142	LR5-LAN product classification

3.3.2. PNS Command Identifier

Constant name	Value	Explanation
PNS_RUN_CONTROL_COMMAND	X053	Operation control command
PNS_CLEAR_COMMAND	0x43	Clear Command
PNS_REBOOT_COMMAND	0x42	restart command

3.3.3. PNS Command Response Data

Constant name	Value	Explanation
PNS_ACK	0x06	Normal Response
PNS_NAK	0x15	Abnormal Response

3.3.4. LED unit pattern for operation control commands

Constant name	Value	Explanation
PNS_RUN_CONTROL_LED_ON	0x00	Off
PNS_RUN_CONTROL_LED_OFF	0x01	On
PNS_RUN_CONTROL_LED_BLINKING_SLOW	0x02	Flashing(slow)
PNS_RUN_CONTROL_LED_BLINKING_MEDIUM	0x03	Flashing(slow)
PNS_RUN_CONTROL_LED_BLINKING_HIGH	0x04	Flashing(slow)
PNS_RUN_CONTROL_LED_FLASHING_SINGLE	0x05	Single flash
PNS_RUN_CONTROL_LED_FLASHING_DOUBLE	0x06	Double flash
PNS_RUN_CONTROL_LED_FLASHING_TRIPLE	0x07	Triple flash
PNS_RUN_CONTROL_LED_NO_CHANGE	0x09	No change

3.3.5. Buzzer pattern for operation control commands

Constant name	Value	Explanation
PNS_RUN_CONTROL_BUZZER_STOP	0x00	Off
PNS_RUN_CONTROL_BUZZER_RING	0x01	On
PNS_RUN_CONTROL_BUZZER_NO_CHANGE	0x09	No change

3.4. Data class description

3.4.1. Motion control data class

Name	PNS_RUN_CONTROL_DATA

Definition	public class PNS_RUN_CONTROL_DATA {
	    /** LED Unit Red pattern */
	    public byte ledRedPattern = 0;
	    /** LED Unit Amber pattern */
	    public byte ledAmberPattern = 0;
	    /** LED Unit Green pattern */
	    public byte ledGreenPattern = 0;
	    /** LED Unit Blue pattern */
	    public byte ledBluePattern = 0;
	    /** LED Unit White pattern */
	    public byte ledWhitePattern = 0;
	    /** Buzzer status */
	    public byte buzzerMode = 0;
	}

Explanation	Each pattern of the LED unit and the buzzer status in the data area sent by the operation control command

3.4.2. Operation control status data

Name	PNS_STATUS_DATA

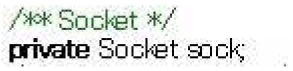
Definition	public class PNS_STATUS_DATA {
	    /** Led pattern1～5 */
	    public byte[]ledPattern = new byte[8];
	    /** Buzzer mode */
	    public byte buzzer = 0;
	}

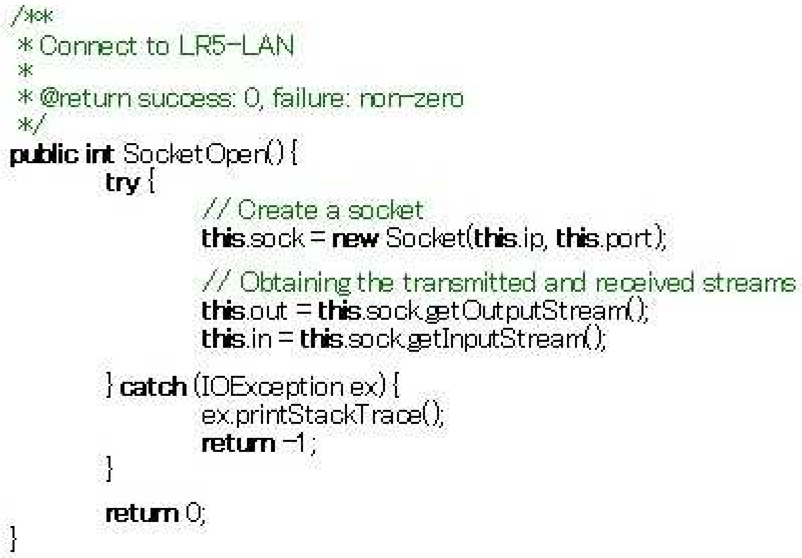
Explanation	Data class of LED unit and buzzer status of response data of operation control status acquisition command

