CC-Link IE Field Network
Basic Sample Code User's Manual
(Master Station)
Version 1.01.4

Revisions

Date	No.	Revision
2016/08/01	Version 1.00	First edition
2016/10/07	Version 1.01	Corrected minor error
2016/12/27	Version 1.01.3	Porting for Linux
2017/03/21	Version 1.01.4	1. Adding the function for discarding request packet of another network on Raspbian(jessie). 2. Changing build sequence.

Contents

Relevant material

The following table lists the materials relevant to this manual.

No.	Publisher	Material name	Material number
1	CC-Link Partner Association (CLPA)	CC-Link IE Field Network Basic Specification (Application Layer Protocol)	BAP-C2010-004
2	CC-Link Partner Association (CLPA)	SLMP (Seamless Message Protocol) Specification (Overview)	BAP-C2006-001
3	CC-Link Partner Association (CLPA)	SLMP (Seamless Message Protocol) Specification (Services)	BAP-C2006-002
4	CC-Link Partner Association (CLPA)	SLMP (Seamless Message Protocol) Specification (Protocol)	BAP-C2006-003

Radix notation

The following radix notation is used in this manual except as otherwise specifically provided.

No.	Radix	Description	Example
1	Decimal	Units representing cardinal numbers are not added after numeric string.	0
2	Hexadecimal	The symbol 0x representing a hexadecimal is added before a numeric string.	0x00

Integral data types

The following integral data types is used in this manual except as otherwise specifically provided.

No.	Integral data type	Sign	Bit	Byte	C programming language (32 bits)
1	int8_t	Signed	8	1	char
2	int16_t	Signed	16	2	short
3	int32_t	Signed	32	4	int, long
4	int64_t	Signed	64	8	long long
5	uint8_t	Unsigned	8	1	unsigned char
6	uint16_t	Unsigned	16	2	unsigned short
7	uint32_t	Unsigned	32	4	unsigned int, unsigned long
8	uint64_t	Unsigned	64	8	unsigned long long

Precautions

The following lists the precautions about this manual.

- Since the attached sample codes are application examples, they do not guarantee the actual operation.
- This manual does not describe the terminology explanation of the CC-Link IE Field Network Basic and SLMP or troubleshooting. Acquire the relevant manuals from vendors of each product for reference, if necessary.
- Note that the descriptions of this manual and the sample code and specifications may be changed without notice.

1 Overview

This manual is for engineers to develop the CC-Link IE Field Network Basic master station application.

2 Terminology

This manual uses the following generic terms and abbreviations for descriptions except as otherwise specifically provided.

Generic term and abbreviation	Description
CCIEF-BASIC	An abbreviation for CC-Link IE Field Network Basic
SLMP	An abbreviation for Seamless Message Protocol
SLMP information	The structure including the following information relevant to the SLMP packet: Network number, node number, processor number, packet data length, command, subcommand, and pointer to data
Master station	An abbreviation for CC-Link IE Field Network Basic master station
Slave station	An abbreviation for CC-Link IE Field Network Basic slave station

3 Function

The sample code provides the following functions.

Table 1 Function of Sample Code

No.	Name	Description
1	Master station	Performs cyclic transmission with the CCIEF-BASICS slave station as a CCIEF-BASIC master station.

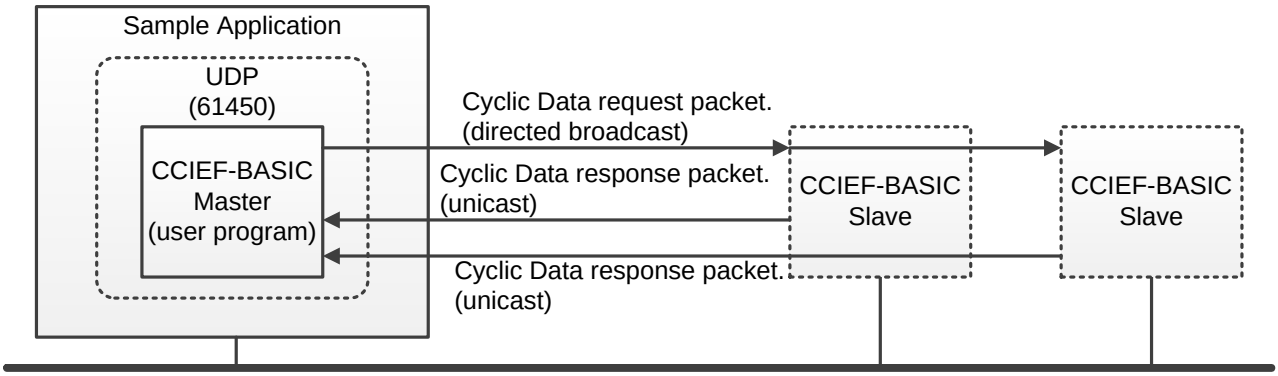

Figure 1 Function of Sample Code

4 Specifications

The following table lists the specifications of the CCIEF-BASIC master station in the sample code.

Table 2 Specifications of the CCIEF-Basic Master Station

Item		Description
Protocol		UDP
Port number		61450
IP address		IPv4 class C: Address range 192.0.0.1 to 223.255.255.254 Network address length: 24 bits, host address length: 8 bits (Subnet mask: 255.255.255.0)
Message format		SLMP with no serial number added (Binary mode)
Transmission format		Directed broadcast (Send), Unicast (Receive)
Number of connected modules		Master station: 1 station Slave station: 64 stations maximum (One station can occupy multiple number of stations)
Number of groups		64 groups maximum
Function		Cyclic transmission
Cyclic data	RX	4096 bits maximum (Sum total of all slave stations)
	RY	4096 bits maximum (Sum total of all slave stations)
	RWw	2048 words maximum (Sum total of all slave stations)
	RWr	2048 words maximum (Sum total of all slave stations)
Parameter		Text file of the CSV ("Comma-Separated Values") format. For details, refer to below.

The sample code defines the parameters of the master station as follows.
The parameters are described in the text file of the CSV ("Comma Separated Values") format.

Table 3 Parameter Specifications of Sample Code

Section	ID	Parameter	Setting range	Description
Group (Group setting)	1	Total number of group settings	1 to 64	Set the total number of groups.
	2	Group number	1 to 64	Set the group number.
	3	Timeout period	1 to 65535 [ms] (0: 500 [ms])	Specify the timeout period of the link scan.
	4	Number of consecutive timeouts until the detection of disconnection	1 to 65535 times (0: 3 times)	Specify the number of consecutive timeouts until the detection of disconnection.
	5	Constant link scan time	1 to 2000 [ms] (0: Constant link scan is not used)	Specify the constant link scan time.
	6 and later	Same items as ID 2 to 5 (variable number of items)	As described on the left	Set the same items as ID 2 to 5 for the number of group set.
Slave (Slave station setting)	1	Total number of slave station settings	1 to 64	Set the total number of the slave stations.
	2	IP address	192.0.0.1 to 223.255.255.254	Set the IP address of the slave station.
	3	Number of occupied stations	1 to 16	Set the number of occupied stations of the slave station.
	4	Group number	1 to 64	Set the group number of the slave station.
	5 and later	Same items as ID 2 to 4 (variable number of items)	As described on the left	Set the same items as ID 2 to 4 for the number of slave stations set.

The following figure shows an example of parameter setting.

[System configuration]

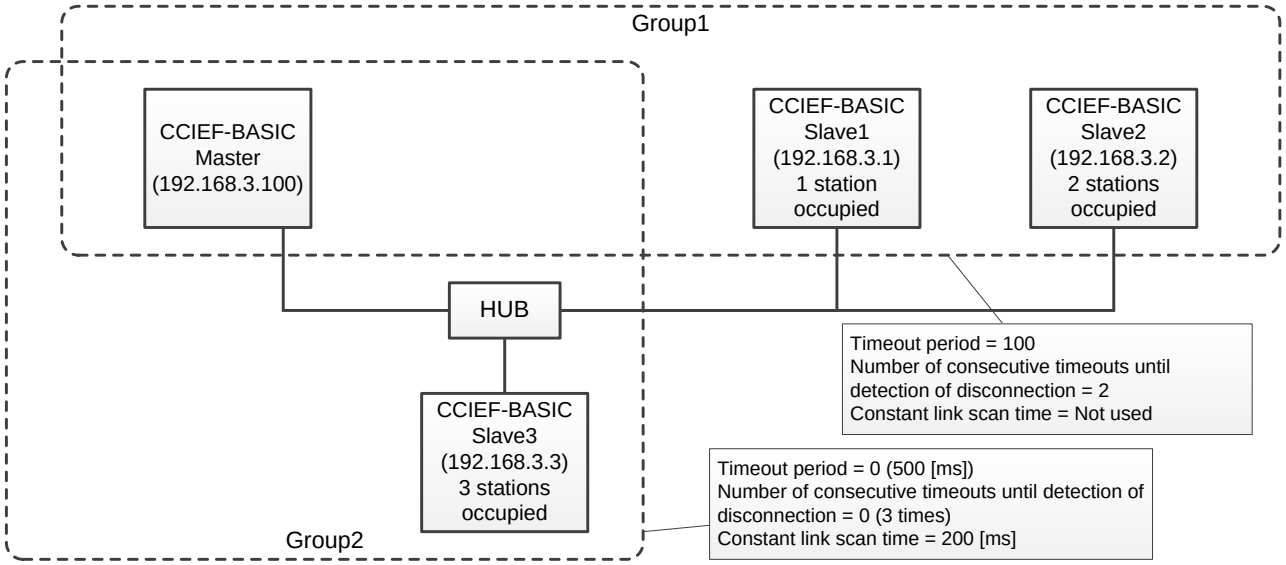

Figure 2 Example of system configuration

[MasterParameter.csv]

```
,,
CCIEF-BASIC Master Sample Parameter,,
,,
Group,,
ID,DATA,COMMENT
1,2,Total number of group
2,1,Number of group
3,100,Group1 Cyclic transmission timeout
4,2,Group1 Count of cyclic transmission timeout
5,0,Group1 Constant link scan time
6,2,Number of group
7,0,Group2 Cyclic transmission timeout
8,0,Group2 Count of cyclic transmission timeout
9,200,Group2 Constant link scan time
,,
Slave,,
ID,DATA,COMMENT
1,3,Total number of slave
2,192.168.3.1,Slave1 IP address
3,1,Slave1 Number of occupied stations
4,1,Slave1 Number of group
5,192.168.3.2,Slave2 IP address
6,2,Slave2 Number of occupied stations
7,1,Slave2 Number of group
8,192.168.3.3,Slave3 IP address
9,3,Slave3 Number of occupied stations
10,2,Slave3 Number of group
```

5 Application Development

5.1 Development environment

The sample codes attached in this manual cause no compile error when "VC++ 2010 (Visual Studio 2010 Visual C++)" is used. Refer to the procedure from compilation to execution of the sample code using VC++ 2010 described in Chapter 5.

5.2 Development procedure

This section describes the procedure to develop an application using the attached sample code.
The sample code is configured with the program parts listed in Table 4. Change the SLMP library according to the implementation environment. In addition, change the contents of the user programs according to the application.

Table 4 Configuration of the Sample Code

No.	Program part	Overview
1	SLMP library	This function generates the SLMP packet and acquires the SLMP information from the packet. Change the program according to the implementation environment.
2	User program	This application program implements functions of the device. Sample codes to execute cyclic transmissions as a CC-Link IE Field Network Basic master station using WinSock are described in this specification as an example. Change the program according to the environment.

The following describes the procedure to develop an application.

(1) Creating user program (CCIEF_BASIC_MASTER.c, CCIEF_BASIC_MASTER.h, CCIEF_BASIC_SLAVES.c, CCIEF_BASIC_SLAVES.h, SOCKET.c, SOCKET.h, TIMER.c, TIMER.h, USER_SAMPLE.c, USER_SAMPLE.h)
Create a user program. For details, refer to section 5.5.

(2) Creating SLMP library (SLMP.c, SLMP.h)
After compiling the source code of the SLMP library included in the attached sample code, execute the librarian to create a library file.

(3) Linking user programs and library file
Link user programs and library file to create a load module file.

5.3 List of the sample code files

The following shows the directory configuration of the sample code.

```
root.    +- library    +- include    ... SLMP library header file
         |             +- src        ... SLMP library code file
         |
         +- sample     +- include    ... User program header file
                       +- src        ... User program code file
```

The following table lists the sample code files.

