Technical Information

YS-net
Peer-to-Peer Communication Functions

YS100 Series

Contents

YOKOGAWA ◆
Yokogawa Electric Corporation

TI 1B7C8-04E

3rd Edition: Oct. 2004 (KP)

Introduction

This technical information describes the peer-to-peer communication functions in the YS100 Series Programmable Single-loop Controller. It also briefly introduces YS-net, the vehicle that enables the communication functions to work.

- Document Configuration

This information contains the following four chapters:

Chapter 1 gives an overview of the YS-net, peer-to-peer communication functions, and the model and suffix codes of the YS170 Controller.

Chapter 2 explains how to wire the YS-net and set up the communication functions of the YS170 Controller.

Chapter 3 explains how to program the peer-to-peer communication functions and how to handle possible failures.

Chapter 4 discusses specific examples of the programs of the peer-to-peer communication functions.

- Applicable Readers

This technical information is written for instrumentation engineers who are responsible for creating the user programs of the YS170. Consequently, readers need to have sufficient background knowledge and experience to be able to create these programs. The readers are not expected to know how the communication is implemented.

- Revision Record

Date	Edition	Contents
June 1995	First Edition	New publication
February 1997	2nd Edition	Enhancement (Register CI and CO added)
October 2004	3rd Edition	Change of the company name

Table 0.1 YS100 Series Document Map

Document Class	Document No.	Title	Usage (◎ : Essential , ○ : For Reference)				
			Programming for YS170	Engineering for function selections and parameter settings	Tuning	Normal Operation	Installation and Maintenance
Technical Information	TI 1B7A1-01E	YS100 SERIES Information	○	○		○	
	TI 1B7C0-01E Note 2	YS100 SERIES Intelligent Self-tuning Controllers			◎		◎
	TI 1B7C1-01E	YS150, YS170 Single-loop Controller Control Functions	◎	◎	◎	◎	
	TI 1B7C2-03E Note 3	YS170 Programmable Functions	◎		○		
	TI 1B7C8-03E Note 1	YS100 SERIES Communication Functions		◎		◎	
	TI 1B7C8-04E Note 5	YS-net Peer-to-peer Communication Functions		◎			
	TI 1B7C8-05E Note 5	YS-net Personal Computer Communication Functions		◎		◎	
Instruction Manual	IM 1B7C1-01E	YS150 Single-loop Multi-function Controller YS170 Single-loop Programmable Controller	○	◎	◎	◎	◎
	IM 1B7C8-01E	YSS10 YS100 SERIES Programming Package	◎				
	IM 1B7C8-03E Note 1	YS100 SERIES RS-485 Communication Functions (/A31) DCS-LCS Communication Functions (/A32)		◎		◎	◎
	IM 1B7D2-01E	YS131 Indicator with Alarm		◎	◎	◎	◎
	IM 1B7D3-01E	YS135 Auto/Manual Station for SV Setting		◎	◎	◎	◎
	IM 1B7D4-01E	YS136 Auto/Manual Station for MV Setting		◎	◎	◎	◎
	IM 1B7D5-01E Note 4	YS110 Standby Manual Station				◎	

Note 1: Only when used with supervisory communication functions

Note 2: Only when using self-tuning functions

Note 3: Only for YS170 programmable controllers

Note 4: The YS110 can be a standby station only for the YS150, YS170, or YS136

Note 5: Only when using YS-net communication functions

1. Overview

This chapter explains the functions of the YS-net, the functions and specifications of the peer-to-peer communication, and the model, suffix codes and option codes of the YS-net supported YS170 Single-loop Controller.

1.1 Overview of YS-net

The YS170 Single-loop Programmable Controller (hereafter called "YS170 Controller") is equipped with a YS-net communication card and thus offers two key functions: peer-to-peer communication and personal computer communication.

The peer-to-peer communication functions allow the YS170 Controller, when it is in programmable mode, to transmit and receive computation and control data via the YS-net.

