LR5-LAN Socket Communication

Sample Program

(Windows C#)

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 Visual C# control without using the DLLs provided by PATLITE.Microsoft

1.1. System Overview

The system configuration diagram of this program is as follows.

The sample program controls one LR5-LAN by socket communication.

2. Development Environment

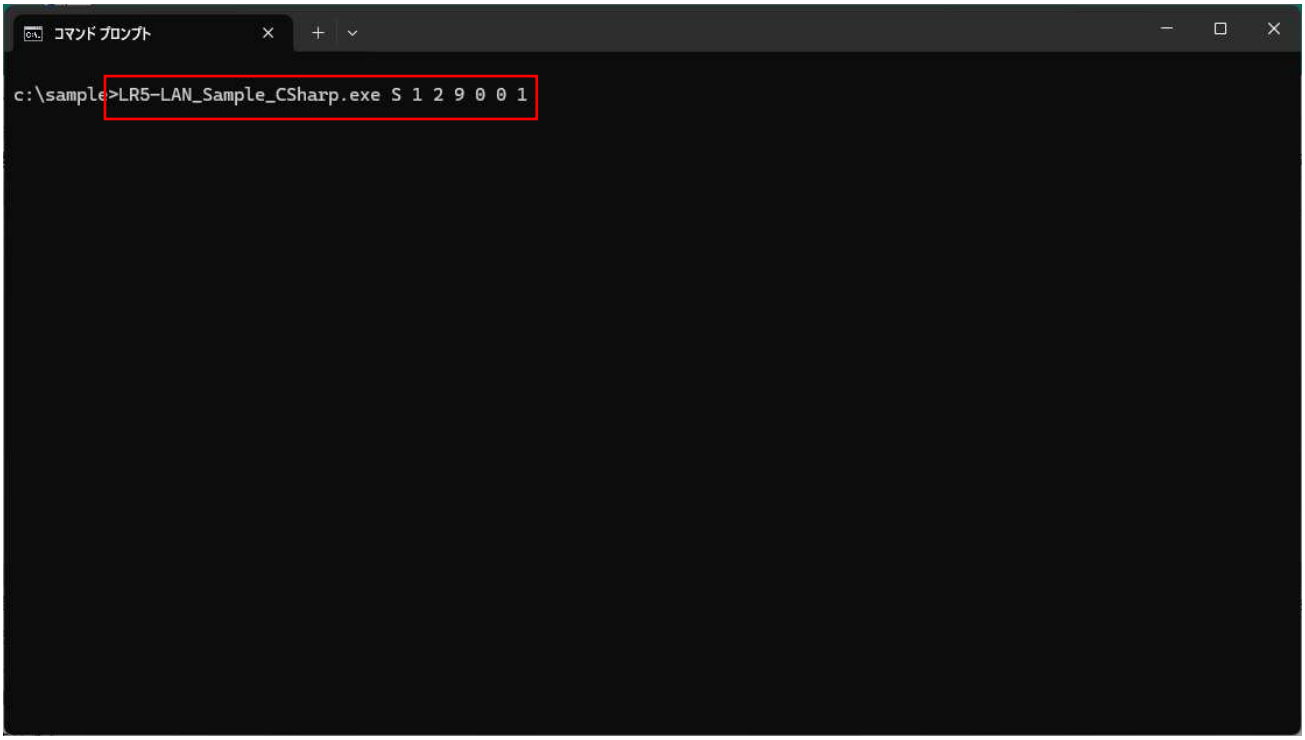
The development environment of the sample program is shown below.

Development Environment		Remarks
Development OS	Windows11 64bit	
Development Language	C#	.Net Framework 4.5 Subsequent
Application	CUI APPLICATION	
Development tool	VisualStudio2022 Professional	

3. Application Overview

3.1. Command Operation

Open Command Prompt, navigate to where the LR5-LAN_Sample_CSharp.exe created during the build is located and specify the command line arguments to execute commands for each operation.