No.	Folder name			File name	Description
1	Root			version.txt	Version information
2				readme.txt	Help file
3		library	include	SLMP.h	SLMP library header
4			src	SLMP.c	SLMP library function
5		sample	include	CCIEF_BASIC_MASTER.h	User program header (Master station)
6				CCIEF_BASIC_SLAVES.h	User program header (Slave station status)
7				SOCKET.h	User program header (Socket)
8				TIMER.h	User program header (Timer)
9				USER_SAMPLE.h	User program header
10			src	CCIEF_BASIC_MASTER.c	User program (Master station)
11				CCIEF_BASIC_SLAVES.c	User program (Slave station status)
12				SOCKET.c	User program (Socket)
13				TIMER.c	User program (Timer)
14				USER_SAMPLE.c	User program

5.4 List of sample code functions

Table 5 lists the functions included in the sample code.

Table 5 List of Sample Code Functions

No.	Program part	File	Function name	Function type	Overview	Disclosed/ undisclosed
1	SLMP library	SLMP.c	SLMP_MakePacketStream	int	SLMP packet generation	Disclosed
2			SLMP_GetSlmpInfo	int	SLMP information acquisition	Disclosed
3			local_itoa	uint8_t	Conversion from numeric string to ASCII	Disclosed
4			local_atoi	uint8_t	Conversion from ASCII to numeric string	Disclosed
5			SLMP_MakeErrorData	int	SLMP error response data generation	Disclosed
6	User program	CCIEF_ BASIC_ MASTER .c	ccief_basic_master_initialize	int	CCIEF-BASIC master station initialization	Disclosed
7			ccief_basic_master_terminate	void	CCIEF-BASIC master station termination	Disclosed
8			ccief_basic_master_main	int	CCIEF-BASIC master station main processing	Disclosed
9			ccief_basic_master_start_ cyclic	int	Cyclic transmission start	Disclosed
10			ccief_basic_master_stop_ cyclic	int	Cyclic transmission stop	Disclosed
11			ccief_basic_master_get_rx	int	RX data acquisition	Disclosed
12			ccief_basic_master_set_ry	int	RY data setting	Disclosed
13			ccief_basic_master_set_rww	int	RWw data setting	Disclosed
14			ccief_basic_master_get_rwr	int	RWr data acquisition	Disclosed
15			ccief_basic_master_get_ pointer	uint16_t *	Device head pointer acquisition	Disclosed
16			ccief_basic_master_set_unit_ info	void	Own station unit information setting	Disclosed
17			ccief_basic_master_get_ slave_info	int	Slave station receive information acquisition	Disclosed
18			ccief_basic_master_get_ group_info	int	Group information acquisition	Disclosed
19			ccief_basic_master_check_ parameter	int	Parameter check	Undisclosed
20			ccief_basic_master_recv	Int	Packet receiving	Undisclosed
21			ccief_basic_master_polling	Int	Regular execution	Undisclosed
22			ccief_basic_master_execute_ state	void	Master station status execution	Undisclosed
23			ccief_basic_master_execute_ state_wait_cyclic	void	Master station status execution (On standby)	Undisclosed

No.	Program part	File	Function name	Function type	Overview	Disclosed/ undisclosed
24	User program	CCIEF_ BASIC_ MASTER .c	ccief_basic_master_execute_ state_persuasion	void	Master station status execution (Master station arbitration being performed)	Undisclosed
25			ccief_basic_master_execute_ state_linkscan_end	void	Master station status execution (Link scan completed)	Undisclosed
26			ccief_basic_master_execute_ state_linkscan	void	Master station status execution (Link scan being performed)	Undisclosed
27			ccief_basic_master_ persuasion_timer_timeout	void	Timeout of frame monitoring time (callback function)	Undisclosed
28			ccief_basic_master_cyclic_ timer_timeout	void	Timeout of cyclic transmission (callback function)	Undisclosed
29			ccief_basic_master_make_ cyclic_data	int	Cyclic transmission data generation	Undisclosed
30			ccief_basic_master_send_ cyclic_data	int	Cyclic transmission data sending	Undisclosed
31			ccief_basic_master_recv_ cyclic_data_response	void	Cyclic transmission data response receiving	Undisclosed
32		CCIEF_ BASIC_ SLAVES. c	ccief_basic_slaves_initialize	void	Slave station status initialization	Disclosed
33			ccief_basic_slaves_execute_ state	void	Slave station status execution	Disclosed
34			ccief_basic_slaves_execute_ state_disconnect	void	Slave station status execution (Disconnected)	Undisclosed
35			ccief_basic_slaves_execute_ state_connecting	void	Slave station status execution (Waiting for return)	Undisclosed
36			ccief_basic_slaves_execute_ state_cyclic_stop	void	Slave station status execution (Cyclic transmission stopped)	Undisclosed
37			ccief_basic_slaves_execute_ state_cyclic_end	void	Slave station status execution (Cyclic transmission completed)	Undisclosed
38			ccief_basic_slaves_execute_ state_cyclic	void	Slave station status execution (Cyclic transmission being performed)	Undisclosed
39		SOCKET. c	socket_initialize	int	Socket initialization	Disclosed
40			socket_terminate	void	Socket termination	Disclosed
41			socket_recv	int	Packet receiving	Disclosed
42			socket_send	int	Packet sending	Disclosed

No.	Program part	File	Function name	Function type	Overview	Disclosed/ undisclosed
43	User program	TIMER.c	timer_initialize	void	Timer initialization	Disclosed
44			timer_terminate	void	Timer termination	Disclosed
45			timer_main	void	Timer main processing	Disclosed
46			timer_start	int	Timer start	Disclosed
47			timer_stop	void	Timer stop	Disclosed
48			timer_get_time	int64_t	Current time acquisition	Disclosed
49			timer_calculate_time_data	int64_t	Clock information calculation	Disclosed
50			timer_analyze_time_data	void	Clock time analysis	Disclosed
51			timer_gettimeofday	int	System time acquisition	Disclosed
52		USER_ SAMPLE .c	main	void	Main processing	Disclosed
53			user_callback_cyclic_link_ scan_end	void	Link scan completed (callback function)	Disclosed
54			user_parameter_file_read	int	Parameter file reading	Undisclosed
55			user_get_input_line	void	Input character string acquisition	Undisclosed
56			user_show_menu_top	void	Menu screen display	Undisclosed
57			user_input_check	int	Input check	Undisclosed
58			user_start_cyclic	void	Cyclic transmission start	Undisclosed
59			user_stop_cyclic	void	Cyclic transmission stop	Undisclosed
60			user_start_application	void	Application start	Undisclosed
61			user_stop_application	void	Application stop	Undisclosed
62			user_show_slave_info	void	Slave station information display	Undisclosed
63			user_show_master_info	void	Master station information display	Undisclosed
64			user_show_parameter	void	Parameter display	Undisclosed
65			user_get_adapter_info	int	Network adapter information acquisition	Undisclosed

Disclosed: The functions to be disclosed outside. Undisclosed: The functions to be used in the local file.

5.5 Creating a user program

Create a user program according to the implementation environment.
The following table lists the user program files.

Table 6 List of User Program Files

No.	File name	Description
1	CCIEF_BASIC_MASTER.c	Executes cyclic transmission.
2	CCIEF_BASIC_SLAVES.c	Executes the processing under the individual states for each slave station.
3	SOCKET.c	Provides a set of functions to execute socket processing.
4	TIMER.c	Provides the library to execute timer processing.
5	USER_SAMPLE.c	Executes initialization and main processing of the master station, and reads parameter files. This program also implements callback functions and executes link scan completion processing for cyclic transmission.

The following shows the points requiring changes in the program due to differences in the operating system and protocol stack in the implementation environment.

Table 7 Points Requiring Changes in the Program due to Differences in the Operating System and Protocol Stack

No.	File name	Function name	Points to be changed
1	CCIEF_BASIC_MASTER.c	ccief_basic_master_initialize	Method to implement the socket function and structure
2		ccief_basic_master_check_parameter	Method to implement the socket function and structure*1
3		ccief_basic_master_recv	Method to implement the socket function and structure
4		ccief_basic_master_execute_state_wait_cyclic	Method to implement the socket function and structure
5	SOCKET.c	socket_initialize	Method to implement the socket function and structure*1
6		socket_terminate	Method to implement the socket function and structure
7		socket_recv	Method to implement the socket function and structure
8		socket_send	Method to implement the socket function and structure
9	TIMER.c	timer_calculate_time_data	Method to acquire the system time
10		timer_gettimeofday	Method to acquire the system time
11	USER_SAMPLE.c	main	Method to implement the socket function and structure
12		user_show_parameter	Method to implement the socket function and structure
13		user_get_adapter_info	Method to acquire the network adapter information

*1 Non blocking mode setting is required.

The following describes the key points in creating a user program in every file.

(1) CCIEF_BASIC_MASTER.c

This program executes cyclic transmission with the slave station. This program also processes the group status of the master station.

* For details, refer to the CC-Link IE Field Network Specification (CC-Link IE Field Network Basic Specification-Application Layer Protocol).

(a) Cyclic data

The sample code defines the cyclic data (RX, RY, RWw, RWr) internally. The user program realizes cyclic transmission with the master station by accessing the cyclic data at an arbitrary timing. As cyclic data access methods, a method to directly access cyclic data devices and a method to acquire the head pointer of each device and access with the pointer are defined.

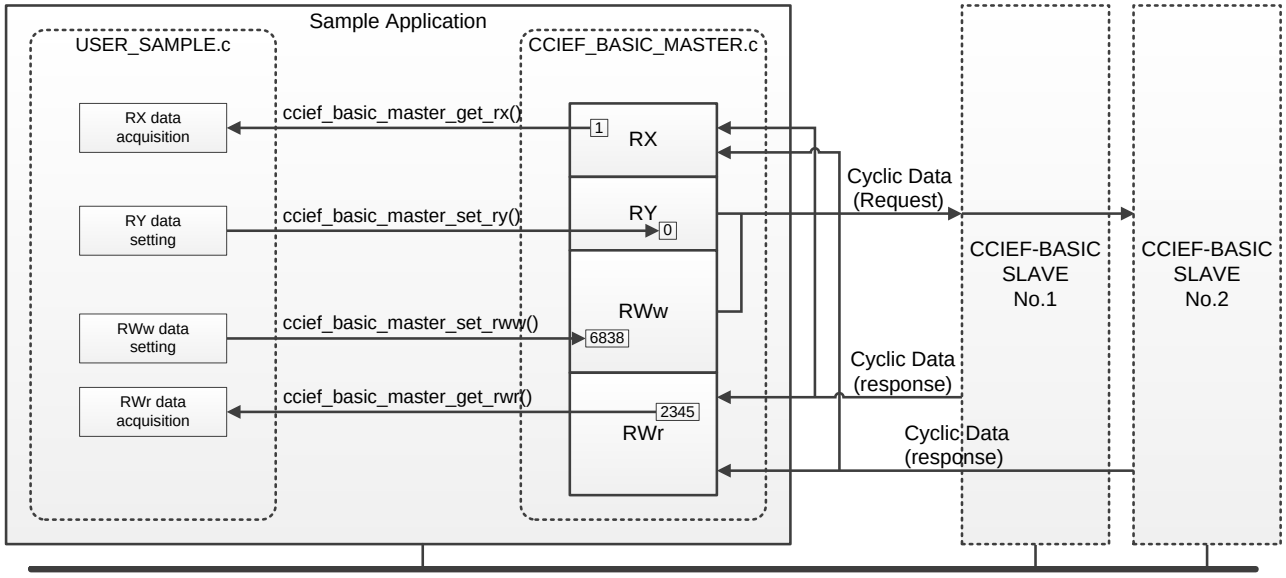

Figure 3 Method to Directly Access Cyclic Data Devices

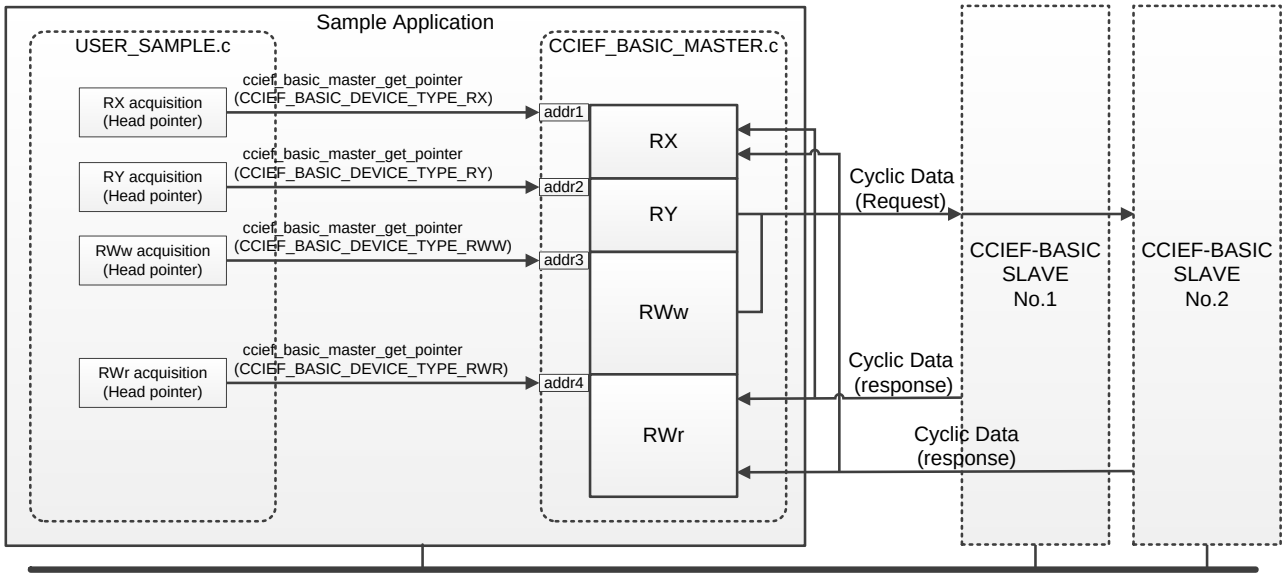

Figure 4 Method to Acquire the Head Pointer of Each Device of the Cyclic Data

(b) Callback function

The sample code defines the callback functions and executes the specified callback function in the following timing.
The user can easily develop a function by implementing the callback functions.