4. Program Overview

Describe only the main points of the program’s operation.。

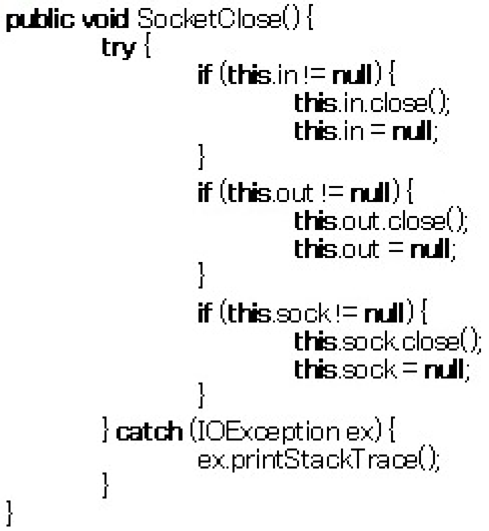
4.1. Connect to LR5-LAN

Program	Explanation
main.java  /** Socket */ private Socket sock;	→ Define the socket member variables

main.java SocketOpen()

/**
* Connect to LR5-LAN
*
* @return success: 0, failure: non-zero
*/
public int SocketOpen() {
try {
// Create a socket
this.sock = new Socket(this.ip, this.port);
// Obtaining the transmitted and received streams
this.out = this.sock.getOutputStream();
this.in = this.sock.getInputStream();
} catch (IOException ex) {
ex.printStackTrace();
return -1;
}
return 0;
}
	→Create a socket and connect
	→ Acquiring the sending and receiving stream

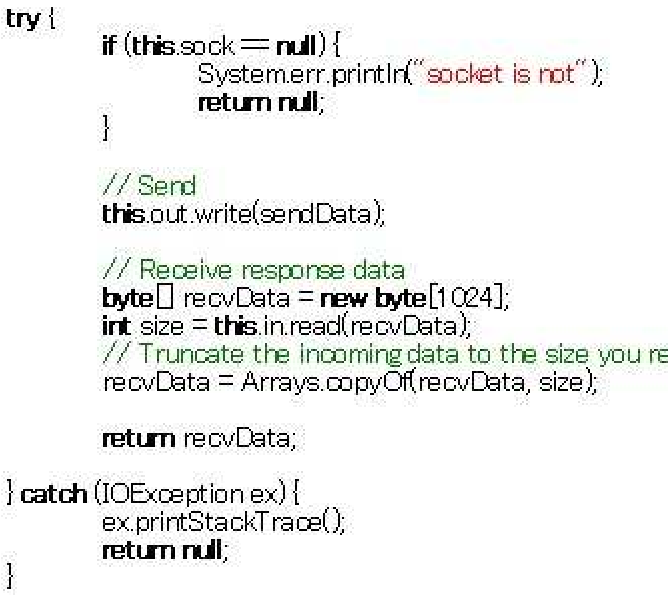
4.2. close socket

Program	Explanation

main.java SocketClose()

public void SocketClose() {
try {
if (this.in != null) {
this.in.close();
this.in = null;
}
if (this.out != null) {
this.out.close();
this.out = null;
}
if (this.sock != null) {
this.sock.close();
this.sock = null;
}
} catch (IOException ex) {
ex.printStackTrace();
}
}
	→Call the close method of the receiving stream
	→Call the close method of the sending stream.
	→Call the socket’s close method

4.3. Send Command

Create transmission data in the transmission data format for each command and send the command data to LR5-LAN Please refer to「4.4PNS Command Operation Control Command Transmission」and onwards for the transmission data format of each command.