3.1.1. Command list

command name	content
Operation control command	Control each color pattern and buzzer (On/Off) of the LED unit
Clear Command	Turn off the LED unit and turn off the buzzer
Status Acquisition Command	Status of signal lines/contact inputs and 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.):LR5-LAN_Sample_CSharp.exe 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.):LR5-LAN_Sample_CSharp.exe 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.):LR5-LAN_Sample_CSharp.exe G

3.2. Function Description

3.2.1. Function List

Function Name	Explanation
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. Connect to LR5-LAN

Function Name	public static int SocketOpen(string ip, int port)	
Parameters	string ip	LR5-LAN IP address
	int port	LR5-LAN port number
Return Value	int	Success: 0, Failure: other than 0
Explanation	Connect to LR5-LAN with specified IP address and port number using socket communication	

How to use functions	// Definition of Socket class variables
	private static Socket sock = null;
	
	// Main function
	static void Main()
	{
	  // Connect to LR5-LAN
	  ret = SocketOpen("192.168.10.1", 10000);
	  if (ret == -1)
	  {
	  return;
	  }
	}	

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

3.2.3. close socket

Function Name	public static void SocketClose()
Parameters	None
Return Value	None
Explanation	CLOSE THE SOCKET CONNECTED TO LR5-LAN

How to use functions	// Main function
	static void Main()
	{
	  // Connect to LR5-LAN
	  ret = SocketOpen("192.168.10.1", 10000);
	  if (ret == -1) {
	  return;
	  }
	
	  // close socket
	  SocketClose();
	}

Remarks	Please refer to 「4.2close socket」For The Program Overview.

3.2.4. Send Command

Function Name	public static int SendCommand(byte[] sendData, out byte[] recvData)	
Parameters	byte[] sendData	Transmission Data
	out byte[] recvData	Received Data
Return Value	int	Success: 0, Failure: other than 0
Explanation	Send data to the connected LR5-LAN and return response data	

How to use functions	// Main function
	static void Main()
	{
	// Connect to LR5-LAN
	ret = SocketOpen("192.168.10.1", 10000);
	if (ret == -1) {
	return;
	}
	
	// Create transmission data
	byte[] sendData = new byte[7];
	byte[] recvData;
	sendData[0] = 0x41;
	sendData[1] = 0x42;
	sendData[2] = 0x53;
	sendData[3] = 0x00;
	sendData[4] = 0x00;
	sendData[5] = 0x00;
	sendData[6] = 0x01;
	
	// Send Command
	ret = SendCommand(sendData, out recvData);
	if (ret != 0) {
	Debug.WriteLine("failed to send data");
	return -1;
	}
	
	// close socket
	SocketClose();
	}	

Remarks	Please refer to 「4.3Send Command」For The Program Overview.	

3.2.5. PNS Command Operation Control Command Transmission

Function Name	public static int PNS_RunControlCommand(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「3.4.1Motion control data structure」For The Program Overview.
Return Value	int	Success: 0, Failure: 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
	static void Main()
	{
	// Connect to LR5-LAN
	ret = SocketOpen("192.168.10.1", 10000);
	if (ret == -1) {
	return;
	}
	
	// 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
	PNS_RUN_CONTROL_DATA runControlData = new PNS_RUN_CONTROL_DATA
	{
	ledRedPattern = PNS_RUN_CONTROL_LED_ON,
	ledAmberPattern = PNS_RUN_CONTROL_LED_BLINKING_SLOW,
	ledGreenPattern = PNS_RUN_CONTROL_LED_NO_CHANGE,
	ledBluePattern = PNS_RUN_CONTROL_LED_ OFF,
	ledWhitePattern = PNS_RUN_CONTROL_LED_FLASHING_TRIPLE,
	buzzerPattern = PNS_RUN_CONTROL_BUZZER_RING
	};
	PNS_RunControlCommand(runControlData);
	// close socket
	SocketClose();
	}	

Remarks	Please refer to「4.4PNS Command Operation Control Command Transmission」For The Program Overview.	