Table 8 List of Callback Functions

No.	Callback function	Execution timing
1	CCIEF_BASIC_MASTER_CALLBACK _CYCLIC_LINK_SCAN_END	This function is executed when the status of the master station transits to "Link scan completed".

[CCIEF_BASIC_MASTER.h]

```
typedef void(*CCIEF_BASIC_MASTER_CALLBACK_CYCLIC_LINK_SCAN_END)(uint8_t ucGroupNumber);
```

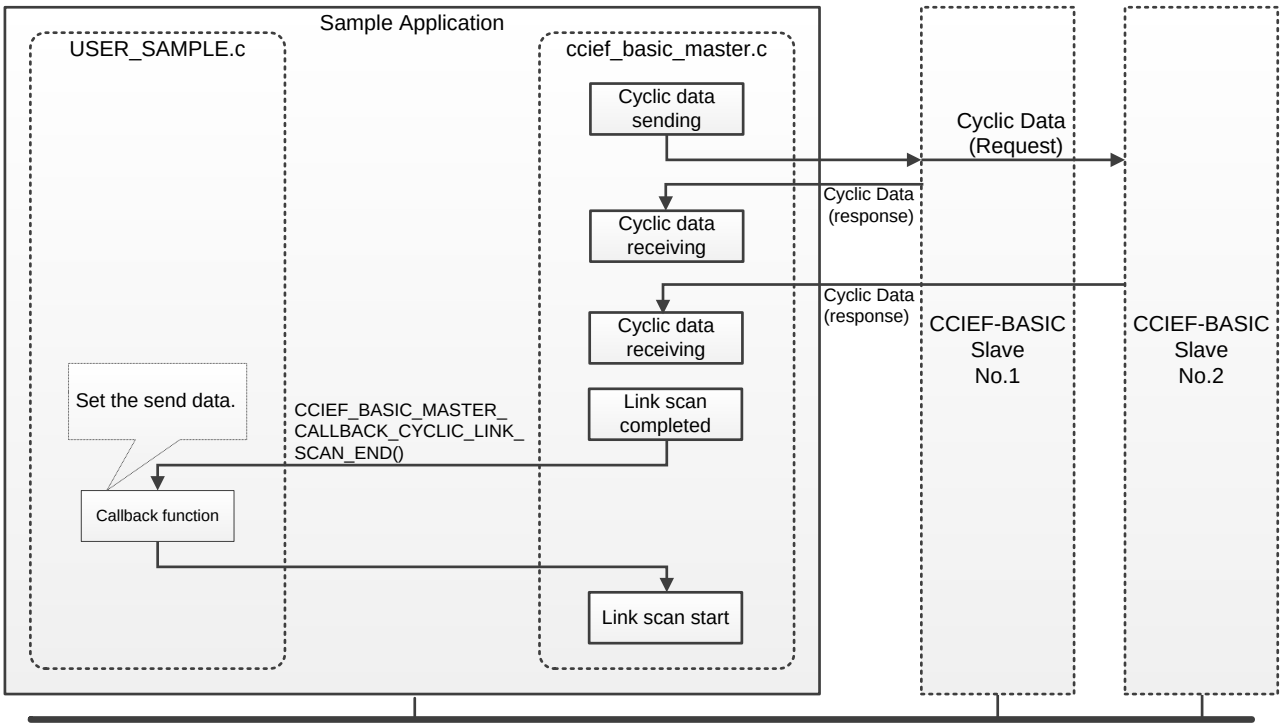

Figure 5 Image of the Callback Function

(c) Own station unit information

The sample code can set the own station uni information to notice to the slave station by request data of the cyclic transmission in the user program.

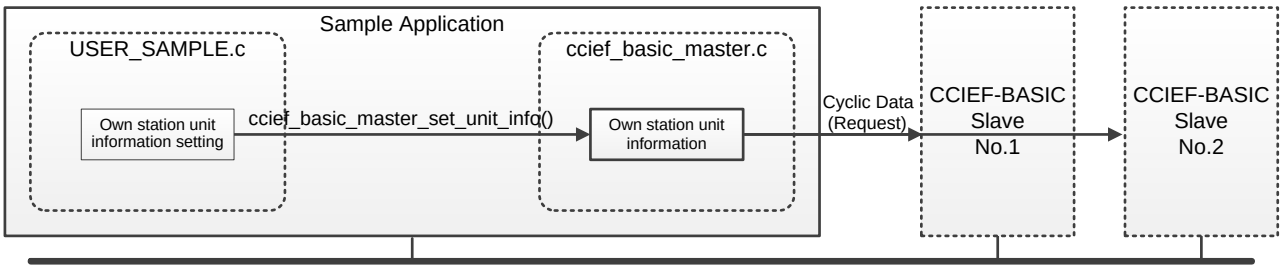

Figure 6 Setting the Own Station Unit Information

(d) Slave station receive information acquisition

The sample code can get the information of each slave station received via the cyclic data by the user program.

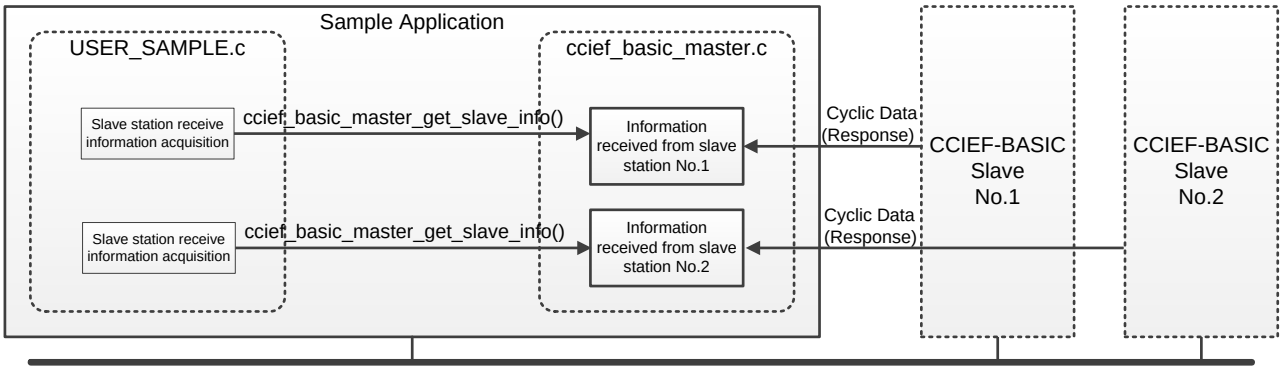

Figure 7 Acquiring the Information Received from the Slave Station

(e) Group information acquisition

The sample code can get the information of each group of the master station by the user program.

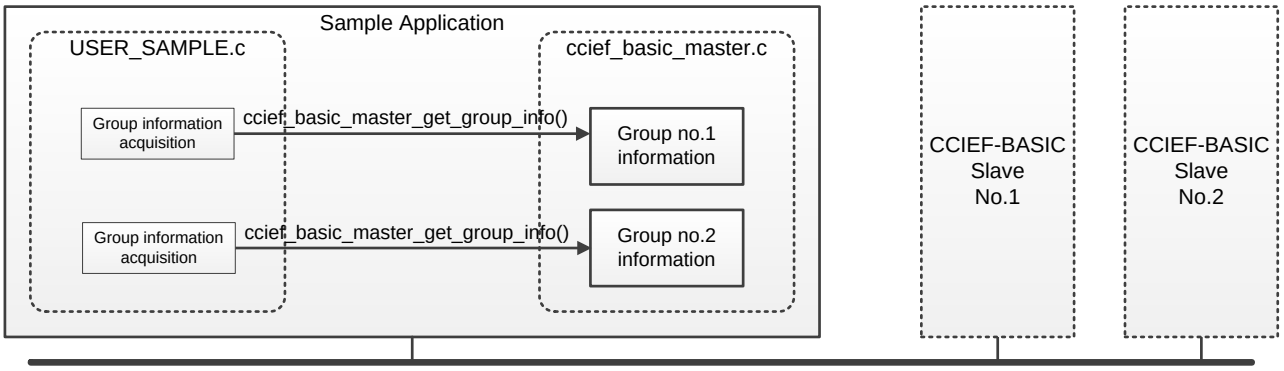

Figure 8 Acquiring Group Information

(2) CCIEF_BASIC_SLAVES.c

This program executes the processing under the individual states for each slave station.
* For details, refer to the CC-Link IE Field Network Specification (CC-Link IE Field Network Basic Specification-Application Layer Protocol).

(3) SOCKET.c

This program provides a set of functions to execute socket processing.
* Change the program according to the implementation environment.

(4) TIMER.c

This program provides the library to execute timer processing.
* Change the method to acquire the elapsed time (processor time) or others according to the implementation environment.

(a) Callback function

The sample code defines the following callback functions. The callback function is executed when the registered timer has timed out.

[TIMER.h]

```
typedef void (*TIMER_CALLBACK)( int iId, void *pCallbackArg );
```

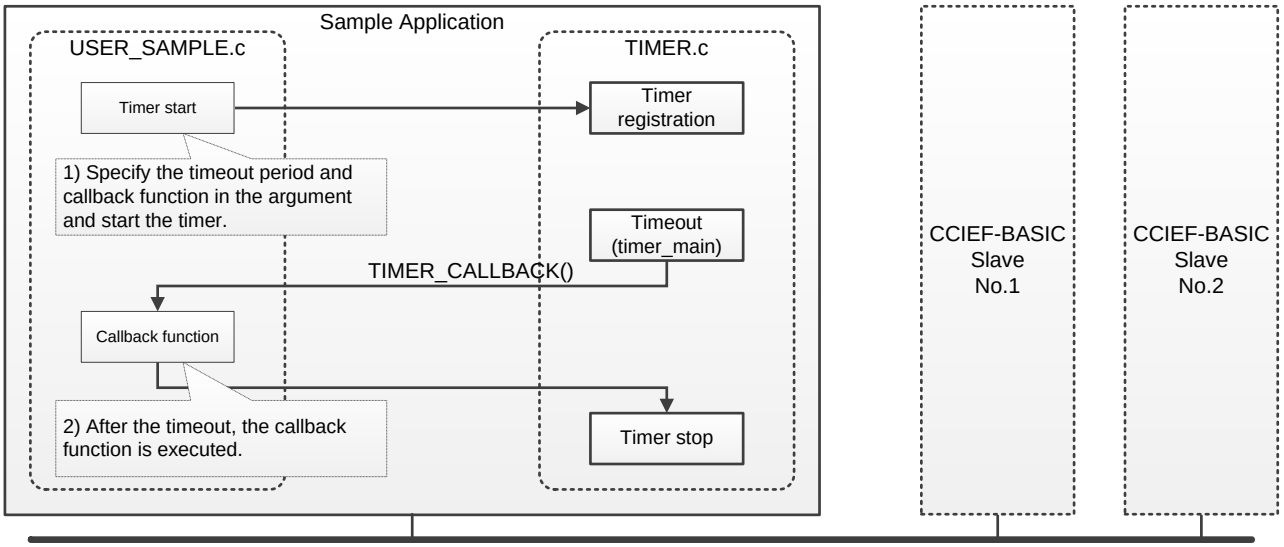

Figure 9 Image of the Callback Function

(b) Flowchart

The following figures show the flowchart of the sample program.

