GL-9981

Programming Manual

[History]

Rev	Page	Remarks	Date	Editor
1.00		Draft	2022.07.28	TE KIM

Table of Contents

1. Default information for GL-9981

Network

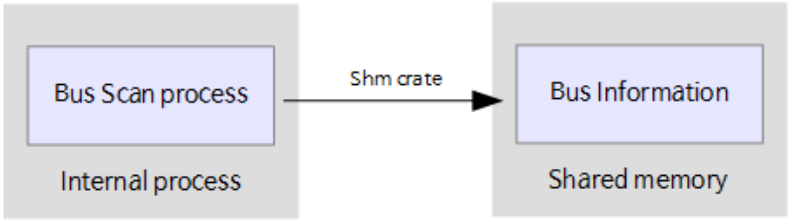
- IP: 192.168.100.72
- Subnetmask: 255.255.255.0
- router: 192.168.100.1

login credentials

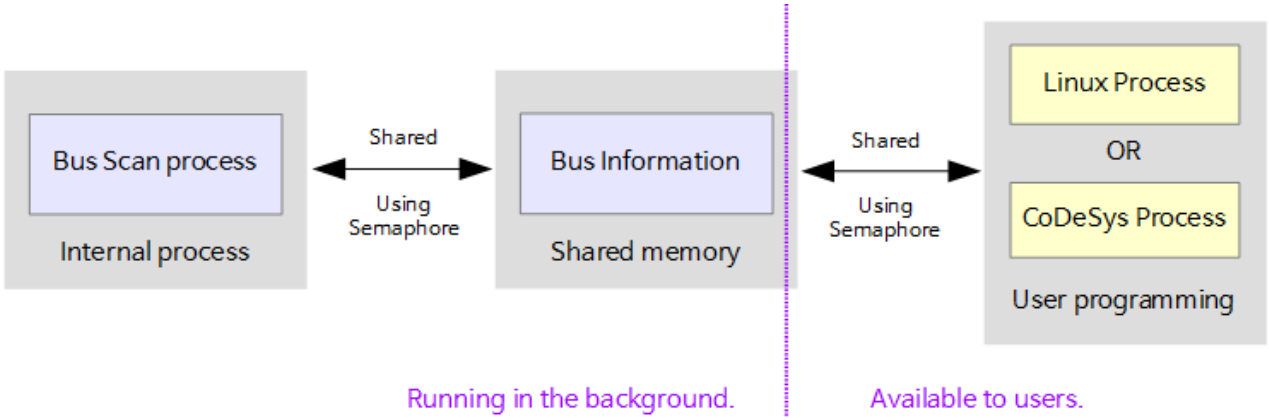
- username: crevis
- Password: user
- root Password: cvuser

2. programming configuration diagram

- Internal process creates shared memory



- Internal processes and user programs share shared memory using semaphores.



!!Caution

 # If any user program is executed earlier than Bus Scan Process, it will not operate normally

- Bus Scan Process (pibus.service) is automatically executed using SYSTEMD.
- File location of pibus.service: /lib/systemd/system/pibus.service
- Location of actual executable file: /home/crevis/pibus/M_PibusScan
- Only one of the Linux Process or CoDeSys process should be running. (Cannot run concurrently)

3. How to use Linux Version (GL-9981-L)

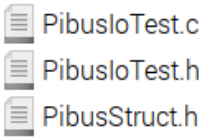
GL-9981 boot and check the IOS LED

- Turn on the power of GL-9981 to boot.
- After booting is complete, the 'BUS Scan process' program is automatically executed.
- When the IO scan is completed and the IOS LED turns green normally,

  You can write and use a user program to control IO.

It provides a method to control IO in the form of a C language example

- The location of the file : /home/crevis/GL-9981-example_rev100



- PibusIoTest.c : Example program that can control IO.
- PibusIotest.h : Definition of pointers and global variables used for IO control.
- PibusStruct.h : Definition of structure used in internal bus.

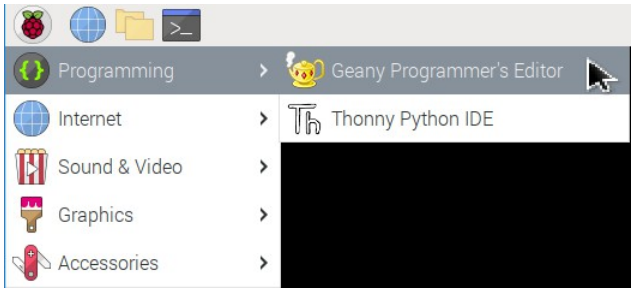
Description of functions used in PibusIoTest.c example

