

Table of Contents

- [1. General](#)
- [2. Description](#)
- [3. Cross compile of open source library for AXC F 2152](#)
 - [3.1. Setup of Linux VM](#)
 - [3.2. Get open source libmodbus](#)
 - [3.3. Compile](#)
- [4. Eclipse](#)
- [5. Prepare your AXC F 2152](#)
 - [5.1. Check your controller firmware](#)
 - [5.2. Transfer the lib.tar file](#)
 - [5.3. Set your library path with ldconfig.](#)
- [6. PC Worx Engineer](#)
 - [Read registers](#)
 - [Write registers](#)
 - [Init error](#)
 - [Read error](#)
 - [Write error](#)

1. General

This is an example for the use of the open source C++ library Libmodbus for a Modbus TCP communication.

Name:	EX_CPP_LibmodbusTCP_2
Type:	Example of use
IDE:	PC Worx Engineer 7.2.1 (Build 3.11.104.0)
Used languages:	ST, C++
Used libraries:	PCWE_LibmodbusTCP, CPP_LibmodbusTCP
Included external C++ libraries:	Libmodbus
Controller:	AXC F 2152
Controller firmware:	1.0.1.8
Modules:	/

2. Description

In the project there are two libraries, one C++ library (**CPP_LibmodbusTCP.pcwlx**) and one PCWE library (**PCWE_LibmodbusTCP.pcwlx**).

PCWE_LibmodbusTCP just consists of two in- and two outputs. These are used in the project to control the CPP-program and show the results.

CPP_LibmodbusTCP does the modbus part. It supports reading and writing the first 32 registers of the Modbus slave. In **CPP_LibmodbusTCP**, the Modbus slave IP address and the port are static. If necessary, the IP address can be changed in the function `CPP_LibmodbusTCP_P::Execute()`.

3. Cross compile of open source library for AXC F 2152

3.1. Setup of Linux VM

First of all, you have to set up a Linux VM (e.g. Xubuntu 16.04.01 or Ubuntu 16.04) and install the PLCnext Technology SDK. You can download and find a tutorial of the installation here : https://www.plcnext-community.net/index.php?option=com_content&view=article&id=75&catid=45&Itemid=263&lang=en

Important: It has to be a Linux VM. This is needed for cross compiling the open source library libmodbus.

3.2. Get open source libmodbus

After the installation, you can visit <http://libmodbus.org> and download the libmodbus version 3.0.6. Copy the downloaded **libmodbus-3.0.6.tar.gz** file in the Linux VM. Execute the command **tar -xvf .tar.gz** to decompress.

libmodbus data from internet:

```
ccax-team-user@ccaxteam-plcnext-vm:~/work$ ls -l
total 500
drwxr-xr-x 6 ccax-team-user ccax-team-user 4096 Jan xx 11:06 libmodbus-3.0.6
-rw----- 1 ccax-team-user ccax-team-user 504757 Jan xx 10:09 libmodbus-3.0.6.tar.gz
```

3.3. Compile

Then you copy the file **SetupConfig.tar** from **EX_CPP_LibmodbusTCP_2** into the same directory:

```
ccax-team-user@ccaxteam-plcnnext-vm:~/work/libmodbus-3.0.6$ ls -l SetupConfig.tar
-rw-rw-r-- 1 ccax-team-user ccax-team-user 10240 Jan xx 11:06 SetupConfig.tar
```

Unpack **SetupConfig.tar** with **tar -x SetupConfig.tar**. You get the executable Linux script SetupConfig.sh.

```
ccax-team-user@ccaxteam-plcnnext-vm:~/work/libmodbus-3.0.6$ ls -l SetupConfig.sh
-rwxrwxrwx 1 ccax-team-user ccax-team-user 521 Jan xx 14:07 SetupConfig.sh
```

Execute the SetupConfig.sh with the command
./SetupConfig.sh.

The log of the execution of **SetupConfig.sh** should look like
SetupConfig.log.