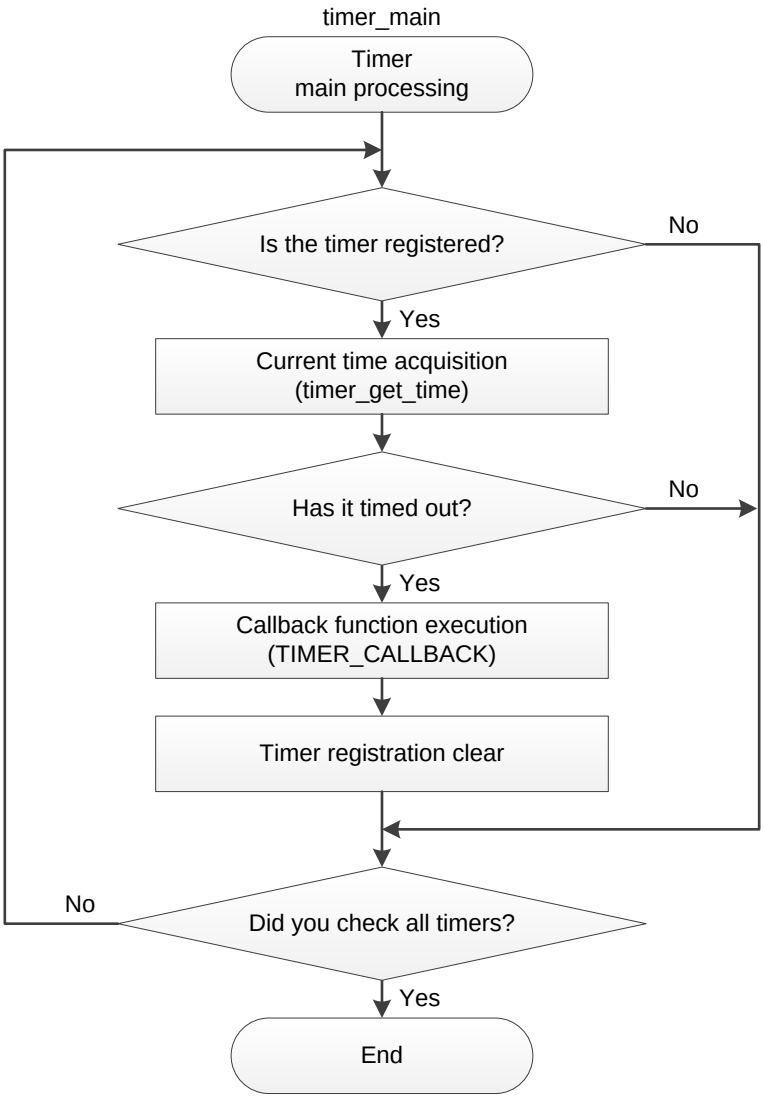

Figure 10 Flowchart for TIMER.c (Example of the Sample Program)

(5) USER_SAMPLE.c

This program initializes the master station and timer function, performs main processing, and reads parameter files. This program also implements callback functions and executes link scan completion processing for cyclic transmission.
By user operation, this program starts and stops the cyclic transmission and displays various information.

(a) Implementing a program

The sample program implements the callback functions provided by CCIEF_BASIC_MASTER.c.
The following table lists the implementation content of the sample program.

Table 9 Implementation Content of the Sample Program

No.	Program	Implementation content	Callback function of the implementation source
1	user_callback_cyclic_link_scan_end	Sets the send data to RY and RWw of the slave station belonging to the specified group number.*1	CCIEF_BASIC_MASTER_CALLBACK_CYCLIC_LINK_SCAN_END (Refer to Table 8.)

*1 Change the program according to the implementation environment.

(b) User operation

After execution of the application, the sample program can execute the following processing by user operation (key input).

Table 10 User Operation

No.	User operation	Execution description
1	'1' key	Starts cyclic of the slave station.*1
2	'2' key	Stops cyclic of the slave station.*1
3	'3' key	This function starts application. (Sets the data in the cyclic device of the master station (RY, RWw))
4	'4' key	Stops application. (Clears the cyclic device of the master station (RY, RWw))
5	'5' key	Displays the slave station information on the screen.*1
6	'6' key	Displays the master station information on the screen.*2
7	'7' key	Displays the content of the parameter setting on the screen.
8	'Esc' key	Terminates application.

*1 All slave stations are targeted.
*2 All groups are targeted.

(c) Flowchart

The following figures show the flowchart of the sample program.

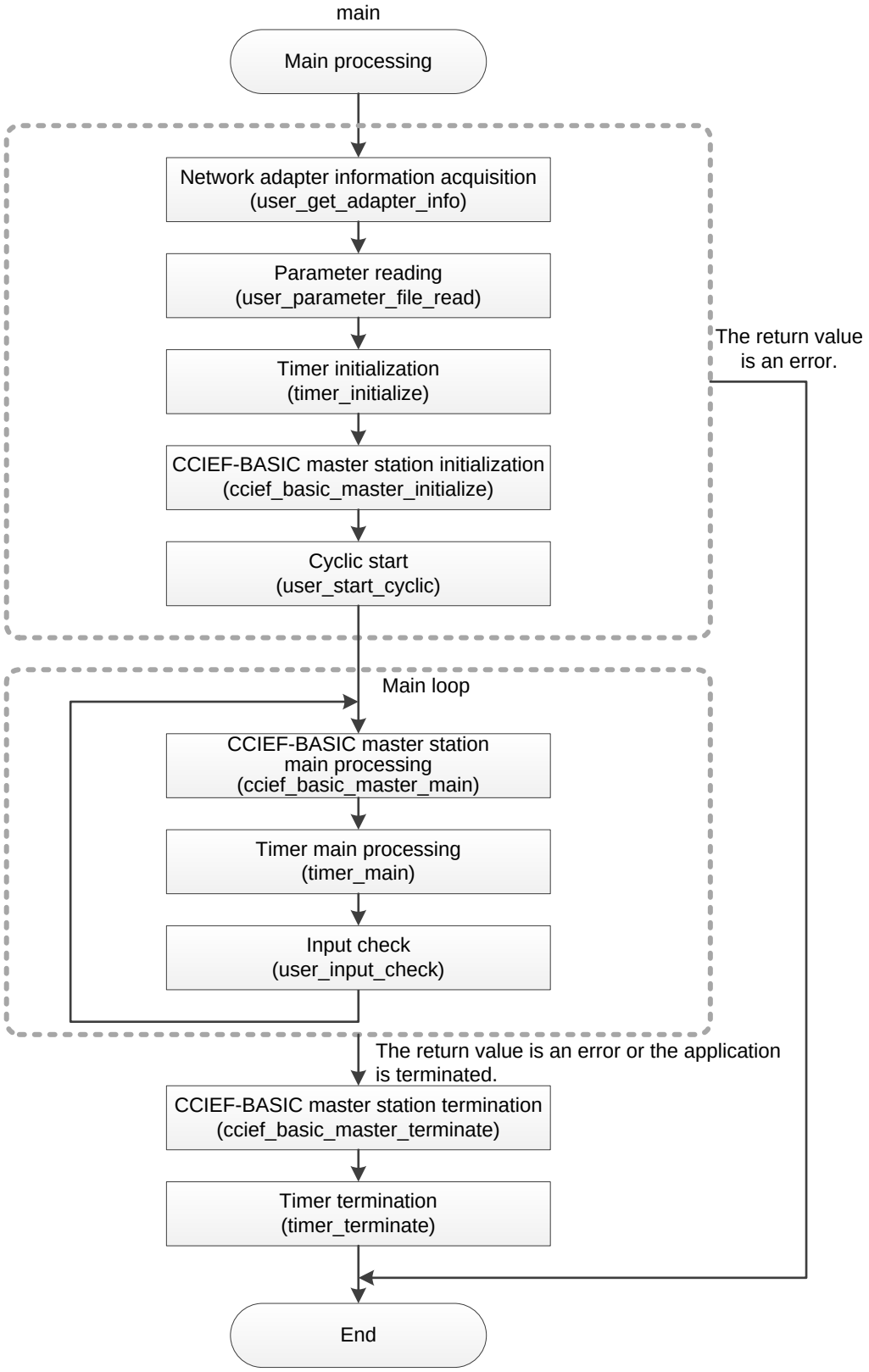

Figure 11 Flowchart for USER_SAMPLE.c (Example of the Sample Program)

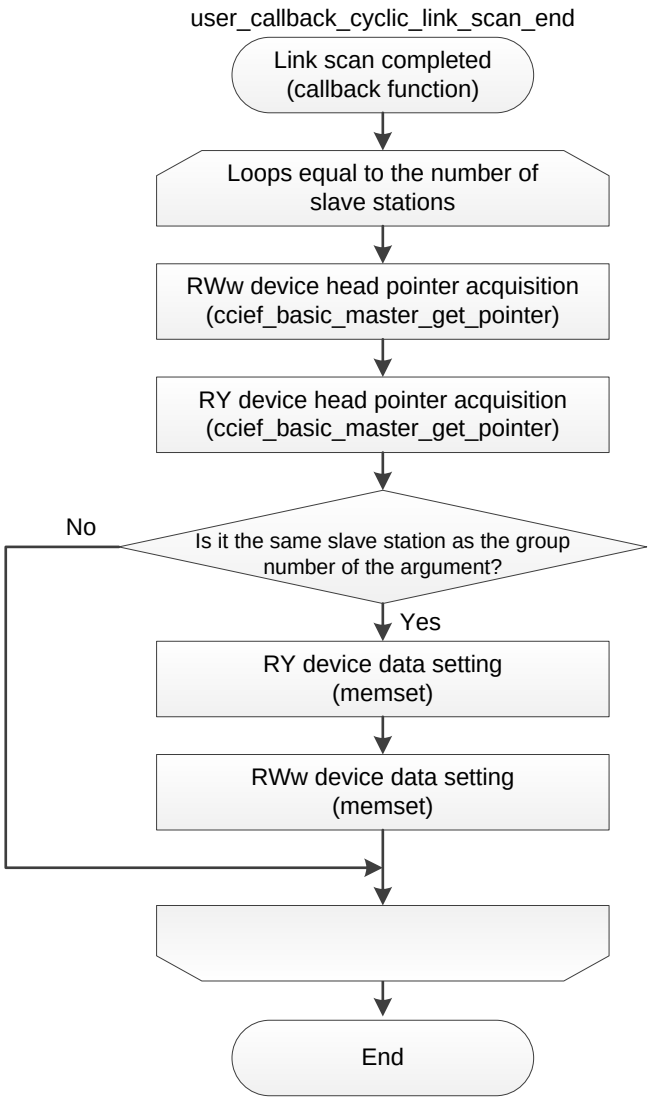

Figure 12 Flowchart 2 for USER_SAMPLE.c (Example of the Sample Program)

5.6 Function details

5.6.1 Definition of the return value

The following codes are used as error codes and end codes returned as function return values in the SLMP library.

[SLMP.h]

```
#define SLMP_ERR_OK                                        0
#define SLMP_ERR_NG                                        (-1)
#define SLMP_ERR_COMMAND_SUBCOMMAND                        (0xC059)
#define SLMP_ERR_WRONG_DATA                                (0xC05C)
#define SLMP_ERR_DATA_LENGTH                               (0xC061)
#define SLMP_ERR_UNDER_EXECUTION                           (0xCEE0)
#define SLMP_ERR_REQ_DATA_SIZE                             (0xCEE1)
#define SLMP_ERR_RES_DATA_SIZE                             (0xCEE2)
#define SLMP_ERR_NO_EXIST_SERVER_NO                        (0xCF10)
#define SLMP_ERR_CAN_NOT_COMMUNICATION_SETTING             (0xCF20)
#define SLMP_ERR_NO_EXIST_PARAM_ID                         (0xCF30)
#define SLMP_ERR_CAN_NOT_PARAMETER_SET                     (0xCF31)
#define SLMP_END_DUPLICATE_MASTER                          (0xCFE0)
#define SLMP_END_INVALID_NUMBER_OF_OCCUPIED_STATIONS       (0xCFE1)
#define SLMP_END_SLAVE                                     (0xCFF0)
#define SLMP_END_DISCONNECTED_REQUEST                      (0xCFFF)
```

The following codes are used as error codes returned as function return values in the user program.