void pibusInit(void)	Initialize the internal bus Initialize the semaphore to use. Initialize shared memory.
int Read_InputImage(u16 Addr, u8* pBuffer, u16 Len)	Read values from input image addr - starting address / pBuffer - use buffer / Len - length Return(-1) - Abnormal Addr and Len values.
int Read_OutputImage(u16 Addr, u8* pBuffer, u16 Len)	Read values from output image addr - starting address / pBuffer - use buffer / Len - length Return(-1) - Abnormal Addr and Len values.
int Write_OutputImage(u16 Addr, u8* pBuffer, u16 Len)	Write values from output image addr - starting address / pBuffer - use buffer / Len - length Return(-1) - Abnormal Addr and Len values.
void Read_Slot(u8 slot, u8* pBuffer) * Input module only	Read the value of the slot slot - slot number / pBuffer - use buffer
void Write_Slot(u8 slot, u8* pBuffer) * Output module only	Writes the value of the slot slot - slot number / pBuffer - use buffer
void Get_Prm(u8 slot, u8* pBuffer)	Read the parameters of the slot slot - slot number / pBuffer - use buffer
void Set_Prm(u8 slot, u8* pBuffer)	Write the parameters of the slot slot - slot number / pBuffer - use buffer

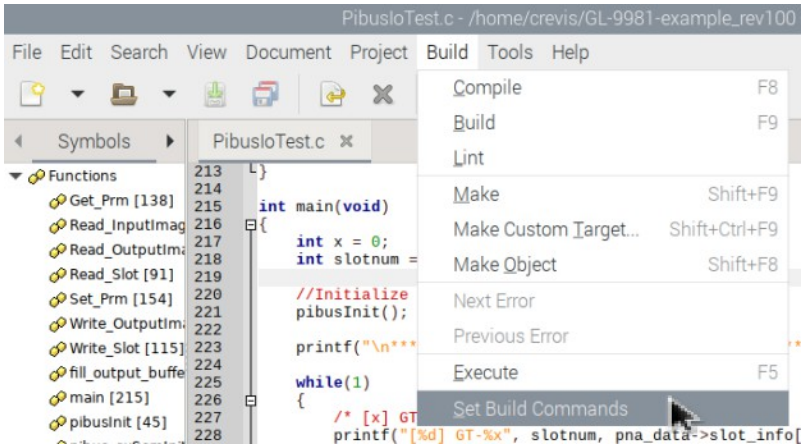
Development Environment - Using the geany editor of Raspberry Pi

1. Connect a monitor, mouse, and keyboard to the Raspberry Pi.

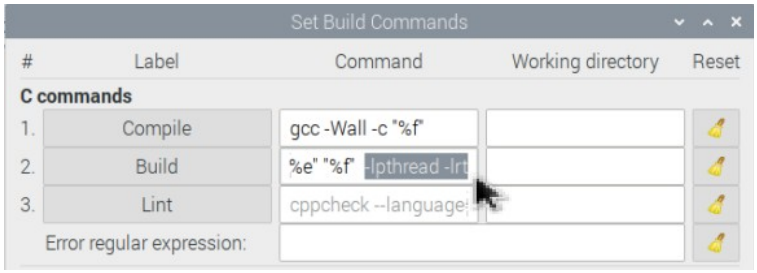
2. When the power is turned on and booting is completed, click 'Programming - Geany Programmer`s Editor'.



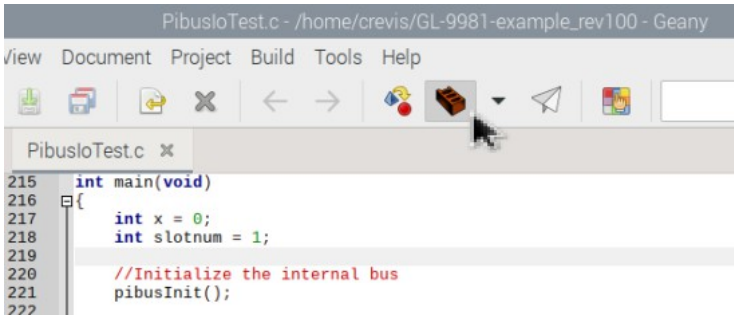
3. Click 'Build - Set Build Commands' to set the build.

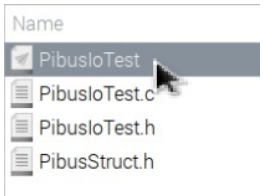


4. Add -lpthread -lrt to the end.



5. If you open the example program and click the build icon, 'compilation finished successfully.' An executable file is created with a message.