3.2.6. Send Clear Command For PNS Command

Function Name	public static int PNS_ClearCommand()	
Parameters	None	
Return Value	Int	Success: 0, Failure: 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
	static void Main()
	{
	// Connect to LR5-LAN
	ret = SocketOpen("192.168.10.1", 10000);
	if (ret == -1) {
	return;
	}
	
	// Send Clear Command For PNS Command
	PNS_ClearCommand();
	
	// close socket
	SocketClose();
	}	

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

3.2.7. Send PNS Command Status Acquisition Command

Function Name	public static int PNS_GetDataCommand(out PNS_STATUS_DATA statusData)	
Parameters	out PNS_STATUS_DATA statusData	Status Acquisition Command の Received Data(LED UNIT AND BUZZER STATUS) For Details, See 「3.4.2Operation control status data」For The Program Overview.
Return Value	Int	Success: 0, Failure: other than 0
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
	static void Main()
	{
	// Connect to LR5-LAN
	ret = SocketOpen("192.168.10.1", 10000);
	if (ret == -1) {
	return;
	}
	
	// Send PNS Command Status Acquisition Command
	PNS_STATUS_DATA statusData;
	PNS_GetDataCommand(out statusData);
	
	// close socket
	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	0x53	Operation control command
PNS_CLEAR_COMMAND	0x43	Clear Command
PNS_GET_DATA_COMMAND	0x47	Status Acquisition 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. Structure Description

3.4.1. Motion control data structure

名前	PNS_RUN_CONTROL_DATA

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

Explanation	Each pattern and buzzer status of the LED unit in the Data Area sent by the Operation control command

3.4.2. Operation control status data

名前	PNS_STATUS_DATA

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

Explanation	Operation control Status Acquisition Command response data LED UNIT AND BUZZER STATUS structure

4. Program Overview

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

4.1. Connect to LR5-LAN

Program	Explanation
Program.cs `private static Socket sock = null;`	→Definition of socket member variables

Program.cs SocketOpen()
`public static int SocketOpen(string ip, int port)`
`{`
`try`
`{`
`// Set the IP address and port`
`IPAddress ipAddress = IPAddress.Parse(ip);`
`IPEndPoint remoteEP = new IPEndPoint(ipAddress, po`
`// Create a socket`
`sock = new Socket(ipAddress.AddressFamily, SocketT`
`if (sock == null)`
`{`
`Debug.WriteLine("failed to create socket");`
`return -1;`
`}`
`// Connect to LA-POE`
`sock.Connect(remoteEP);`
`}`
`catch(Exception ex)`
`{`
`Debug.WriteLine(ex.Message);`
`SocketClose();`
`return -1;`
`}`
`return 0;`
`}`
	→Specify the device IP address and port number
	Default IP address: 192.168.10.1
	Default port number: 10000
	→Create a socket instance
	→Connect to the device using the socket Connect function

4.2. close socket

Program	Explanation

Program.cs SocketClose()
 	
`public static void SocketClose()`
`{`
`  if (sock != null)`
`  {`
`  // Close the socket.`
`  sock.Shutdown(SocketShutdown.Both);`
`  sock.Close();`
`  }`
 	
`}`
	→Shut down the socket and then call close

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.

Program	Explanation

Program.cs SendCommand()
 	
`if (sock == null)`
`{`
`  Debug.WriteLine("socket is not");`
`  return -1;`
`}`
 	
`// Send`
`ret = sock.Send(sendData);`
`if (ret < 0)`
`{`
`  Debug.WriteLine("failed to send");`
`  return -1;`
`}`
 	
`// Receive response data`
`byte[] bytes = new byte[1024];`
`int recvSize = sock.Receive(bytes);`
`if (recvSize < 0)`
`{`
`  Debug.WriteLine("failed to recv");`
`  return -1;`
`}`
`recvData = new byte[recvSize];`
`Array.Copy(bytes, recvData, recvSize);`
	→Send the created Transmission Data using the Send function
	
	→After sending, use the Receive function to get the response from the device.