The personal computer communication functions enable the YS170 Controller to communicate with a personal computer (hereafter called "PC"). This feature makes it possible to perform monitoring, operation and PID tuning from a PC.

The YS-net can be used in three ways:

(1) peer-to-peer communication only,
(2) personal computer communication only
(3) both peer-to-peer communication and personal computer communication.

This technical information specifically deals with the functions, specifications and programming method when using the peer-to-peer communication [(1)] (hereafter called "peer-to-peer communication only" mode). For details about the other modes [(2) and (3)], see the TI 1B7C8-05E technical information, "YS-net Personal Computer Communication Functions."

1.2 Peer-to-peer Communication Functions

In the peer-to-peer communication only mode, up to 16 units of the YS170 Controller can be linked to the YS-net, 4 of which are allowed to transmit 4 analog and 16 status data items and receive 16 analog and 64 status data items (called transmitting/receiving controllers). Each of the remaining 12 units can only receive 16 analog and 64 status data items (called receiving controllers).

The user can, without being aware of the presence of the YS-net communication network, perform data communication simply by reading data from the peer-to-peer communication register (data reception) or by writing data to the register (data transmission).

Specifications of Peer-to-peer Communication

Number of controllers that can be linked	: up to 16 (4 transmitting/receiving controllers and 12 receiving controllers)
Amount of data transmitted	: 4 analog and 16 status data items (for each data transmitting/receiving controller)
Amount of data received	: 16 analog and 64 status data items
Update interval of transmitted data	: 200 ms (asynchronous with the run interval of the user program)
Communication failure detection time	: 2 sec

Specifications of YS-net Communication

Transmission rate	: 78.125 bits/sec
Communication wiring	: twisted-pair (AWG22)

Connection : daisy chain
Distance : up to 1000 m

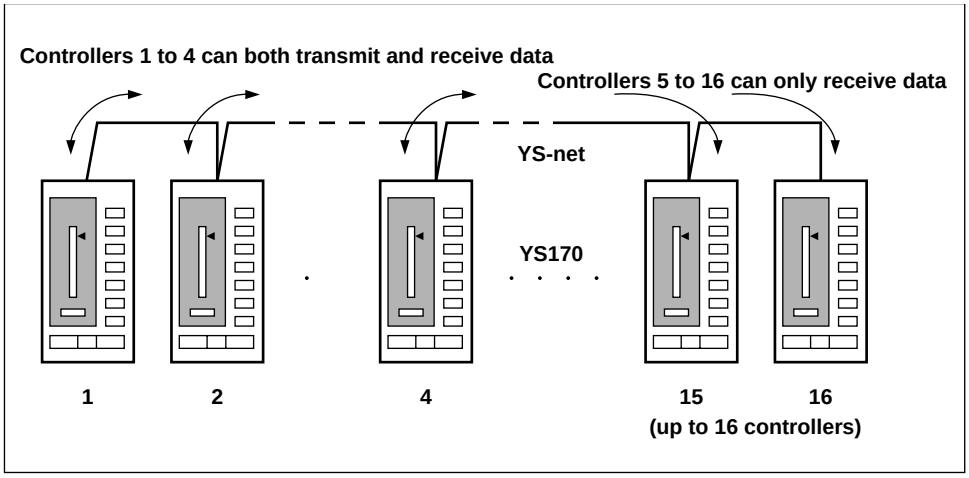

Figure 1.1 Functions in Peer-to-peer Communication Only Mode

1.3 Model and Suffix Codes

Model	Suffix Code	Option Code	Remarks
YS170			Single-loop programmable controller
Use	– 0		general purpose
	1		always 1
Power supply	1		100-V power supply
	2		220-V power supply
Option code (lists only those for communication)		/ A31	RS-485 communication
		/ A32	DCS-LCS communication
		/ A33	YS-net communication

2. Wiring and Setting Up the Controller for YS-net Communication

This chapter explains how to wire the YS-net communication cables and set up end-point controllers.