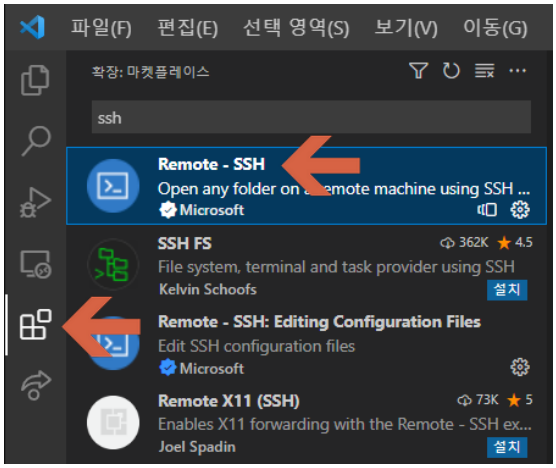




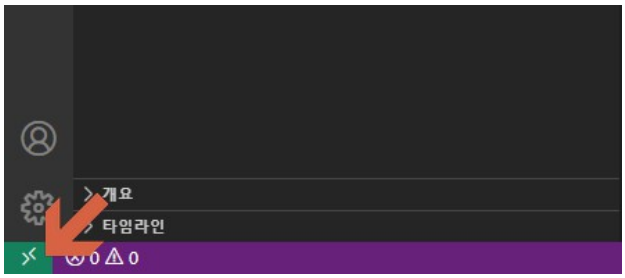
Development environment - Remote access using vscode

1. Run after installing 'vscode' on the PC to be connected remotely.

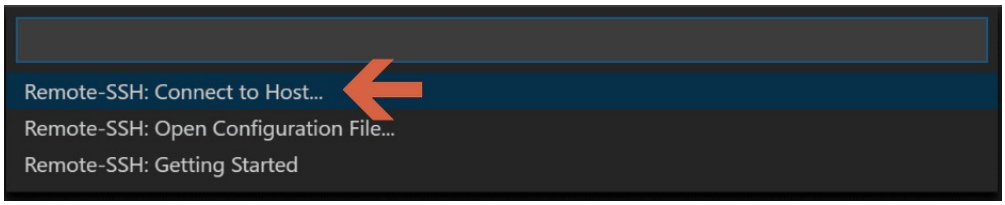
2. Click the extension tab and install 'Remote - SSH'.



3. Click the remote icon at the bottom left of the screen.

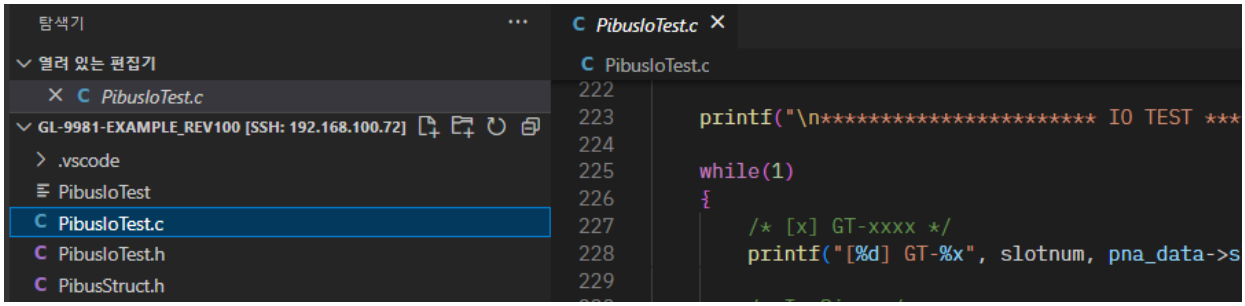


4. Click 'Connect to Host'.



5. Enter the IP to connect to. (The Raspberry Pi and PC to be connected must have the same IP band.)

6. After successful remote connection, open the folder to be edited.



Execution of the PibusIoTest.c example

- It can be compiled with the following command.

crevis@raspberrypi:~/GL-9981-example_rev100 $ gcc -o PibusIoTest PibusIoTest.c -lpthread -lrt

- This is an example of reading and writing connected IO.

- When you run it, you will see the following screen.

```
********** IO TEST **********
[1] GT-1238
In: ff
Out:
Get Prm: 55 aa
====================================================
[2] GT-2628
In:
Out: ff
Get Prm: 00 00
====================================================
[3] GT-2764
In:
Out: ff
Get Prm: 05 05
====================================================
[4] GT-3118
In: ff 0f ff 0f fd 0f ff 0f ff 0f ff 0f ff 0f ff 0f
Out:
Get Prm: ff aa 00 00 00 00 00 00 00 9a
====================================================
[5] GT-4118
In:
Out: ff 0f ff 0f ff 0f ff 0f ff 0f ff 0f ff 0f ff 0f
Get Prm: 00 00 00 00
====================================================
[6] GT-2764
In:
Out: ff
Get Prm: 00 02
====================================================
[7] GT-22ba
In:
Out: ff ff ff ff
Get Prm: 00 00 00 00 00 00 00 00
====================================================
.....
```