[CCIEF_BASIC_MASTER.h]

```
#define CCIEF_BASIC_MASTER_ERR_OK                          0
#define CCIEF_BASIC_MASTER_ERR_NG                          (-1)
#define CCIEF_BASIC_MASTER_ERR_DEVICE_RANGE                (-100)
#define CCIEF_BASIC_MASTER_ERR_MASTER_DUPLICATION          (-200)
#define CCIEF_BASIC_MASTER_ERR_SLAVE_DUPLICATION           (-300)
```

[SOCKET.h]

```
#define SOCKET_ERR_OK                                      0
#define SOCKET_ERR_SOCKET                                  (-100)
#define SOCKET_ERR_RECV                                    (-103)
#define SOCKET_ERR_SEND                                    (-104)
#define SOCKET_ERR_NO_RECEIVABLE                           (-200)
```

[TIMER.h]

```
#define TIMER_OK                                           0
#define TIMER_RESOURCE_NONE                                (-1)
```

[USER_SAMPLE.h]

```
#define USER_ERR_OK                                        0
#define USER_ERR_NG                                        (-1)
#define USER_EXIT                                          1
```

5.6.2 SLMP_MakePacketStream

Table 11 SLMP_MakePacketStream

Function	SLMP packet generation			
File name	SLMP.c		Disclosed/undisclosed	Disclosed
Call format	int SLMP_MakePacketStream (uint32_t ulFrameType, const SLMP_INFO *p, uint8_t *pucStream)			
Argument	Type	Variable name	Description	I/O
	uint32_t	ulFrameType	Frame type	Input
	const SLMP_INFO *	p	SLMP information	Input
	unsigned unit8_t *	pucStream	Send packet	Output
Return value	SLMP_ERR_OK: Normal SLMP_ERR_NG: Error			
Description	This function generates an SLMP communication packet.			

The following shows the configuration of SLMP_INFO based on the sample code.

[SLMP.h]

```
typedef struct
{
        uint32_t    ulFrameType;            /* Frame Type */
        uint16_t    usSerialNumber;         /* Serial Number */
        uint16_t    usNetNumber;            /* Network Number */
        uint16_t    usNodeNumber;           /* Node Number */
        uint16_t    usProcNumber;           /* Processor Number */
        uint16_t    usDataLength;           /* Data Length */
        uint16_t    usTimer;                /* Timer Value */
        uint16_t    usCommand;              /* Command */
        uint16_t    usSubCommand;           /* Sub Command */
        uint16_t    usEndCode;              /* End Code */
        uint8_t     *pucData;               /* Data */
}SLMP_INFO;
```

5.6.3 SLMP_GetSlmpInfo

Table 12 SLMP_GetSlmpInfo

Function	SLMP information acquisition			
File name	SLMP.c		Disclosed/undisclosed	Disclosed
Call format	int SLMP_GetSlmpInfo (SLMP_INFO *p, const uint8_t *pucStream)			
Argument	Type	Variable name	Description	I/O
	SLMP_INFO *	p	SLMP information	Output
	uint8_t *	pucStream	Receive packet	Input
Return value	SLMP_ERR_OK: Normal SLMP_ERR_NG: Error			
Description	This function acquires SLMP information.			

5.6.4 local_itoa

Table 13 local_itoa

Function	Conversion from numeric string to ASCII			
File name	SLMP.c		Disclosed/undisclosed	Disclosed
Call format	uint8_t local_itoa (uint8_t ucInt)			
Argument	Type	Variable name	Description	I/O
	uint8_t	ucInt	Numeric string	Input
Return value	uint8_t: ASCII code			
Description	This function converts numeric string to ASCII.			

5.6.5 local_atoi

Table 14 local_atoi

Function	Conversion from ASCII to numeric string			
File name	SLMP.c		Disclosed/undisclosed	Disclosed
Call format	uint8_t local_atoi (uint8_t ucInt)			
Argument	Type	Variable name	Description	I/O
	uint8_t	ucInt	Numeric string	Input
Return value	uint8_t: Numeric string			
Description	This function converts ASCII to numeric string.			

5.6.6 SLMP_MakeErrorData

Table 15 SLMP_MakeErrorData

Function	SLMP error response data generation			
File name	SLMP.c		Disclosed/undisclosed	Disclosed
Call format	int SLMP_MakeErrorData (const SLMP_INFO *p, uint8_t *pucStream, uint16_t *pusDataSize)			
Argument	Type	Variable name	Description	I/O
	SLMP_INFO *	p	SLMP information	Input
	uint8_t *	pucStream	Response data	Output
	uint16_t *	pusDataSize	Response data size	Output
Return value	SLMP_ERR_OK: Normal SLMP_ERR_NG: Error			
Description	This function generates SLMP error response data.			

5.6.7 ccief_basic_master_initialize

Table 16 ccief_basic_master_initialize

Function	CCIEF-BASIC master station initialization			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_initialize (uint32_t ulIpAddress, uint32_t ulSubnetMask, CCIEF_BASIC_MASTER_PARAMETER *pParameter, CCIEF_BASIC_MASTER_CALLBACK_CYCLIC_LINK_SCAN_END pCyclicLinkScanEndFunc)			
Argument	Type	Variable name	Description	I/O
	uint32_t	ulIpAddress	Master station IP address	Input
	uint32_t	ulSubnetMask	Master station subnet mask	Input
	CCIEF_BASIC_MASTER_ PARAMETER *	pParameter	Master station parameter	Input
	CCIEF_BASIC_MASTER_ CALLBACK_CYCLIC_LINK_ SCAN_END	pCyclicLinkScanEndFunc	Callback function (Link scan completed)	Input
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_SOCKET: Socket error			
Description	This function: - initializes master station; - initializes each variables; and - generates a socket.			

The following shows the configuration of CCIEF_BASIC_MASTER_PARAMETER based on the sample code.

[CCIEF_BASIC_MASTER.h]

```
typedef struct
{
        uint8_t     ucGroupNumber;                  /* Group number */
        uint16_t    usCyclicTransmissionTimeout;    /* Cyclic transmission timeout */
        uint16_t    usCyclicTransmissionTimeoutCount;  /* Count of cyclic transmission timeout */
        uint16_t    usConstantLinkScanTime;         /* Constant link scan time */
} CCIEF_BASIC_GROUP_PARAMETER;

typedef struct
{
        uint32_t    ulIpAddress;                    /* Slave ip address */
        uint16_t    usOccupiedStationNumber;        /* Number of occupied stations */
        uint8_t     ucGroupNumber;                  /* Group number */
} CCIEF_BASIC_SLAVE_PARAMETER;

typedef struct
{
        int                                 iTotalGroupNumber;          /* Total number of the groups */
        CCIEF_BASIC_GROUP_PARAMETER         Group[CCIEF_BASIC_MAX_GROUP_NUMBER];
                                                                        /* Parameter of the groups */
        int                                 iTotalSlaveNumber;          /* Total number of the slaves */
        CCIEF_BASIC_SLAVE_PARAMETER         Slave[CCIEF_BASIC_MAX_SLAVE_NUMBER];
                                                                        /* Parameter of the slaves */
} CCIEF_BASIC_MASTER_PARAMETER;
```

5.6.8 ccief_basic_master_terminate

Table 17 ccief_basic_master_terminate

Function	CCIEF-BASIC master station termination			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	void ccief_basic_master_terminate (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function: - terminates the master station; and - closes the socket.			

5.6.9 ccief_basic_master_main

Table 18 ccief_basic_master_main

Function	CCIEF-BASIC master station main processing			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_main (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal SOCKET_ERR_RECV: Socket error			
Description	This function: - receives a packet; and - executes regular processing.			

5.6.10 ccief_basic_master_start_cyclic

Table 19 ccief_basic_master_start_cyclic

Function	Cyclic transmission start			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_start_cyclic (int iSlaveNumber)			
Argument	Type	Variable name	Description	I/O
	int	iSlaveNumber	Slave station number	Input
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_NG: Error			
Description	This function starts cyclic transmission of the slave station specified by the argument iSlaveNumber.			

5.6.11 ccief_basic_master_stop_cyclic

Table 20 ccief_basic_master_stop_cyclic

Function	Cyclic transmission stop			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_stop_cyclic (int iSlaveNumber)			
Argument	Type	Variable name	Description	I/O
	int	iSlaveNumber	Slave station number	Input
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_NG: Error			
Description	This function stops cyclic transmission of the slave station specified by the argument iSlaveNumber.			

5.6.12 ccief_basic_master_get_rx

Table 21 ccief_basic_master_get_rx

Function	RX data acquisition			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_get_rx (int iNumber, int *piValue)			
Argument	Type	Variable name	Description	I/O
	int	iNumber	Device number	Input
	int *	piValue	Data storage location pointer Stored value: 0 (bit off) 1 (bit on)	Output
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_DEVICE_RANGE: Device range error			
Description	This function acquires RX data of the device number specified by the argument iNumber.			

5.6.13 ccief_basic_master_set_ry

Table 22 ccief_basic_master_set_ry

Function	RY data setting			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_set_ry (int iNumber, int iValue)			
Argument	Type	Variable name	Description	I/O
	int	iNumber	Device number	Input
	int	iValue	Set value 0 (bit off) 1 (bit on)	Input
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_NG: Error CCIEF_BASIC_MASTER_ERR_DEVICE_RANGE: Device range error			
Description	This function sets an argument iValue in the RY of the device number specified by the argument iNumber.			

5.6.14 ccief_basic_master_set_rww

Table 23 ccief_basic_master_set_rww

Function	RWw data setting			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_set_rww (int iNumber, uint16_t usValue)			
Argument	Type	Variable name	Description	I/O
	int	iNumber	Device number	Input
	uint16_t	usValue	Set value	Input
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_DEVICE_RANGE: Device range error			
Description	This function sets an argument iValue in the RWw of the device number specified by the argument iNumber.			

5.6.15 ccief_basic_master_get_rwr

Table 24 ccief_basic_master_get_rwr

Function	RWr data acquisition			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_get_rwr (int iNumber, uint16_t *pusValue)			
Argument	Type	Variable name	Description	I/O
	int	iNumber	Device number	Input
	uint16_t *	pusValue	Data storage location pointer	Output
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_DEVICE_RANGE: Device range error			
Description	This function acquires RWr data of the device number specified by the argument iNumber.			

5.6.16 ccief_basic_master_get_pointer

Table 25 ccief_basic_master_get_pointer

Function	Device head pointer acquisition			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	uint16_t *ccief_basic_master_get_pointer (int iDeviceType)			
Argument	Type	Variable name	Description	I/O
	int	iDeviceType	Device type	Input
Return value	Device head pointer			
Description	This function acquires a head pointer of the device.			

The following shows the definition of the device types based on the sample code.

[CCIEF_BASIC_MASTER.h]

```
#define CCIEF_BASIC_DEVICE_TYPE_RX      1      /* Type of device for RX */
#define CCIEF_BASIC_DEVICE_TYPE_RY      2      /* Type of device for RY */
#define CCIEF_BASIC_DEVICE_TYPE_RWW     3      /* Type of device for RWw */
#define CCIEF_BASIC_DEVICE_TYPE_RWR     4      /* Type of device for RWr */
```

5.6.17 ccief_basic_master_set_unit_info

Table 26 ccief_basic_master_set_unit_info

Function	Own station unit information setting			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	void ccief_basic_master_set_unit_info (uint16_t usUnitInfo)			
Argument	Type	Variable name	Description	I/O
	uint16_t	usUnitInfo	Application operating status	Input
Return value	-			
Description	This function sets the own station unit information.			

The following shows the definition of the application operating status based on the sample code.

[CCIEF_BASIC_MASTER.h]

```
#define CCIEF_BASIC_UNIT_INFO_APPLICATION_STOP      0x0000    /* Stopping application for setting the unit info */
#define CCIEF_BASIC_UNIT_INFO_APPLICATION_RUNNING   0x0001    /* Running application for setting the unit info */
```

5.6.18 ccief_basic_master_get_slave_info

Table 27 ccief_basic_master_get_slave_info

Function	Slave station receive information acquisition			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_get_slave_info (int iSlaveNumber, CCIEF_BASIC_SLAVE_INFO *pSlaveInfo)			
Argument	Type	Variable name	Description	I/O
	int	iSlaveNumber	Slave station number	Input
	CCIEF_BASIC_SLAVE_INFO *	pSlaveInfo	Storage location pointer for the information received from the slave station	Output
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_NG: Error			
Description	This function stores the slave station information specified by the argument iSlaveNumber in the argument pSlaveInfo.			

The following shows the configuration of CCIEF_BASIC_SLAVE_INFO based on the sample code.