2.1 Connecting YS-net Communication Cables

The YS-net uses the terminals shown in Table 2.1 of those on the terminal block at the back of the controller.

Table 2.1 List of Communication Terminals

Terminal No.	YS-net CommunicationTerminal
17	DA
18	DB

2.2 Wiring to Communication Cables

Figure 2.1 shows how to wire controllers to the YS-net communication cables. YS170 controllers are linked with the communication line in a daisy-chain type of connection. Terminal 17 (DA) on one controller is connected to the equivalent on an adjacent controller; likewise, terminal 18 (DB) is connected to the equivalent on an adjacent controller.

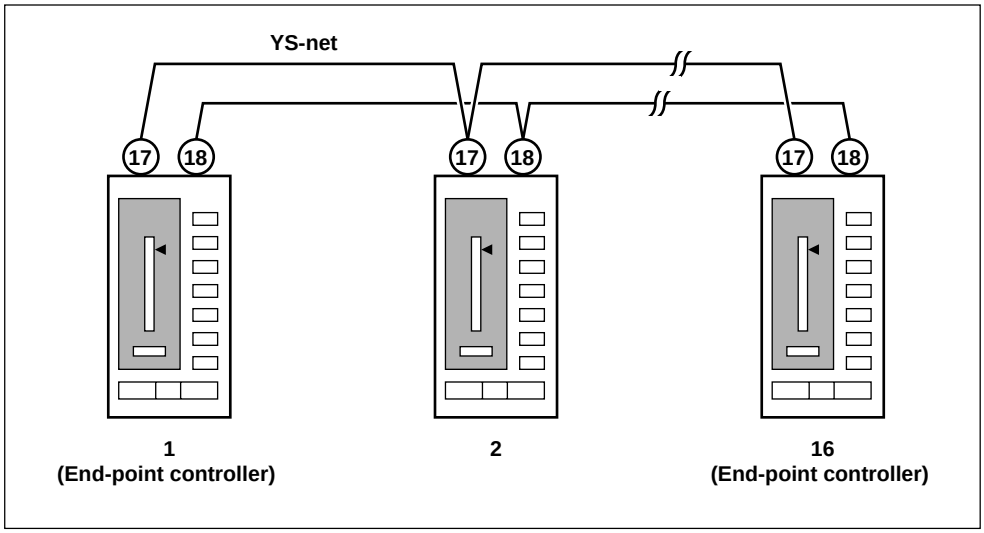

Figure 2.1 Link to the YS-net

2.3 Setting Up the End-point Controllers

The controllers at both ends of a YS-net communication cable (terminal points) require the end-points of communication to be set. To terminate either end of the communication line, remove the YS-net communication card from the controller and set the JP1 jumper connector inside the card (see Figure 2.2) to the ON position.

Do not terminate the YS-net communication line at a controller which is not at either end of the line (make sure that the JP1 jumper connector on that controller is in the OFF position).