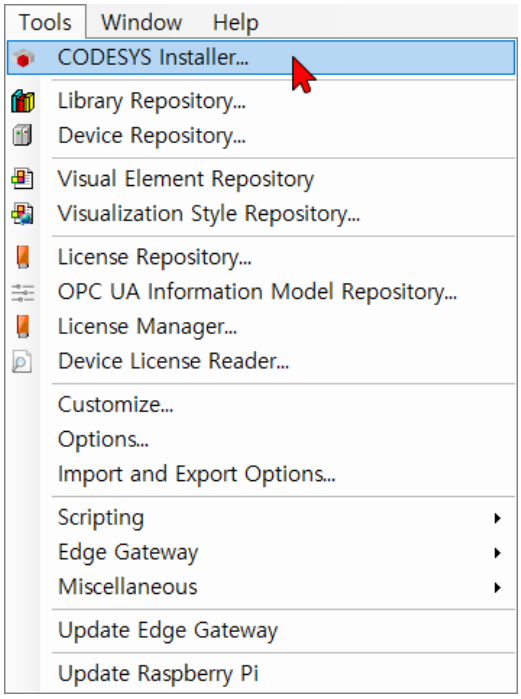
4. How to use CoDeSys Version (GL-9981-C)

GL-9981 boot and check the IOS LED & Check internal bus status

- Turn on the power of GL-9981 to boot.
- After booting is complete, the 'BUS Scan process' program is automatically executed.
- In Codesys, you can check that the internal bus status is RUN and write a user program to control IO.

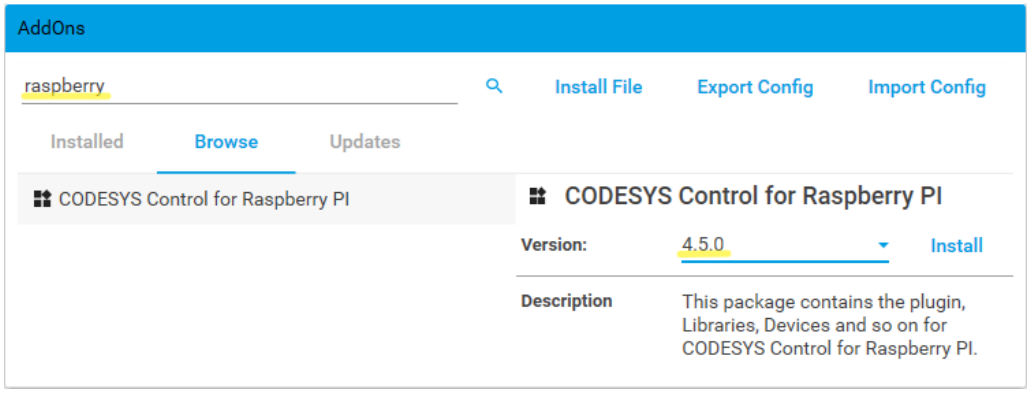
Install CoDeSys runtime

1. Open 'CODESYS' on the PC to be connected to the Raspberry. Click 'Tools - CODESYS Installer'.



2. After selecting Browse in the AddOns area of the CODESYS Installer window, search for 'raspberry'.

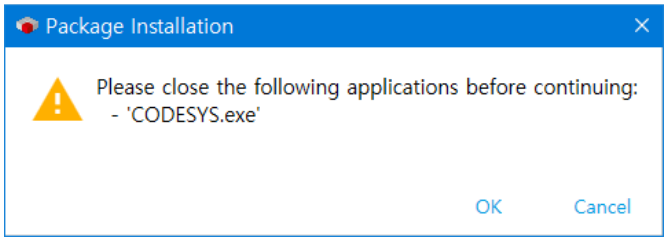
Select 'CODESYS Control for Raspberry PI' and press 'Install' with the desired version.



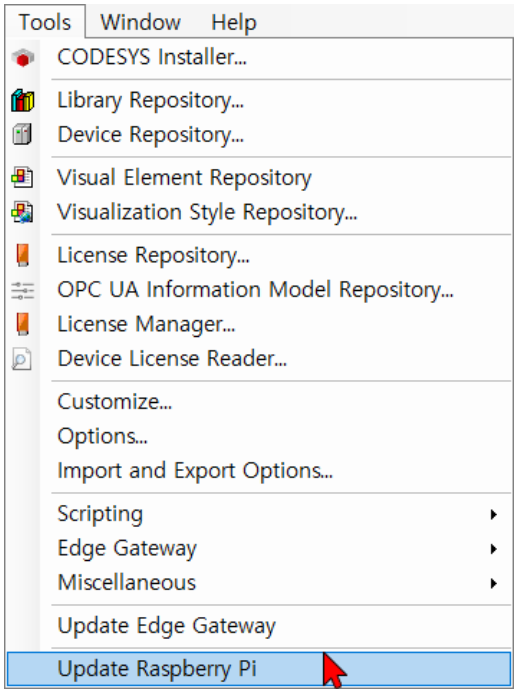
3. In the next step, check 'I accept the license agrreement' and click 'Continue'.