<table>
<tr><th>Program</th><th>Explanation</th></tr>
<tr><td>main.java SendCommand()

<pre><code>try {
        if (this.sock == null) {
                System.err.println(“socket is not”);
                return null;
        }

        // Send
        this.out.write(sendData);

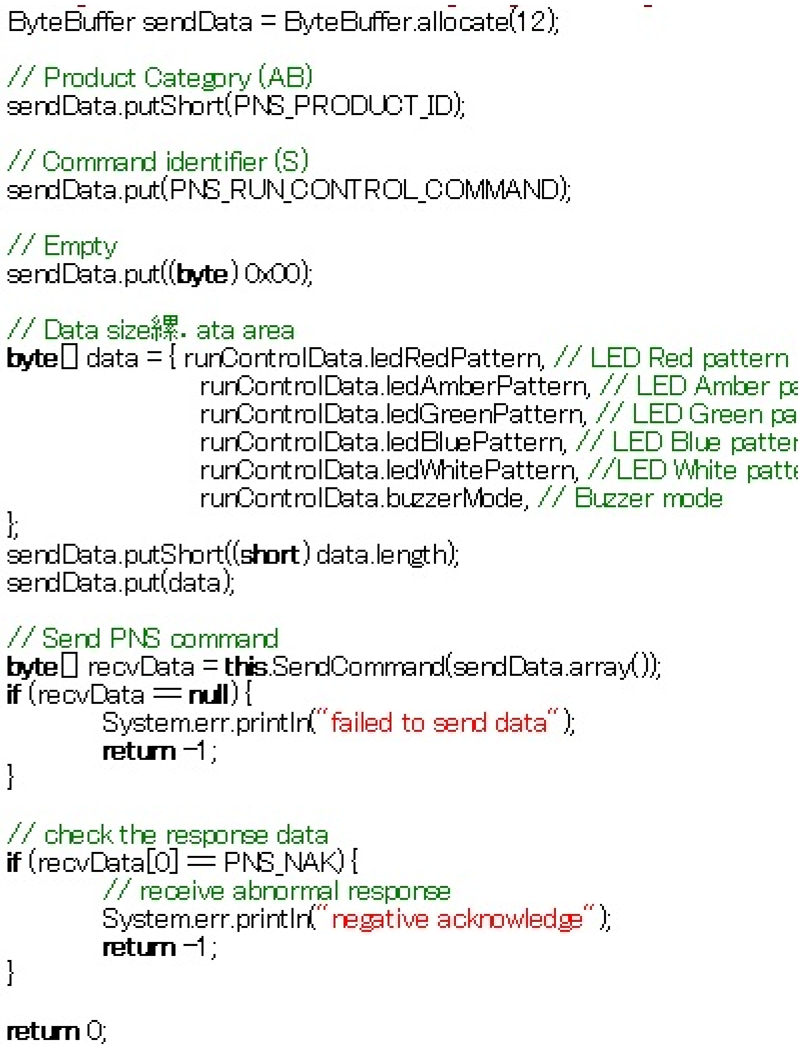
        // Receive response data
        byte[] recvData = new byte[1024];
        int size = this.in.read(recvData);
        // Truncate the incoming data to the size you re
        recvData = Arrays.copyOf(recvData, size);

        return recvData;

} catch (IOException ex) {
        ex.printStackTrace();
        return null;
}</code></pre></td><td>→Send the created transmission data using the send method
→Get the response from the device using the recv method after sending</td></tr>
</table>

4.4. PNS Command Operation Control Command Transmission

<table>
<tr><th>Program</th><th>Explanation</th></tr>
<tr><td>
main.java PNS_RunControlCommand()



<pre>
ByteBuffer sendData = ByteBuffer.allocate(12);

// Product Category (AB)
sendData.putShort(PNS_PRODUCT_ID);

// Command identifier (S)
sendData.put(PNS_RUN_CONTROL_COMMAND);

// Empty
sendData.put((byte) 0x00);

// Data size繆. ata area
byte[] data = { runControlData.ledRedPattern, // LED Red pattern
                runControlData.ledAmberPattern, // LED Amber pa
                runControlData.ledGreenPattern, // LED Green pa
                runControlData.ledBluePattern, // LED Blue patter
                runControlData.ledWhitePattern, //LED White patte
                runControlData.buzzerMode, // Buzzer mode
};
sendData.putShort((short) data.length);
sendData.put(data);

// Send PNS command
byte[] recvData = this.SendCommand(sendData.array());
if (recvData == null) {
        System.err.println("failed to send data");
        return -1;
}

// check the response data
if (recvData[0] == PNS_NAK) {
        // receive abnormal response
        System.err.println("negative acknowledge");
        return -1;
}