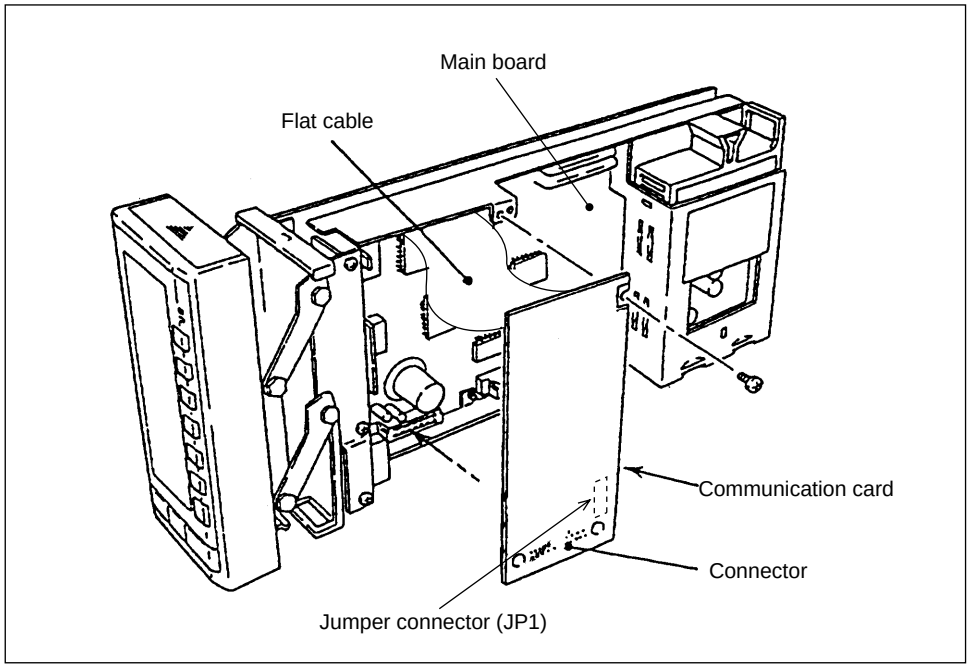

Figure 2.2 Setting an End-point Controller

3. Programming the Communication Functions

This chapter explains how to program the peer-to-peer communication functions. The procedure comprises two steps: setting the communication device numbers and describing the peer-to-peer communication in the user program.

Figure 3.1 shows the flow of programming. There is no specific order between setting the communication device number and describing the peer-to-peer communication in a user program; start working on either of these procedures.

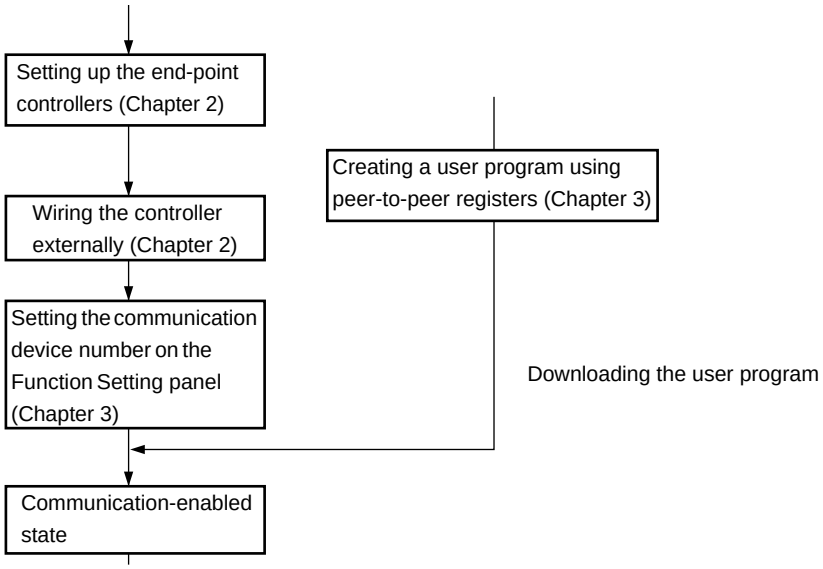

Figure 3.1 Flow of Programming

3.1 Setting the Communication Device Number

Set the communication device number in the ADRS field of Function Setting panel 1 (panel name: CONFIG 1) in a group of engineering panels. For details on how to select and set data items on a panel, see Chapter 9, "Engineering Operations," in the IM 1B7C1-01E instruction manual, "YS150 Single-loop Multi-function Controller, YS170 Single-loop Programmable Controller."

<Setting the device number>

ADRS: Set any number from 0 to 16. The meaning of each number is as follows:

0 = Do not use this number in the Peer-to-peer Communication Only mode. YS-net function can be enhanced with extra engineering. In that case number 0 is used.