4. Click 'OK' after terminating the running CODESYS.

When the installation is complete, the 'Tools - Update Raspberry Pi' menu appears.



5. Run Codesys again and click 'Tools - Update Raspberry Pi'.

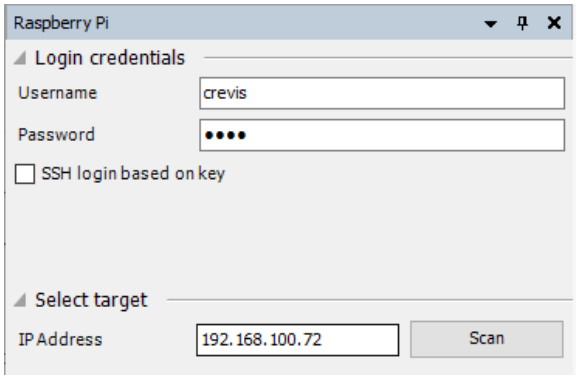


6. Enter the information of Login credentials and Select target.

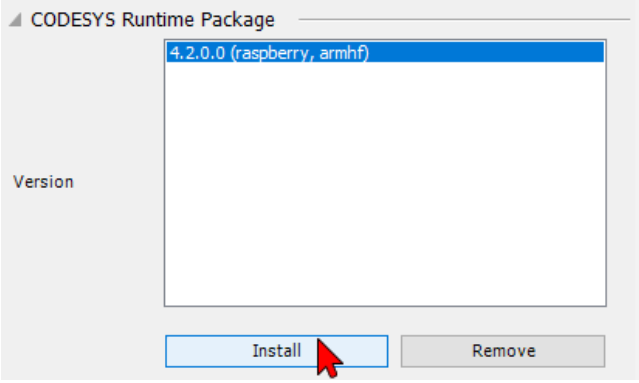
(The Raspberry Pi and PC to be connected must have the same IP band.)

(default)

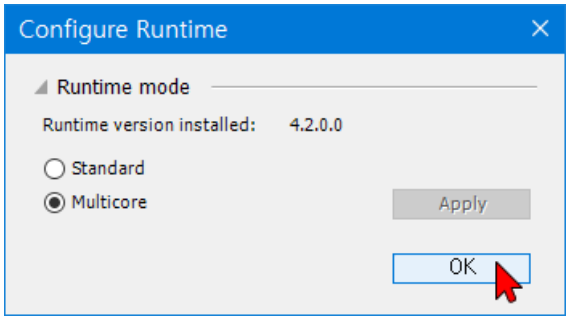
- Username: crevis / Password: user
- IPAddress : 192.168.100.72



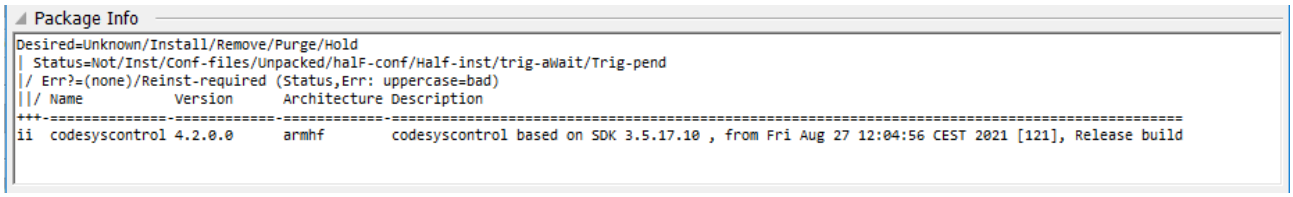
7. Select the version of the installed CODESYS Runtime Package and click 'Install'.