[CCIEF_BASIC_MASTER.h]

```
typedef struct
{
        uint16_t                        usVenderCode;           /* Vender code */
        uint16_t                        usReserve1;             /* Reserve */
        uint32_t                        ulModelCode;            /* Model code */
        uint16_t                        usMachineVersion;       /* Machine version */
        uint16_t                        usReserve2;             /* Reserve */
        uint16_t                        usUnitInfo;             /* Information of the unit */
        uint16_t                        usErrCode;              /* Error code */
        uint32_t                        ulUnitData;             /* Data of the unit */
} CCIEF_BASIC_SLAVE_NOTIFY_INFO;

typedef struct
{
        uint16_t                        usProtocolVersion;      /* Protocol version */
        uint16_t                        usEndCode;              /* Error code of the slave */
        uint32_t                        ulId;                   /* Id of the slave */
        uint8_t                         ucGroupNumber;          /* Group number of the slave */
        uint16_t                        usFrameSequenceNumber;  /* Frame sequence number */
        int                             usOccupiedStationNumber;/* Number of occupied stations */
        int                             iState;                 /* State of the slave */
        CCIEF_BASIC_SLAVE_NOTIFY_INFO   NotifyInfo;             /* Notify information from the slave */
        int                             iCyclicState;           /* Cyclic state */
        int                             iStationNumber;         /* Number of stations */
} CCIEF_BASIC_SLAVE_INFO;
```

5.6.19 ccief_basic_master_get_group_info

Table 28 ccief_basic_master_get_group_info

Function	Group information acquisition			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Disclosed
Call format	int ccief_basic_master_get_group_info (int iGroupNumber, CCIEF_BASIC_GROUP_INFO *pGroupInfo)			
Argument	Type	Variable name	Description	I/O
	int	iGroupNumber	Slave station number	Input
	CCIEF_BASIC_GROUP_INFO *	pGroupInfo	Storage location pointer for the group information	Output
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_NG: Error			
Description	This function stores the master station information specified by the argument iSlaveNumber in the argument pGroupInfo.			

The following shows the configuration of CCIEF_BASIC_GROUP_INFO based on the sample code.

[CCIEF_BASIC_MASTER.h]

```
typedef struct
{
        uint16_t    usProtocolVersion;              /* Protocol version */
        uint32_t    ulId;                           /* Id of the master */
        uint8_t     ucGroupNumber;                  /* Group number of the slave */
        int         iTotalSlaveNumber;              /* Total number of the slaves */
        int         usTotalOccupiedStationNumber;   /* Total number of occupied stations */
        int         iState;                         /* State of Master */
        uint16_t    usUnitInfo;                     /* Information of the unit */
        uint16_t    usFrameSequenceNumber;          /* Frame sequence number */
        uint16_t    usParameterId;                  /* Parameter id */
        int64_t     llTimeData;                     /* Data of time */
        int64_t     llLinkScanTimeCurrent;          /* Current link scan time[us] */
        int64_t     llLinkScanTimeMinimum;          /* Minimum link scan time[us] */
        int64_t     llLinkScanTimeMaximum;          /* Maximum link scan time[us] */
} CCIEF_BASIC_GROUP_INFO;
```

5.6.20 ccief_basic_master_check_parameter

Table 29 ccief_basic_master_check_parameter

Function	Parameter check			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	int ccief_basic_master_check_parameter (CCIEF_BASIC_MASTER_PARAMETER *pParameter)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_MASTER_PARAMETER *	pParameter	Master station parameter	Input
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal CCIEF_BASIC_MASTER_ERR_NG: Error			
Description	This function checks the master station parameter specified by the argument pParameter.			

5.6.21 ccief_basic_master_recv

Table 30 ccief_basic_master_recv

Function	Packet receiving			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	int ccief_basic_master_recv (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal SOCKET_ERR_RECV: Socket error CCIEF_BASIC_MASTER_ERR_MASTER_DUPLICATION: Master station duplication			
Description	This function receives a packet of the CCIEF-BASIC slave station. Change the program according to the implementation environment. * This function must be regularly executed.			

5.6.22 ccief_basic_master_polling

Table 31 ccief_basic_master_polling

Function	Regular execution			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	int ccief_basic_master_polling (CCIEF_BASIC_MASTER_GROUP_INFO *pGroup)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_MASTER_GROUP_INFO *	pGroup	Group information	Input
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal			
Description	This function: - executes regular processing of the cyclic transmission group specified by the argument pGroup; and - checks link scan completion. * This function must be regularly executed.			

5.6.23 ccief_basic_master_execute_state

Table 32 ccief_basic_master_execute_state

Function	Master station status execution			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_master_execute_state (CCIEF_BASIC_MASTER_GROUP_INFO *pGroup, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_MASTER_GROUP_INFO *	pGroup	Group information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function: - executes the master station state; and - sorts the processing for each status type. * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.24 ccief_basic_master_execute_state_wait_cyclic

Table 33 ccief_basic_master_execute_state_wait_cyclic

Function	Master station status execution (On standby)			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_master_execute_state_wait_cyclic (CCIEF_BASIC_MASTER_GROUP_INFO *pGroup, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_MASTER_GROUP_INFO *	pGroup	Group information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the master station state "On standby". * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.25 ccief_basic_master_execute_state_persuasion

Table 34 ccief_basic_master_execute_state_persuasion

Function	Master station status execution (Master station arbitration being performed)			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_master_execute_state_persuasion (CCIEF_BASIC_MASTER_GROUP_INFO *pGroup, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_MASTER_GROUP_INFO *	pGroup	Group information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the master station state "Master station arbitration being performed". * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.26 ccief_basic_master_execute_state_linkscan_end

Table 35 ccief_basic_master_execute_state_linkscan_end

Function	Master station status execution (Link scan completed)			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_master_execute_state_linkscan_end (CCIEF_BASIC_MASTER_GROUP_INFO *pGroup, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_MASTER_GROUP_INFO *	pGroup	Group information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the master station state "Link scan completed". * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.27 ccief_basic_master_execute_state_linkscan

Table 36 ccief_basic_master_execute_state_linkscan

Function	Master station status execution (Link scan being performed)		
File name	CCIEF_BASIC_MASTER.c	Disclosed/undisclosed	Undisclosed

Call format	void ccief_basic_master_execute_state_linkscan (CCIEF_BASIC_MASTER_GROUP_INFO *pGroup, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_MASTER_GROUP_INFO *	pGroup	Group information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the master station state "Link scan being performed". v			

5.6.28 ccief_basic_master_persuasion_timer_timeout

Table 37 ccief_basic_master_persuasion_timer_timeout

Function	Timeout of frame monitoring time (callback function)			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_master_persuasion_timer_timeout (int iId, void *pCallbackArg)			
Argument	Type	Variable name	Description	I/O
	int	iId	Timer ID	Input
	void *	pCallbackArg	Storage source pointer for the group information	Input
Return value	-			
Description	This callback function executes by the tier function when the frame monitoring time has timed out. The function issues the event for "Timeout of frame monitoring time".			

5.6.29 ccief_basic_master_cyclic_timer_timeout

Table 38 ccief_basic_master_cyclic_timer_timeout

Function	Timeout of cyclic transmission (callback function)			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_master_cyclic_timer_timeout (int iId, void *pCallbackArg)			
Argument	Type	Variable name	Description	I/O
	int	iId	Timer ID	Input
	void *	pCallbackArg	Storage source pointer for the group information	Input
Return value	-			
Description	This callback function is executed by the timer function when the cyclic transmission has timed out. The function issues the event for "Link scan completed".			

5.6.30 ccief_basic_master_make_cyclic_data

Table 39 ccief_basic_master_make_cyclic_data

Function	Cyclic transmission data generation			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	int ccief_basic_master_make_cyclic_data (CCIEF_BASIC_MASTER_GROUP_INFO *pGroup)			
Argument	Type	Variable name	Description	I/O

	CCIEF_BASIC_MASTER_GROUP_INFO *	pGroup	Group information	Input
Return value	Data size (byte)			
Description	This function: - creates a request data for cyclic transmission; and - set the created data size to the return value.			

5.6.31 ccief_basic_master_send_cyclic_data

Table 40 ccief_basic_master_send_cyclic_data

Function	Cyclic transmission data sending			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	int ccief_basic_master_send_cyclic_data (uint32_t ulIpAddress, uint8_t *pucData, int iDataSize)			
Argument	Type	Variable name	Description	I/O
	uint32_t	ulIpAddress	Group information	Input
	uint8_t *	pucData	Storage source pointer for send data	Input
	int	iDataSize	Send data size	Input
Return value	CCIEF_BASIC_MASTER_ERR_OK: Normal			
Description	This function sends a request data for cyclic transmission.			

5.6.32 ccief_basic_master_recv_cyclic_data_response

Table 41 ccief_basic_master_recv_cyclic_data_response

Function	Cyclic transmission data response receiving			
File name	CCIEF_BASIC_MASTER.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_master_recv_cyclic_data_response (uint8_t *pucData)			
Argument	Type	Variable name	Description	I/O
	uint8_t *	pucData	Storage source pointer for receive data	Input
Return value	-			
Description	This function receives the response data of cyclic transmission.			

5.6.33 ccief_basic_slaves_initialize

Table 42 ccief_basic_slaves_initialize

Function	Slave station status initialization			
File name	CCIEF_BASIC_SLAVES.c		Disclosed/undisclosed	Disclosed
Call format	void ccief_basic_slaves_initialize (CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *pSlave)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *	pSlave	Slave station cyclic transmission information	Input
Return value	-			
Description	This function initializes the slave station status.			

The following shows the configuration of the CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO based on the sample code.

[CCIEF_BASIC_SLAVES.h]

```
typedef struct
{
    CCIEF_BASIC_SLAVE_PARAMETER  *pParameter;                    /* Parameter */
    int                          iNumber;                        /* Slave Number */
    uint32_t                     ulId;                           /* Id number */
    int                          iStationNumber;                 /* Number of stations */
    int                          iGroupStationNumber;            /* Number of stations for the group */
    int                          iCyclicStart;                   /* Start cyclic of the user operation */
    int                          iState;                         /* State of Slave */
    int                          iCyclicState;                   /* Cyclic state */
    int                          iCyclicStateSet;                /* Setting of the cyclic state */
    int                          iReceiveComplete;               /* State of response recieve */
    int                          iDuplicateState;                /* State of slave duplication */
    uint16_t                     *pusFrameSequenceNumber;        /* Frame sequence number of the master */
    uint16_t                     usProtocolVersion;              /* Protocol version of the slave */
    uint16_t                     usEndCode;                      /* End code of the slave */
    uint16_t                     usFrameSequenceNumber;          /* Frame sequence number of the slave */
    CCIEF_BASIC_SLAVE_NOTIFY_INFO NotifyInfo;                    /* Notification information of the slave */
    uint16_t                     usCyclicTransmissionTimeoutCount; /* Count of cyclic transmission timeout */
    uint16_t                     usTimeoutCount;          /* Counter of timeout for the cyclic transmission timeout */
    uint16_t                     *pusRWw;                        /* Pointer of RWw for the packet */
    uint16_t                     *pusRY;                         /* Pointer of RY for the packet */
    uint16_t                     *pusRWr;                        /* Pointer of RWr for the packet */
    uint16_t                     *pusRX;                         /* Pointer of RX for the packet */
    uint16_t                     *pusSlaveRWr;                   /* Pointer of RWr for the slave */
    uint16_t                     *pusSlaveRX;                    /* Pointer of RX for the slave */
} CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO;
```

5.6.34 ccief_basic_slaves_execute_state

Table 43 ccief_basic_slaves_execute_state

Function	Slave station status execution			
File name	CCIEF_BASIC_SLAVES.c		Disclosed/undisclosed	Disclosed
Call format	void ccief_basic_slaves_execute_state (CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *pSlave, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *	pSlave	Slave station cyclic transmission information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function: - executes the slave station state. - sorts the processing for each status type. * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.35 ccief_basic_slaves_execute_state_disconnect

Table 44 ccief_basic_slaves_execute_state_disconnect

Function	Slave station status execution (Disconnected)			
File name	CCIEF_BASIC_SLAVES.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_slaves_execute_state_disconnect (CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *pSlave, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *	pSlave	Slave station cyclic transmission information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the slave station state "Disconnected". * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.36 ccief_basic_slaves_execute_state_connecting

Table 45 ccief_basic_slaves_execute_state_connecting

Function	Slave station status execution (Waiting for return)			
File name	CCIEF_BASIC_SLAVES.c	Disclosed/undisclosed	Undisclosed	
Call format	void ccief_basic_slaves_execute_state_connecting (CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *pSlave, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *	pSlave	Slave station cyclic transmission information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the slave station state "Waiting for return". * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.37 ccief_basic_slaves_execute_state_cyclic_stop

Table 46 ccief_basic_slaves_execute_state_cyclic_stop

Function	Slave station status execution (Cyclic transmission stopped)			
File name	CCIEF_BASIC_SLAVES.c	Disclosed/undisclosed	Undisclosed	
Call format	void ccief_basic_slaves_execute_state_cyclic_stop (CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *pSlave, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *	pSlave	Slave station cyclic transmission information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the slave station state "Cyclic transmission stopped". * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.38 ccief_basic_slaves_execute_state_cyclic_end

Table 47 ccief_basic_slaves_execute_state_cyclic_end

Function	Slave station status execution (Cyclic transmission completed)			
File name	CCIEF_BASIC_SLAVES.c	Disclosed/undisclosed	Undisclosed	
Call format	void ccief_basic_slaves_execute_state_cyclic_end (CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *pSlave, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *	pSlave	Slave station cyclic transmission information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the slave station state "Cyclic transmission completed". * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.39 ccief_basic_slaves_execute_state_cyclic

Table 48 ccief_basic_slaves_execute_state_cyclic

Function	Slave station status execution (Cyclic transmission being performed)			
File name	CCIEF_BASIC_SLAVES.c		Disclosed/undisclosed	Undisclosed
Call format	void ccief_basic_slaves_execute_state_cyclic (CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *pSlave, int iEvent)			
Argument	Type	Variable name	Description	I/O
	CCIEF_BASIC_SLAVES_CYCLIC_DATA_INFO *	pSlave	Slave station cyclic transmission information	Input
	int	iEvent	Event	Input
Return value	-			
Description	This function executes the processing for the slave station state "Cyclic transmission being performed". * For details, refer to the CC-Link IE Field Network Basic Specification (Application Layer Protocol).			