1-4 = enables the controller to both transmit and receive data.

5-16 = allows the controller to only receive data.

Note : Do not set the same communication device number on two or more YS170 controllers. Instruments with the same device number will not be able to communicate properly.

3.2 Peer-to-peer Communication Registers

Data can be transmitted or received in the peer-to-peer communication as analog data or status data. The status data is represented either as OFF = 0 or as ON = 1, according to the rules of the user program.

Transmitted/received data can be used by the user program through peer-to-peer registers. Table 3.1 lists the peer-to-peer communication registers, and Table 3.2 shows the read (input)/write (output) instructions.

Table 3.1 Peer-to-peer Communication Registers

Register Name	Designation	Description	Range of Values
CXn	Peer-to-peer communication analog input register	n = 01 to 04: data received from device number 1 n = 05 to 08: data received from device number 2 n = 09 to 12: data received from device number 3 n = 13 to 16: data received from device number 4 n = 17 to 32: spare numbers (Note)	-800.0% to 800.0%
CYn	Peer-to-peer communication analog output register	n = 01 to 04: data transmitted to another controller n = 05 to 32: spare numbers (Note)	-800.0% to 800.0%
CIn	Status input registers for peer-to-peer communication	n = 01 to 16 : data received from device number 1 n = 17 to 32 : data received from device number 2 n = 33 to 48 : data received from device number 3 n = 49 to 64 : data received from device number 4 n = 65 to 96 : spare numbers (Note)	0(=0.0%) 1(=100.0%)
COn	Status output registers for peer-to-peer communication	n = 01 to 16 : data transmitted to another controller n = 17 to 96 : spare numbers (Note)	0(=0.0%) 1(=100.0%)
CFn	Reception timeout flag	n = 01 to 04: indicates whether the data received from device number n is normal or abnormal. n = 05 to 32: spare numbers (Note)	0(=0.0%): normal, 1(=100.0%): abnormal

Note: Do not use the spare registers; these registers are reserved for future functional expansion.

Table 3.2 Read/Write Commands in User Program

Command	Description
LD CXn	Reads CXn into computation register (S1)
LD CIn	Reads CIn into computation register (S1)
LD CFn	Reads CFn into computation register (S1)
ST CYn	Saves data in computation register (S1) to CYn
ST COn	Saves data in computation register (S1) to COn

3.3 Handling Communication Failures

This section explains how YS170 controllers respond if they fail to communicate properly.

Table 3.3 Causes of Communication Failures and Actions Taken

Item	Cause of Failure	Action at Receiving Controller	Action at Transmitting Controller
1	Broken wire on YS-net, or failed communication card of receiving controller	Retains the peer-to-peer communication input data received before failure. If the failure continues for at least 2 seconds, the reception timeout flag is set to 1 (fail).	Cannot detect the failure. However, when the transmitting controller receives data, it detects the failure as a receiving controller.
2	Downloading/uploading of user program or function setup in progress at transmitting controller	Same as item 1	Stops functioning.
3	Downloading/uploading of user program or function setup in progress at receiving controller	Stops functioning. However, the receiving controller still receives peer-to-peer communication data normally and saves the data to CX or CI registers.	Same as item 1
4	Failed transmitting controller	Same as item 1	Fails.
5	Power failure at transmitting controller	Same as item 1	Remains as a power failure. For actions upon power recovery, see the section below.
6	Power failure at receiving controller	Remains as a power failure. For actions upon power recovery, see the section below.	Same as item 1

3.4 Actions Upon Power Recovery

This section explains how a controller which is performing peer-to-peer communication behaves when it recovers from a power failure.

1) Cold Start and Initial Start

The controller restarts with the values of CX, CY, CI, and CO registers as 0%. If a transmitting controller or user program writes data to these registers, the data becomes effective. The reception timeout flag is set to 1 (fail) at the restart, then set to 0 (normal) when the communication link recovers.