4.4. PNS Command Operation Control Command Transmission

Program	Explanation

Program.cs PNS_RunControlCommand()
 	
`byte[] sendData = { };`
 	
`// Product Category (AB)`
`sendData = sendData.Concat(BitConverter.GetBytes(PNS_PRODU`
 	
`// Command Identifier(S)`
`sendData = sendData.Concat(new byte[] { PNS_RUN_CONTROL_CO`
 	
`// Empty(0)`
`sendData = sendData.Concat(new byte[] { 0 }).ToArray();`
 	
`// data size, data area`
`byte[] data = {`
`runControlData.ledRedPattern, // LED Red pattern`
`runControlData.ledAmberPattern, // LED Amber patte`
`runControlData.ledGreenPattern, // LED Green patte`
`runControlData.ledBluePattern, // LED Blue pattern`
`runControlData.ledWhitePattern, // LED White patte`
`runControlData.buzzerMode // Buzzer mode`
`};`
`sendData = sendData.Concat(BitConverter.GetBytes((ushort)d`
`sendData = sendData.Concat(data).ToArray();`
 	
`// Send PNS command`
`byte[] recvData;`
`ret = SendCommand(sendData, out recvData);`
`if (ret != 0)`
`{`
`Console.Write("failed to send data");`
`return -1;`
`}`
 	
`// check the response data`
`if (recvData[0] == PNS_NAK)`
`{`
`// receive abnormal response`
`Console.Write("negative acknowledge");`
`return -1;`
`}`
	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)
	→7～1 : Data Area
	Data size is 6 bytes
	Set the value of "3.4.1 Motion control data structure" in the Data Area.
	
	→Call "4.3 Send Command/Receive" and send data to the device
	
	→Check response data after sending
	Normal Response : ACK (0x06)
	Abnormal Response : NAK (0x15)

4.5. Send Clear Command For PNS Command

Program	Explanation

Program.cs PNS_ClearCommand()
`byte[] sendData = { };`
 	
`// Product Category (AB)`
`sendData = sendData.Concat(BitConverter.GetBytes(PNS_PRODU`
 	
`// Command identifier (C)`
`sendData = sendData.Concat(new byte[] { PNS_CLEAR_COMMAND`
 	
`// Empty (0)`
`sendData = sendData.Concat(new byte[] { 0 }).ToArray();`
 	
`// Data size`
`sendData = sendData.Concat(BitConverter.GetBytes((ushort)0`
 	
`// Send PNS command`
`byte[] recvData;`
`ret = SendCommand(sendData, out recvData);`
`if (ret != 0)`
`{`
`  Debug.WriteLine("failed to send data");`
`  return -1;`
`}`
 	
`// check the response data`
`if (recvData[0] == PNS_NAK)`
`{`
`  // receive abnormal response`
`  Debug.WriteLine("negative acknowledge");`
`  return -1;`
`}`
	Create Transmission Data in the following order
	→1st byte：Product Differentiation（A：0x41）
	→：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/Receive” and 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

Program	Explanation

Program.cs PNS_GetDataCommand()
 	
`byte[] sendData = { };`
 	
`// Product Category (AB)`
`sendData = sendData.Concat(BitConverter.GetBytes(PNS_PRODU`
 	
`// Command identifier (G)`
`sendData = sendData.Concat(new byte[] { PNS_GET_DATA_COMMAI`
 	
`// Empty (0)`
`sendData = sendData.Concat(new byte[] { 0 }).ToArray();`
 	
`// Data size`
`sendData = sendData.Concat(BitConverter.GetBytes((short)0)`
 	
`// Send PNS command`
`byte[] recvData;`
`ret = SendCommand(sendData, out recvData);`
`if (ret != 0)`
`{`
`  Console.Write("failed to send data");`
`  return -1;`
`}`
 	
`// check the response data`
`if (recvData[0] == PNS_NAK)`
`{`
`  // receive abnormal response`
`  Console.Write("negative acknowledge");`
`  return -1;`
`}`
 	
`// LED Pattern 1~5`
`statusData.input = new byte[5];`
`Array.Copy(recvData, statusData.ledPattern, statusData.ledI`
 	
`// Buzzer Mode`
`statusData.buzzer = recvData[5];`
	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/Receive” and send data to the device
	
	→Check response data after sending
	Normal Response: Response data of ″3.3.3Operation control status data″ is obtained.
	Abnormal Response：NAK（0x15）
	
	Acquire each data of response data using the following process.
	→LED UNIT STATUS
	・1st byte：LED Unit Redstatus
	・2nd byte：LED Unit Amberstatus
	・3rd byte：LED Unit Greenstatus
	・4th byte：LED Unit Bluestatus
	・5th byte：LED Unit Whitestatus
	・6th byte：Buzzer status