5.6.40 socket_initialize

Table 49 socket_initialize

Function	Socket initialization			
File name	SOCKET.c		Disclosed/undisclosed	Disclosed
Call format	int socket_initialize (SOCKET *sock, uint32_t ulIpAddress, uint16_t usPortNumber)			
Argument	Type	Variable name	Description	I/O
	SOCKET *	sock	Storage location pointer for the socket descriptor	Output
	uint32_t	ulIpAddress	IP address	Input
	uint16_t	usPortNumber	Port number	Input
Return value	SOCKET_ERR_OK: Normal SOCKET_ERR_SOCKET: Socket generation error			
Description	This function: - initializes the socket; and - returns the socket descriptor. Change the program according to the implementation environment.			

5.6.41 socket_terminate

Table 50 socket_terminate

Function	Socket termination			
File name	SOCKET.c		Disclosed/undisclosed	Disclosed
Call format	void socket_terminate (SOCKET sock)			
Argument	Type	Variable name	Description	I/O
	SOCKET	sock	Socket descriptor	Input
Return value	-			
Description	This function terminates the socket. Change the program according to the implementation environment.			

5.6.42 socket_recv

Table 51 socket_recv

Function	Packet receiving			
File name	SOCKET.c		Disclosed/undisclosed	Disclosed
Call format	int socket_recv (SOCKET sock, uint8_t *pucStream, int iLength, uint32_t *pulRecvAddr, uint16_t *pusRecvPortNumber)			
Argument	Type	Variable name	Description	I/O
	SOCKET	sock	Socket descriptor	Input
	uint8_t *	pucStream	Receive packet	Output
	int	iLength	Receive packet length	Input
	uint32_t *	pulRecvAddr	Send source IP address	Output
	uint16_t *	pusRecvPortNumber	Send source port number	Output
Return value	SOCKET_ERR_OK: Normal SOCKET_ERR_NO_RECEIVABLE: No receive data SOCKET_ERR_RECV: Socket receive error			
Description	This function receives a packet. Change the program according to the implementation environment. * This function must be regularly executed.			

5.6.43 socket_send

Table 52 socket_send

Function	Packet sending			
File name	SOCKET.c		Disclosed/undisclosed	Disclosed
Call format	int socket_send (SOCKET sock, uint8_t *pucStream, int iLength, uint32_t ulSendAddr, uint16_t usSendPortNumber)			
Argument	Type	Variable name	Description	I/O
	SOCKET	sock	Socket descriptor	Input
	uint8_t *	pucStream	Send packet	Input
	int	iLength	Send packet length	Input
	uint32_t	ulSendAddr	Send destination IP address	Input
	uint16_t	usSendPortNumber	Send destination port number	Input
Return value	SOCKET_ERR_OK: Normal SOCKET_ERR_SEND: Socket send error			
Description	This function sends a packet. Change the program according to the implementation environment.			

5.6.44 timer_initialize

Table 53 timer_initialize

Function	Timer initialization			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	void timer_initialize (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function initializes the timer function. * The maximum number of the timer is defined with "TIMER_MAX".			

5.6.45 timer_terminate

Table 54 timer_terminate

Function	Timer termination			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	void timer_terminate (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function terminates the timer function.			

5.6.46 timer_main

Table 55 timer_main

Function	Timer main processing			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	void timer_main (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function executes timer main processing. When the time is up, this function executes the callback function specified by the user. * This function must be regularly executed.			

5.6.47 timer_start

Table 56 timer_start

Function	Timer start			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	int timer_start (long lTime, int *piId, TIMER_CALLBACK pCallbackFunc, void *pCallbackArg)			
Argument	Type	Variable name	Description	I/O
	long	lTime	Timeout period [ms]	Input
	int *	piId	Storage location pointer for the timer ID	Output
	TIMER_CALLBACK	pCallbackFunc	Storage source pointer for the callback function	Input
	void *	pCallbackArg	Argument of the callback function	Input
Return value	TIMER_OK: Normal TIMER_RESOURCE_NONE: Timer exhaustion			
Description	This function: - starts the timer; - sets the started timer ID to the storage location specified by the argument piId; - registers the callback function specified by the argument pCallbackFunc; and - registers the callback argument specified by the argument pCallbackArg.			

5.6.48 timer_stop

Table 57 timer_stop

Function	Timer stop			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	void timer_main (int iId)			
Argument	Type	Variable name	Description	I/O
	int	iId	Timer ID	Input
Return value	-			
Description	This function stops the timer specified by the argument piId.			

5.6.49 timer_get_time

Table 58 timer_get_time

Function	Current time acquisition			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	int64_t timer_get_time (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	Current time (system time) [us]			
Description	This function acquires current system time.			

5.6.50 timer_calculate_time_data

Table 59 timer_calculate_time_data

Function	Clock information calculation			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	int64_t timer_calculate_time_data (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	Clock information			
Description	This function calculates the clock information (UNIX time) based on the current system time. * The clock information is calculated based on the current time.			

5.6.51 timer_analyze_time_data

Table 60 timer_analyze_time_data

Function	Clock time analysis			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	void timer_analyze_time_data (int64_t llTime, TIMER_TIME_DATA *pTimeData)			
Argument	Type	Variable name	Description	I/O
	int64_t	llTime	Clock information (UNIX time)	Input
	TIMER_TIME_DATA *	pTimeData	Storage location pointer for clock data	Output
Return value	-			
Description	This function analyzes the clock information (UNIX time) specified by the argument llTime and store it in the argument pTimeData.			

The following shows the configuration of the TIMER_TIME_DATA based on the sample code.

[TIMER.h]

```
typedef struct
{
        uint16_t    usYear;         /* Year */
        uint16_t    usMonth;        /* Month */
        uint16_t    usDay;          /* Day */
        uint16_t    usHour;         /* Hour */
        uint16_t    usMinute;       /* Minute */
        uint16_t    usSecond;       /* Second */
        uint16_t    usMilliseconds; /* Milliseconds */
} TIMER_TIME_DATA;
```

5.6.52 timer_gettimeofday

Table 61 timer_gettimeofday

Function	System time acquisition			
File name	TIMER.c		Disclosed/undisclosed	Disclosed
Call format	int timer_gettimeofday (struct timeval *tv, struct timezone *tz)			
Argument	Type	Variable name	Description	I/O
	struct timeval *	tv	Storage location pointer for the system time [us]	Output
	struct timezone *	tz	Storage location pointer for the timezone	Output
Return value	TIMER_OK: Normal			
Description	This function acquires the system time [us]. Change the program according to the implementation environment. * The sample code describes an example of implementing gettimeofday which is a POSIX standard function on WindowsOS.			

5.6.53 main

Table 62 main

Function	Main processing			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Disclosed
Call format	void main (int argc, char *argv[])			
Argument	Type	Variable name	Description	I/O
	int	argc	Total number of command line arguments	Input
	char *	argv[]	Command line argument	Input
Return value	-			
Description	This function: - acquires network adapter information; - reads parameters; - initializes the timer function; and - executes the main processing of the timer function (loop processing). Change the program according to the implementation environment.			

5.6.54 user_callback_cyclic_link_scan_end

Table 63 user_callback_cyclic_link_scan_end

Function	Link scan completed (callback function)			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Disclosed
Call format	void user_callback_cyclic_link_scan_end (uint8_t ucGroupNumber)			
Argument	Type	Variable name	Description	I/O
	uint8_t	ucGroupNumber	Group number	Input
Return value	-			
Description	The function is executed upon completion of the link scan by the master station. When there are multiple groups for cyclic transmission, this function is executed for each group number of the argument ucGroupNumber. Change the program according to the implementation environment. * In the sample code, this function sets the slave station ID to the RY and RWw of the slave station belonging to the specified group number.			

5.6.55 user_parameter_file_read

Table 64 user_parameter_file_read

Function	Parameter file reading			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	int user_parameter_file_read (char *file_path, CCIEF_BASIC_MASTER_PARAMETER *pParameter)			
Argument	Type	Variable name	Description	I/O
	char *	file_path	File	Input
	CCIEF_BASIC_MASTER_PARAMETER *	pParameter	Storage location pointer for the master station parameter	Output
Return value	USER_ERR_OK: Normal USER_ERR_NG: Error			
Description	This function reads the file specified by the argument file_path and stores it in the argument pParameter. Change the program according to the implementation environment.			

5.6.56 user_get_input_line

Table 65 user_get_input_line

Function	Input character string acquisition			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_get_input_line (char *pcLine, int iLineLength)			
Argument	Type	Variable name	Description	I/O
	char *	pcLine	Storage location pointer for character strings	Output
	int	iLineLength	Maximum input characters	Input
Return value	-			
Description	This function acquires the character string input by the user. (Input is completed when the Enter key is pressed.) Change the program according to the implementation environment.			

5.6.57 user_show_menu_top

Table 66 user_show_menu_top

Function	Menu screen display			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_show_menu_top (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function displays the menu screen. Change the program according to the implementation environment.			

5.6.58 user_input_check

Table 67 user_input_check

Function	Input check			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	int user_input_check (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	USER_ERR_OK: Normal USER_EXIT: Exit			
Description	This function checks the key input of the user on the menu screen. Change the program according to the implementation environment. * This function must be regularly executed.			

5.6.59 user_start_cyclic

Table 68 user_start_cyclic

Function	Cyclic transmission start			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_start_cyclic (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function starts cyclic transmission. (Starts the cyclic transmission for all slave stations specified by the parameter setting.) Change the program according to the implementation environment.			

5.6.60 user_stop_cyclic

Table 69 user_stop_cyclic

Function	Cyclic transmission stop			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_stop_cyclic (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function stops cyclic transmission. (Stops the cyclic transmission for all slave stations specified by the parameter setting.) Change the program according to the implementation environment.			

5.6.61 user_start_application

Table 70 user_start_application

Function	Application start			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_start_application (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function starts application. Change the program according to the implementation environment.			

5.6.62 user_stop_application

Table 71 user_stop_application

Function	Application stop			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_stop_application (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function stops application. Change the program according to the implementation environment.			

5.6.63 user_show_slave_info

Table 72 user_show_slave_info

Function	Slave station information display			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_show_slave_info (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function displays the slave station information on the screen. (Displays the information for all slave stations set by the parameters.) Change the program according to the implementation environment.			

5.6.64 user_show_master_info

Table 73 user_show_master_info

Function	Master station information display			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_show_master_info (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function displays the master station information on the screen. Change the program according to the implementation environment.			

5.6.65 user_show_parameter

Table 74 user_show_parameter

Function	Parameter display			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	void user_show_parameter (void)			
Argument	Type	Variable name	Description	I/O
	-	-	-	-
Return value	-			
Description	This function displays the master station parameters on the screen. Change the program according to the implementation environment.			