8. Select 'Multicore' and click 'OK'.

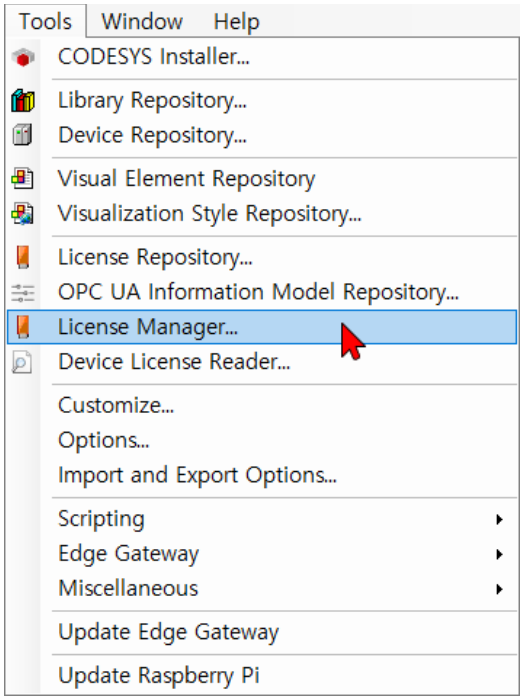


9. If you click 'system Info', you can check the installation information in 'Package Info'.

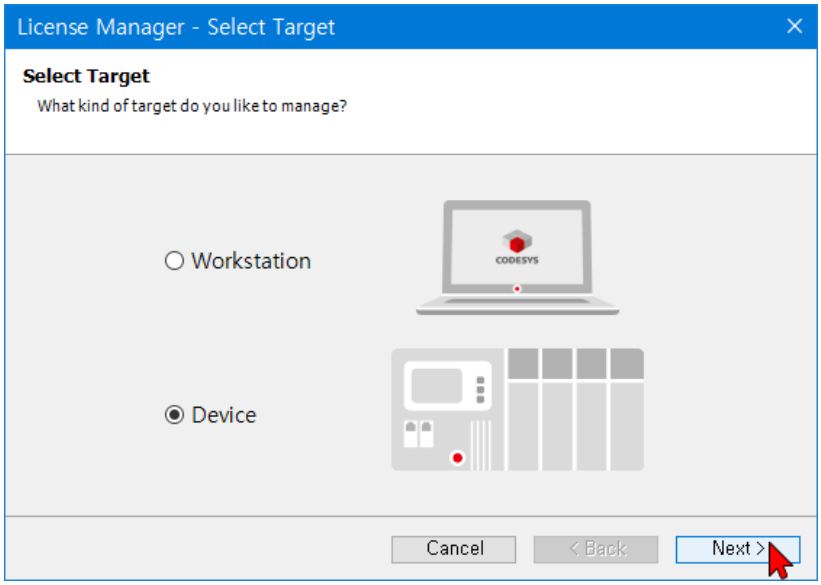


CODESYS License Activation

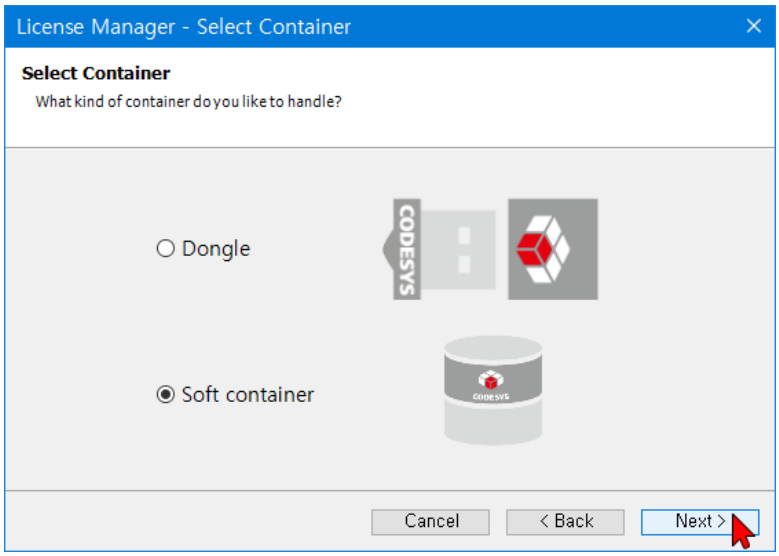
1. Click 'Tools - License Manager'.



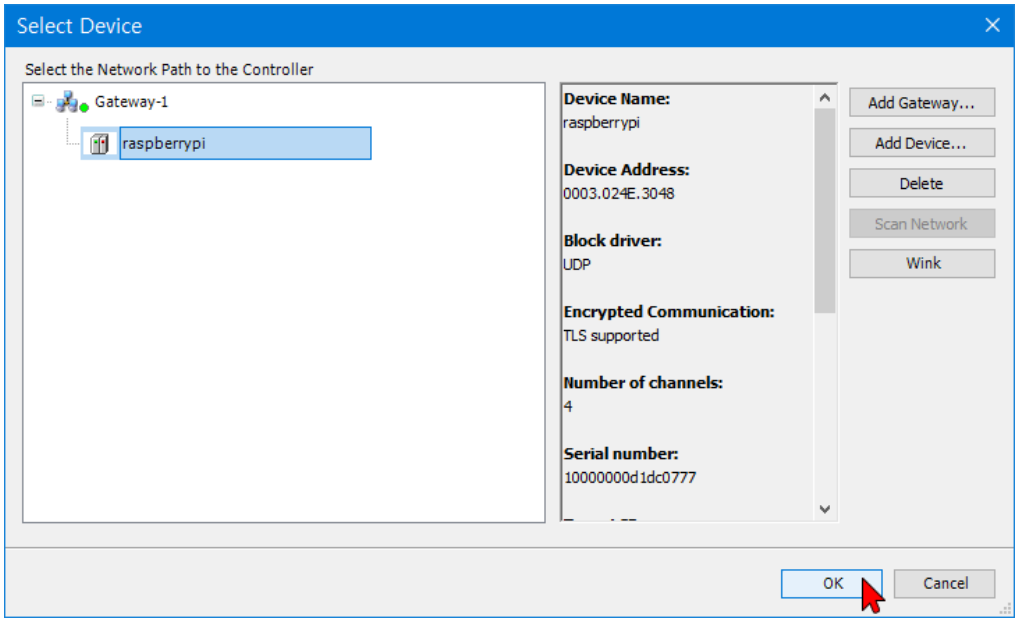
2. Select 'device' and click 'Next'.