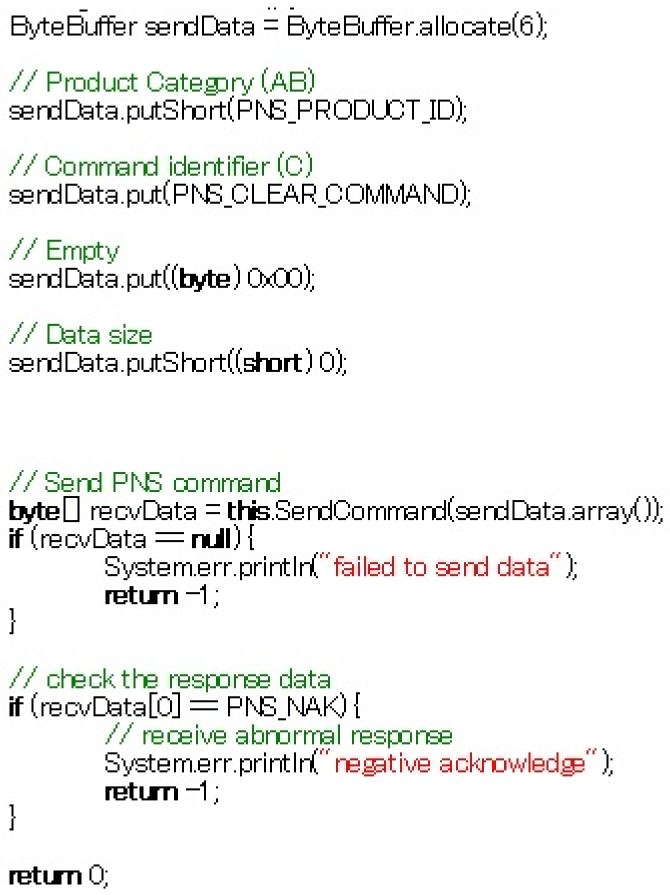
return 0;
</pre>
</td><td>
Create Transmission Data in the following order
→1st byte：Product Differentiation（A：0x41）
→2nd byte：Product Differentiation（B：0x42）
→3rd byte：ID（S：0x53）
→4th byte：Unused（0x00）
→5th byte：Data Size（0x00）
→6th byte：Data Size（0x06）
→7～12nd byte：Data Area
Data size is 6 bytes
Set the value of ″3.3.1 Motion control data structure″ in the Data Area.

→Call ″4.3 Send Command″ to send data to the device

→Check response data after sending
Normal Response：ACK(0x06)
Abnormal Response：NAK(0x15)
</td></tr>
</table>