5.6.66 user_get_adapter_info

Table 75 user_get_adapter_info

Function	Network adapter information acquisition			
File name	USER_SAMPLE.c		Disclosed/undisclosed	Undisclosed
Call format	int user_get_adapter_info (USER_ADAPTER_INFO *pGetAdapterInfo)			
Argument	Type	Variable name	Description	I/O
	USER_ADAPTER_INFO *	pGetAdapterInfo	Storage location pointer for network adapter information	Output
Return value	USER_ERR_OK: Normal USER_ERR_NG: Error			
Description	This function acquires the network adapter information. Change the program according to the environment. * The sample code consists network adapter acquiring example on Windows OS.			

6 Appendix: Procedure from compilation to execution of sample code

This section describes the procedure from compilation to execution of the sample code when "gcc" is used.
This code is compiled on the CentOS based system.

6.1 Specifications

Execute the sample code under the environment shown in Figure 13.

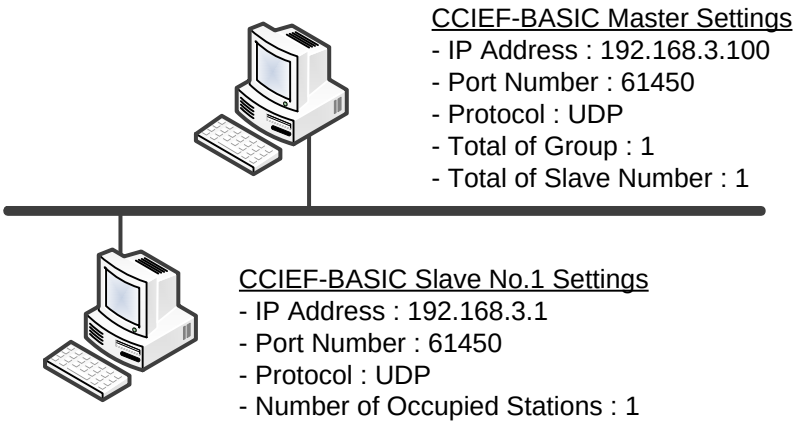

Figure 13 Execution Environment of the Sample Code

The sample code executes the cyclic transmission with the CCIEF-BASIC slave station (sample application[*1]).
The command line argument specifies the parameter file set by the user and starts the application.
If multiple network adapters are mounted in the execution environment, the selection window of the mounted network adapter is displayed. The sample code is started with the network adapter selected by the user.

*1 For details, refer to the "CC-Link IE Field Network Basic Sample Code User's Manual (Slave Station)".

6.2 Creating an application

This section describes the procedure to create an executable module using gcc.

(1) Extract sample code tree.

(2) cd CCIEF-BASIC_Master

(3) Do the following command.

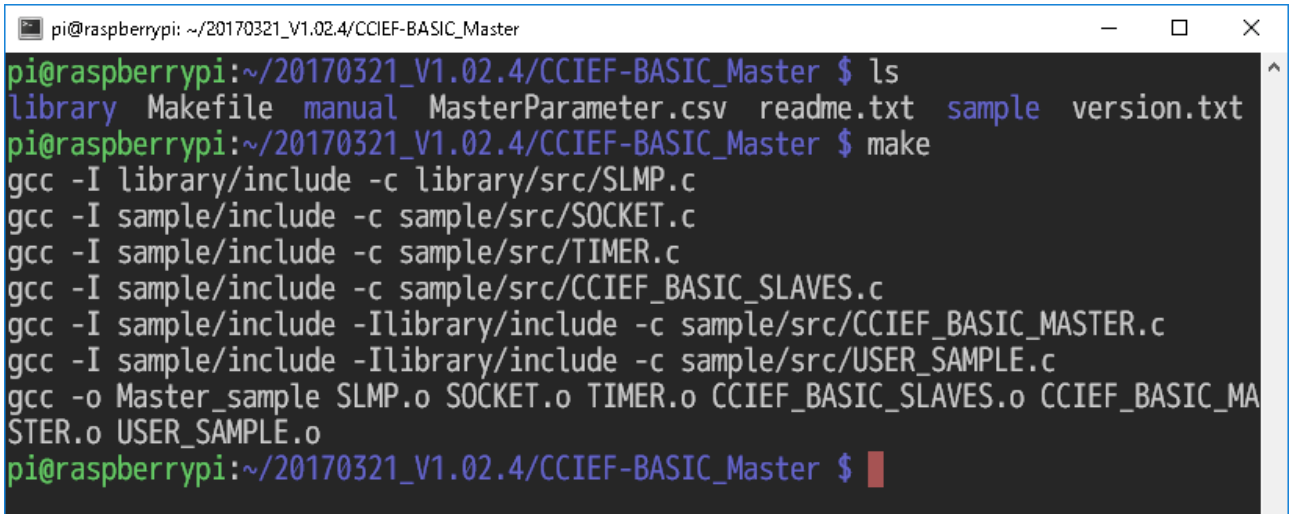

Figure 14 Compile command

6.3 Executing an application

This section describes the procedure to execute an application using gcc.

(1) Create a parameter file of the master station. (For details, refer to "4 Specifications")

[MasterParameter.csv]

```
,,
CCIEF-BASIC Master Sample Parameter,,
,,
Group,,
ID,DATA,COMMENT
1,1,Total number of group
2,1,Number of group
3,100,Group1 Cyclic transmission timeout
4,2,Group1 Count of cyclic transmission timeout
5,0,Group1 Constant link scan time
,,
Slave,,
ID,DATA,COMMENT
1,1,Total number of slave
2,192.168.3.1,Slave1 IP address
3,1,Slave1 Number of occupied stations
4,1,Slave1 Number of group
```

(2) Execute application as following.

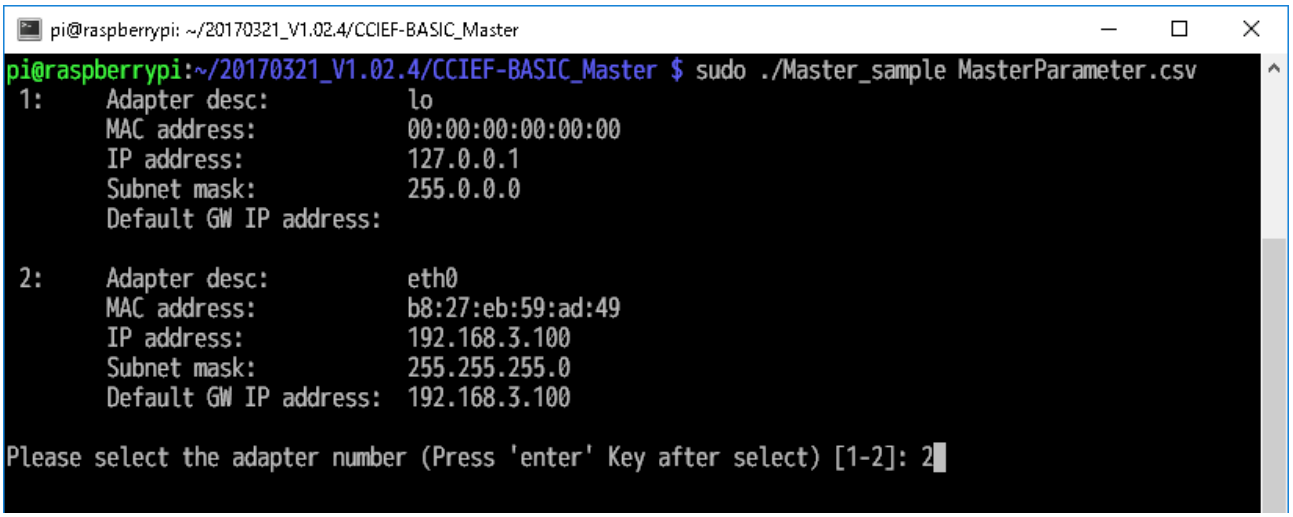

Figure 15 Execute Application

(3) When the application is normally executed, the output is as follows.

```
Start CC-Link IE Field Basic Master Station!

Show master parameter!

    Master:
      IP Address:                      192.168.3.100 (Master ID:0xC0A80364)
      Subnet mask:                     255.255.255.0
      Defautl GW IP address:           192.168.3.254

    Total Number of Group:             1
      Group No.1:
        Discconection Time[ms]:        100 (0:500[ms])
        Discconection Timeout Count:   2 (0:3)
        Constant Link Scan Time[ms]:   Not use

    Total Number of Slave:             1
      Slave No.1:
        IP Address:                    192.168.3.1 (Slave ID:0xC0A80301)
        Occupied Station Number:       1
        Group Number:                  1

Start cyclic of all the slaves!

Start the application!

Please input the followng key values if you want any action.

    '1'   - Start the cyclic.
    '2'   - Stop the cyclic.
    '3'   - Start the application.
    '4'   - Stop the application.
    '5'   - Show information of the slave.
    '6'   - Show information of the master.
    '7'   - Show the parameter.
    'Esc' - Exit the application.
```

Figure 16 Start Window of the Master Station

(4) The following shows examples of the user operation of the application.

(a) Press the '1' key.

```
Start cyclic of all the slaves!
```

Figure 17 Pressing the '1' Key

(b) Press the '2' key.

```
Stop cyclic of all the slaves!
```

Figure 18 Pressing the '2' Key

(c) Press the '3' key.

```
Start the application!
```

Figure 19 Pressing the '3' Key

(d) Press the '4' key.

```
Stop the application!
```

Figure 20 Pressing the '4' Key

(e) Press the '5' key.

```
Show the state of all the slaves!

    Slave No.1:
      Slave ID:                    0xC0A80301
      Occupied Station Number:     1
      Group No.:                   1
      State:                       5 [CYCLIC]
      Protocol Version:            0x0001
      Error Code:                  0x0000
      Slave Notify Information:
        Vender Code:               0x1234
        Model Code:                0x00010001
        Machine Version:           0x0001
        Unit Info:                 0x0001
        Error Code:                0x0000
        Unit Data:                 0x00000000
      Frame sequence number:       0x0385

        RX  | F E D C B A 9 8 7 6 5 4 3 2 1 0 |  data
       ---------------------------------------------
        0000 | 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 | 0x0386
        0010 | 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 | 0x0386
        0020 | 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 | 0x0386
        0030 | 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 | 0x0386

        RY  | F E D C B A 9 8 7 6 5 4 3 2 1 0 |  data
       ---------------------------------------------
        0000 | 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 | 0x0387
        0010 | 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 | 0x0387
        0020 | 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 | 0x0387
        0030 | 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 | 0x0387

        RWw |   +7     +6     +5     +4     +3     +2     +1     +0
       ------------------------------------------------------------
        0000 | 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387
        0008 | 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387
        0010 | 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387
        0018 | 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387 0x0387

        RWr |   +7     +6     +5     +4     +3     +2     +1     +0
       ------------------------------------------------------------
        0000 | 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386
        0008 | 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386
        0010 | 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386
        0018 | 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386 0x0386
```

Figure 21 Pressing the '5' Key

(f) Press the '6' key.

```
Show the state of master!

    Master:
      Protocol Version:                 0x0001
      Master ID:                        0xC0A80364
      Unit Info:                        0x0001
      Parameter ID:                     0x6C32

    Group No.1:
      Total Number of Slave:            1
      Total Number of Occupied Station: 1
      State:                            4 [LINK_SCAN_END]
      Time Data:                        1459868384442 [2016-03-31 14:59:44.442]
      Frame sequence number:            0x060E
      Link scan time(Current):          1.133[ms]
      Link scan time(Minimum):          0.964[ms]
      Link scan time(Maximum):          3.046[ms]
      Group:
          Master(ID:100)
              |--- Slave No.1  (ID:0xC0A80301  CyclicState:ON  State:5 [CYCLIC])
```

Figure 22 Pressing the '6' Key

(g) Press the '7' key.

```
Show master parameter!

    Master:
      IP Address:                       192.168.3.100 (Master ID:0xC0A80364)
      Subnet mask:                      255.255.255.0
      Defautl GW IP address:            192.168.3.254
    Total Number of Group:              1
      Group No.1:
        Discconection Time[ms]:         100 (0:500[ms])
        Discconection Timeout Count:    2 (0:3)
        Constant Link Scan Time[ms]:    Not use
    Total number of slave:              1
      Slave No.1:
        IP Address:                     192.168.3.1 (Slave ID:0xC0A80301)
        Occupied Station Number:        1
        Group Number:                   1
```

Figure 23 Pressing the '7' Key

(h) Press the 'Esc' key.

```
Exit the application? (if you want exit, please press 'Y')

Application has exited.(please any press)
```

Figure 24 Pressing the 'Esc' Key