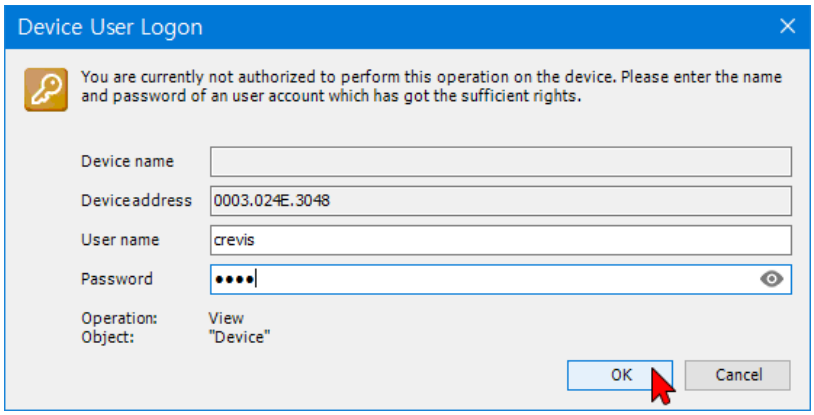
3. Select 'Soft conrainer' and click 'Next'.



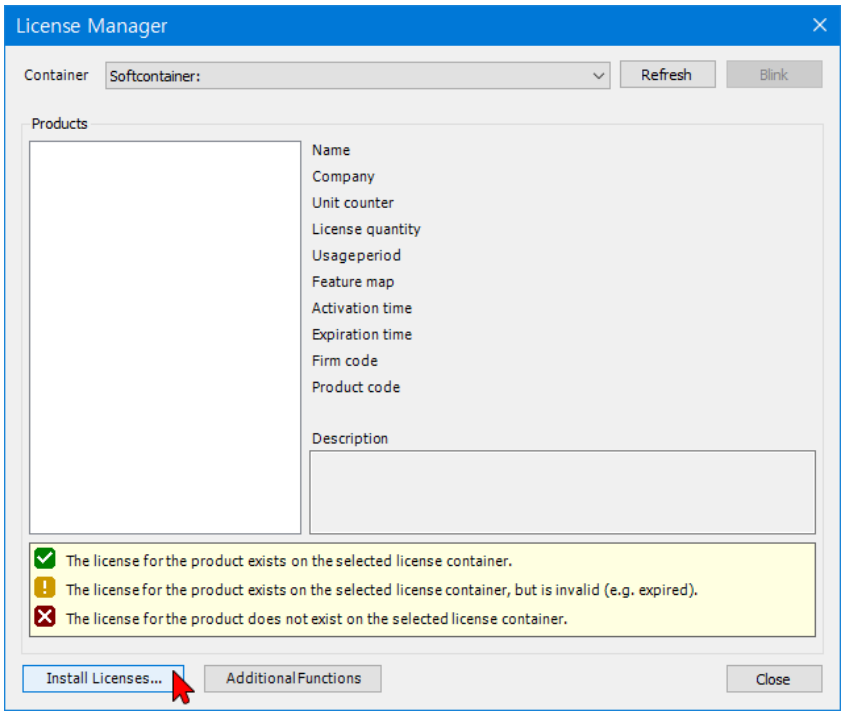
4. After selecting the scanned 'Raspberry Pi' click 'OK.