4.5. Send Clear Command For PNS Command

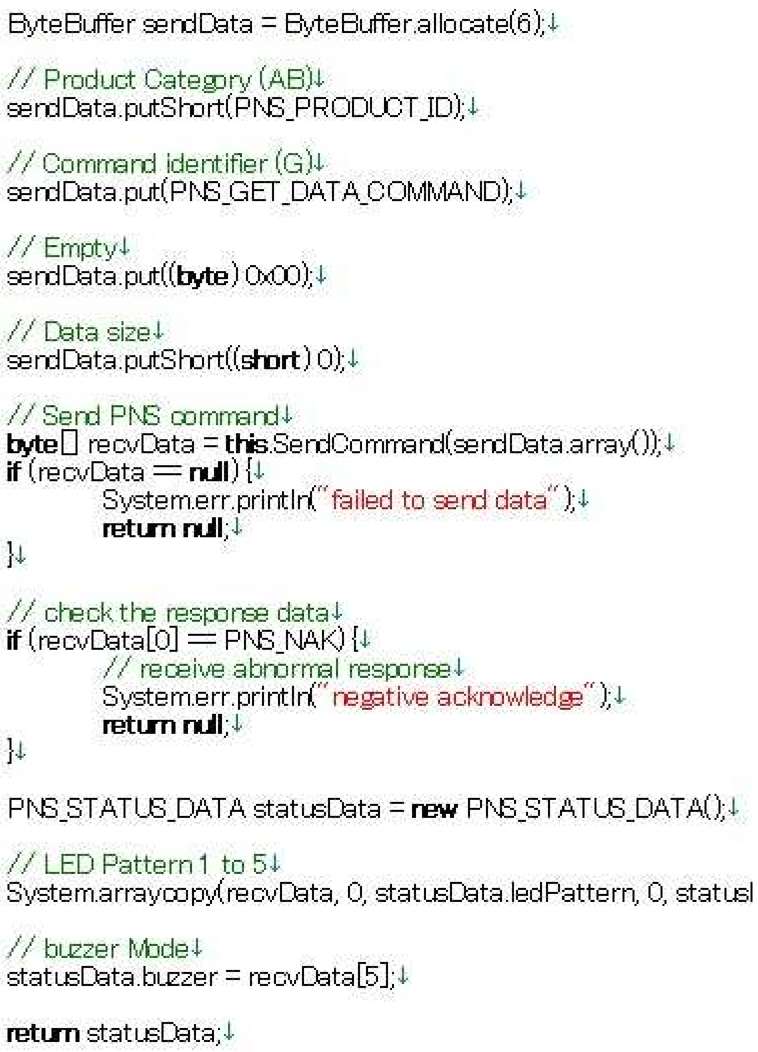
Program	Explanation

main.java PNS_ClearCommand()

ByteBuffer sendData = ByteBuffer.allocate(6);
// Product Category (AB)
sendData.putShort(PNS_PRODUCT_ID);
// Command identifier (C)
sendData.put(PNS_CLEAR_COMMAND);
// Empty
sendData.put((byte) 0x00);
// Data size
sendData.putShort((short) 0);
// Send PNS command
byte[] recvData = this.SendCommand(sendData.array());
if (recvData == null) {
        System.err.println("failed to send data");
        return -1;
}
// check the response data
if (recvData[0] == PNS_NAK) {
        // receive abnormal response
        System.err.println("negative acknowledge");
        return -1;
}
return 0;
	Create Transmission Data in the following order
	→1st byte：Product Differentiation（A：0x41）
	→2nd byte：Product Differentiation（B：0x42）
	→3rd byte：ID（C：0x43）
	→4th byte：Unused（0x00）
	→5th byte：Data Size（0x00）
	→6th byte：Data Size（0x00）
	Data size is 0 bytes
	No data area
	→Call "4.3 Send Command" to send data to the device
	→Check response data after sending
	Normal Response：ACK(0x06)
	Abnormal Response：NAK(0x15)

4.6. Send Pns Command Status Acquisition Command

<table>
<tr><th>Program</th><th>Explanation</th></tr>
<tr><td>

main.java PNS_GetDataCommand()



```
ByteBuffer sendData = ByteBuffer.allocate(6);↓

// Product Category (AB)↓
sendData.putShort(PNS_PRODUCT_ID);↓

// Command identifier (G)↓
sendData.put(PNS_GET_DATA_COMMAND);↓

// Empty↓
sendData.put((byte) 0x00);↓

// Data size↓
sendData.putShort((short) 0);↓

// Send PNS command↓
byte[] recvData = this.SendCommand(sendData.array());↓
if (recvData == null) {↓
        System.err.println("failed to send data");↓
        return null;↓
}↓

// check the response data↓
if (recvData[0] == PNS_NAK) {↓
        // receive abnormal response↓
        System.err.println("negative acknowledge");↓
        return null;↓
}↓

PNS_STATUS_DATA statusData = new PNS_STATUS_DATA();↓

// LED Pattern 1 to 5↓
System.arraycopy(recvData, 0, statusData.ledPattern, 0, statusl

// buzzer Mode↓
statusData.buzzer = recvData[5];↓

return statusData;↓
```

</td><td>

Create Transmission Data in the following order
→1st byte：Product Differentiation（A：0x41）
→2nd byte：Product Differentiation（B：0x42）
→3rd byte：ID（G：0x47）
→4th byte：Unused（0x00）
→5th byte：Data Size（0x00）
→6th byte：Data Size（0x00）
Data size is 0 bytes
No data area
→Call "4.3 Send Command" to send data to the device

→Check response data after sending
Normal Response：ACK(0x06)
Abnormal Response：NAK(0x15)

Acquire each data of response data using the following process.
→LED UNIT STATUS
・1st byte：LED Unit Red status
・2nd byte：LED Unit Amber status
・3rd byte：LED Unit Green status
・4th byte：LED Unit Blue status
・5th byte：LED Unit White status
・6th byte：Buzzer Status

</td></tr>
</table>