2) Hot Start

The CX, CY, CI, and CO registers retain the data values received before the power failure. If a transmitting controller or user program writes data to these registers, the data becomes effective. The reception timeout flag is set to 1 (fail) at the restart, then set to 0 (normal) when the communication link recovers.

4. Program Examples

This chapter assumes the following peer-to-peer communication system:

- Three loops of controllers: Loop 1, Loop 2 and Loop 3.
- Loop 1 needs to have the setpoint value (SV1) and Cas/Auto status (CAF1 flag) of Loop 2.
- Loop 2 needs to have the manipulated output value (MV1) of Loop 1.
- Loop 3 needs to have the process variables (PV1) of Loop 1 and Loop 2.

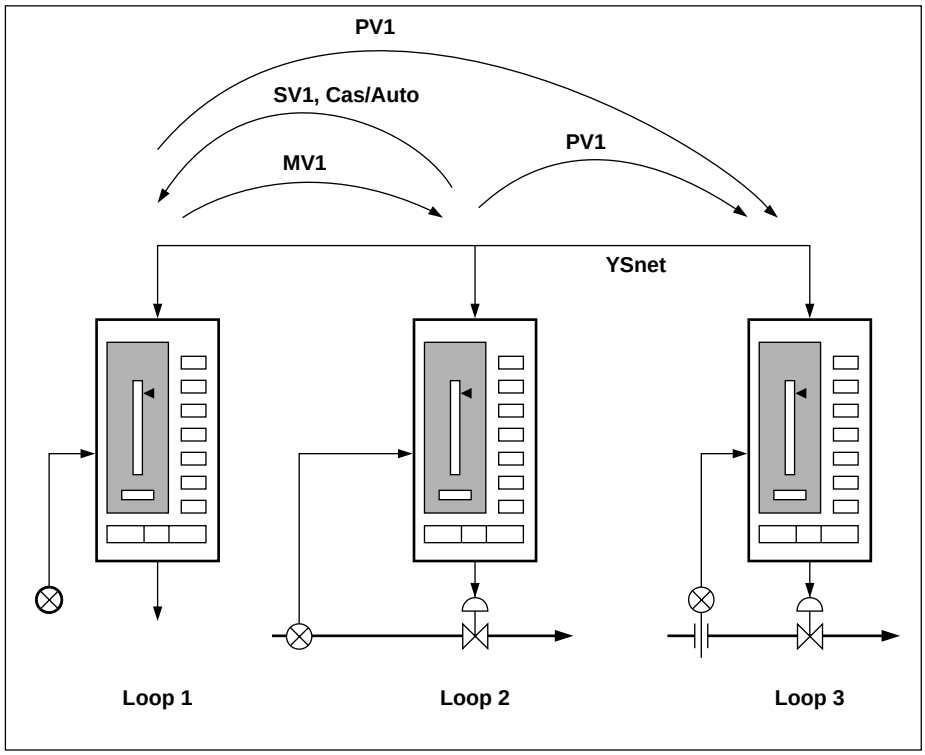

Figure 4.1 Example of Peer-to-peer Communication

4.1 Assigning Device Numbers

The above example assumes the device numbers to be 1 and 2, because Loops 1 and 2 need to both transmit and receive data. Loop 3 only receives data and is therefore given the device number 5. Since this example contains rather few controllers, Loop 3 can also have device number 3, a number for a transmitting/receiving controller.

4.2 Assigning Communication Registers

The data items shown in Table 4.1 have been assigned to the peer-to-peer communication registers of the respective controllers. The assignments to the communication input registers are shared by all the controllers (in this case "Loop1", "Loop2", and "Loop3") on the YS-net.