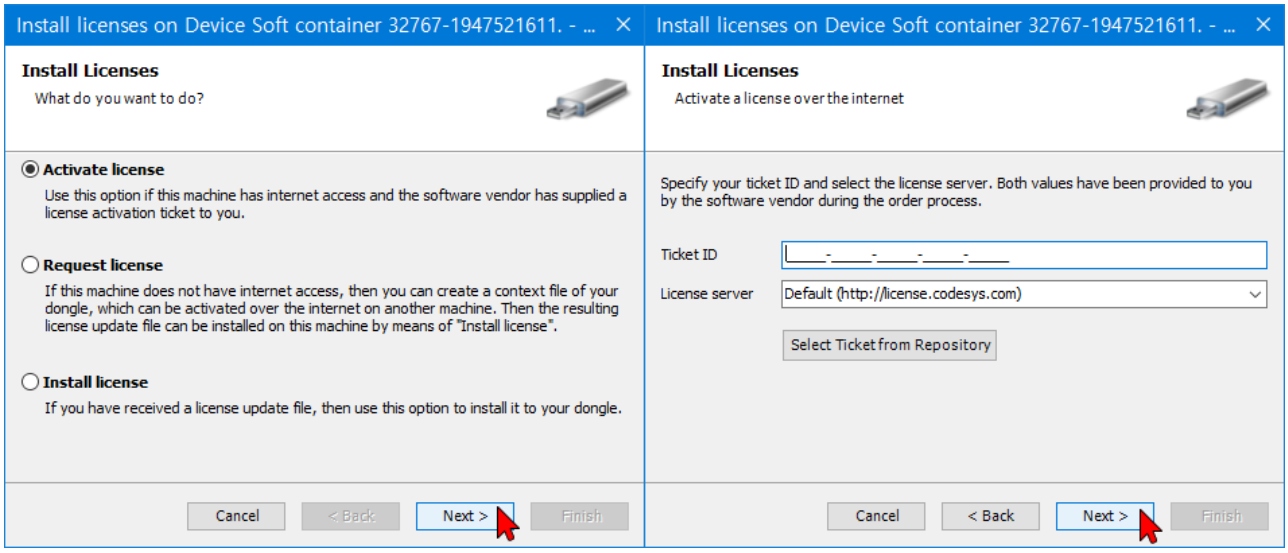
5. Enter your user name and password and click 'OK'.



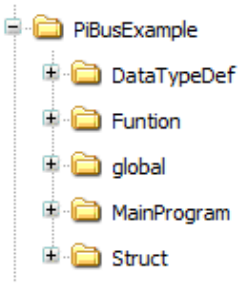
6. Click 'install licenses...'



7. Select activate license and click 'Next'. Enter Ticket ID to complete the installation.



Provides a way to control IO with Codesys project example file



- DataTypeDef : Datatype definition.
- Funtion : Functions to be used for IO control.
- global : Global variable.
- MainProgram : The main program to be actually driven.
- Struct : Definition of structure used in internal bus.

Functions used for IO control (Funtion folder)

- pibus_Control

pibusInit	Initialize semaphores and shared memory Confirm that the internal bus status is run and proceed to the next step VAR_INPUT - none VAR_OUTPUT - execution result
Read_InputImage	Read values from input image VAR_INPUT - Addr: starting address / pBuffer: use buffer / Len: length VAR_OUTPUT - execution result
Read_OutputImage	Read values from output image VAR_INPUT - Addr: starting address / pBuffer: use buffer / Len: length VAR_OUTPUT - execution result
Write_OutputImage	Write values from output image VAR_INPUT - Addr: starting address / pBuffer: use buffer / Len: length VAR_OUTPUT - execution result
Read_Slot * Input module only	Read the value of the slot VAR_INPUT - slot: slot number / pBuffer: use buffer VAR_OUTPUT - execution result
Write_Slot * Output module only	Writes the value of the slot VAR_INPUT - slot: slot number / pBuffer: use buffer VAR_OUTPUT - execution result
Get_Prm	Read the parameters of the slot VAR_INPUT - slot: slot number / pBuffer: use buffer VAR_OUTPUT - execution result
Set_Prm	Write the parameters of the slot VAR_INPUT - slot: slot number / pBuffer: use buffer VAR_OUTPUT - execution result

- sem_Control

semInit	Initialize the semaphore to use VAR_OUTPUT - execution result
semEnter	Using semaphore
semLeave	Return semaphore

- shm_Control

shmInit	Initialize shared memory VAR_OUTPUT - execution result
shmRead_X	Read shared memory from handler to buffer VAR_OUTPUT - execution result
shmWrite_X	The value stored in the buffer is written to the shared memory using a handler VAR_OUTPUT - execution result

function test example program (MainProgram folder)

- This is an example of manually selecting and executing the desired test type after initialization.

Device.Application.pibus_function_test

Expression	Type	Value
init	INT	255
test_selet	INT	0
test_time	INT	0
slot	BYTE	0
⊞ io_rw_byte	ARRAY [0..255] OF u8	
addr	WORD	0
len	BYTE	0

(Refer to the 'Pibus_GL9981_Sample' project for the program)

1. If initialization proceeds normally, the **init** variable has a value of 255.
 - It is possible to test only when initialization is performed normally.

2. If the desired test is write, the value to be written is stored in advance in the **io_rw_byte** buffer.

3. Also set the variable **slot** or **len** or **addr** to the desired value.
 - slot: slot number / len: length / addr: starting address

4. Select the desired test with the **test_selet** variable.

5. When the test is executed, the **test_time** variable has a value of 1 to prevent repetition.