Table 4.1 Example of Assignments to the Peer-to-peer Registers of Transmitting Controllers

		Controller	
		Loop 1	Loop 2
Peer-to-peer analog output register	CY01	Process variable (PV1)	Process variable (PV1)
	CY02	Manipulated output variable (MV1)	Setpoint value (SV1)
	CY03	Unused	Cas/Auto status (CAF1)
	CY04	Unused	Unused

Table 4.2 Example of Assignments to the Communication Input Registers of Controllers on YS-net

		Data in Register
Peer-to-peer analog input register	CX01	CY01 of controller with device number 1 = process variable 1 (PV1) of Loop 1 controller
	CX02	CY02 of controller with device number 1 = manipulated output variable 1 (MV1) of Loop 1 controller
	CX03	CY03 of controller with device number 1: unused
	CX04	CY04 of controller with device number 1: unused
	CX05	CY01 of controller with device number 2 = process variable 1 (PV1) of Loop 2 controller
	CX06	CY02 of controller with device number 2 = setpoint value 1 (SV1) of Loop 2 controller
	CX07	CY03 of controller with device number 2 = Cas/Auto status (CAF1) of Loop 2 controller
	CX08	CY04 of controller with device number 2: unused
Reception timeout flag	CF01	Reception timeout flag used to determine whether communication from the controller with device number 1 is normal or abnormal
	CF02	Reception timeout flag used to determine whether communication from the controller with device number 2 is normal or abnormal

4.3 Examples of User Programs

The following are examples of programs for the system described at the beginning of this chapter.

1) Example of user program for Loop 1

```
    LD      CF02            ; reception timeout flag
    GIF     @ERR            ; jumps to @ERR in case of reception failure
; <<process when peer-to-peer communication data is normal>>
    LD      CX07            ; reads Cas/Auto status of Loop 2
    GIF     @LOOP-CAS       ; jumps to @LOOP-CAS when in Cas mode
    .........
    .........
    LD      CX06            ; reads SV1 of Loop 2
    ST      MV1             ;
@LOOP-CAS
    .........
; <<creation of transmitted data>>
    LD      PV1
    ST      CY01            ; writes process variable 1 (PV1) to transmission
                              register CY01
    LD      MV1             ; reception timeout flag
    ST      CY02            ; writes manipulated output variable (MV1) to transmission
                              register CY02
    .........
; <<process when peer-to-peer communication data is abnormal>>
@ERR
    .........
    .........
```

2) Example of user program for Loop 2

```
    LD      CF01            ; reception timeout flag
    GIF     @ERR            ; jumps to @ERR in case of reception failure
; <<process when peer-to-peer communication data is normal>>
    LD      CX02            ; reads manipulated output variable of Loop 1
    ST      CSV1            ; writes to cascade setpoint value
    .........
    .........
    .........
; <<creation of transmitted data>>
    LD      PV1
    ST      CY01            ; writes process variable 1 (PV1) to transmission register
                              CY01
    LD      SV1
    ST      CY02            ; writes setpoint value 1 (SV1) to transmission register
                              CY02
    LD      CAF1
    ST      CY02            ; writes Cas/Auto mode 1 (CAF1) transmission register
                              CY03
    .........
; <<process when peer-to-peer communication data is abnormal>>
@ERR
    .........
    .........
```

3) Example of user program for Loop 3

```
    LD      CX01        ; reads PV1 of Loop 1
    ST      T01         ; saves to temporary register 1
    LD      CF01        ; reads reception timeout flag 1
    GIF     @ERR        ; jumps to @ERR in case of abnormal input data
; <<process when peer-to-peer communication data of Loop 1  is normal>>
    LD      CX05        ; reads PV1 of Loop 2
    ST      T02         ; saves to temporary register 2
    LD      CF02        ; reads reception timeout flag 2
    GIF     @ERR        ; jumps to @ERR in case of abnormal input data
; <<process when peer-to-peer communication data of Loop 2  is normal>>
    .....
    .....
; <<performs correction computation, etc. using T01 and T02>>
    .....
    .....
; <<process when peer-to-peer communication data is abnormal>>
@ERR
    .....
    .....
```