Important: If you directly copy a script from Windows to Linux, you have to convert it with **dos2unix SetupConfig.sh** and make it executable with **chmod 777 SetupConfig.sh**. Otherwise it will not work.

Now you should find the compiled files in **/opt/pxc/install/**.

```
ccax-team-user@ccaxteam-plcnnext-vm:/opt/pxc/install/lib$ ls -l
total 88
-rw-r--r-- 1 ccax-team-user ccax-team-user 39896 Jan xx 09:39 libmodbus.a
-rwxr-xr-x 1 ccax-team-user ccax-team-user 960 Jan xx 09:39 libmodbus.la
lrwxrwxrwx 1 ccax-team-user ccax-team-user 18 Jan xx 09:39 libmodbus.so -> libmodbus.so.5.0.5
lrwxrwxrwx 1 ccax-team-user ccax-team-user 18 Jan xx 09:39 libmodbus.so.5 -> libmodbus.so.5.0.5
-rwxr-xr-x 1 ccax-team-user ccax-team-user 40296 Jan xx 09:39 libmodbus.so.5.0.5
drwxrwxr-x 2 ccax-team-user ccax-team-user 4096 Jan xx 09:39 pkgconfig
```

Create a .tar file of the lib folder. Run the command : **tar -cf lib.tar ***.

```
ccax-team-user@ccaxteam-plcnnext-vm:/opt/pxc/install/lib$ ls -l lib.tar
-rw-rw-r-- 1 ccax-team-user ccax-team-user 92160 Jan xx 09:48 lib.tar
```

lib.tar needs to be copied to the AXC F 2152 later.

4. Eclipse

This is an example environment. All environment settings in this example are made according to this.

```
c:\
[EX_CPP]: C++ example root directory
[CPP_LibmodbusTCP]: C++ example
[libs_external]: external libraries, e.g. open source
[libmodbus-3.0.6] : Original data downloaded from internet
[src]
    Makefile.am
    Makefile.in
    modbus.c
    modbus.h
    ...
[libs_pcwe]: All PC Worx Engineer libraries are created in this directory
[sysroots]: PLCnext system root directory
```

Copy the compiled data from your Linux VM:

```
c:\
[EX_CPP]
...
[libs_external] :
[libmodbus-3.0.6] :

[lib]: Directory for cross compiled library.
Create this directory manually.
    pkgconfig\
    libmodbus.a
    libmodbus.la
    libmodbus.so
    libmodbus.so.5
```

libmodbus.so.5.0.5

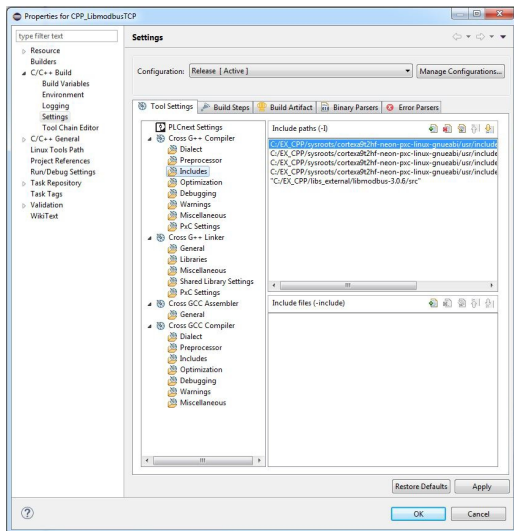
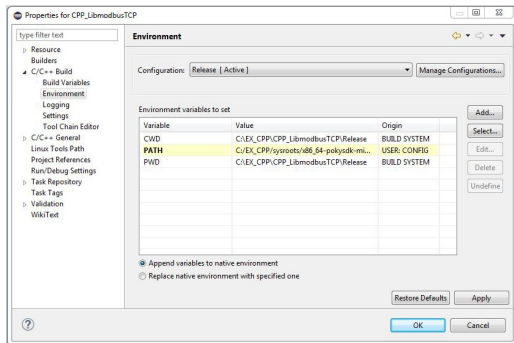
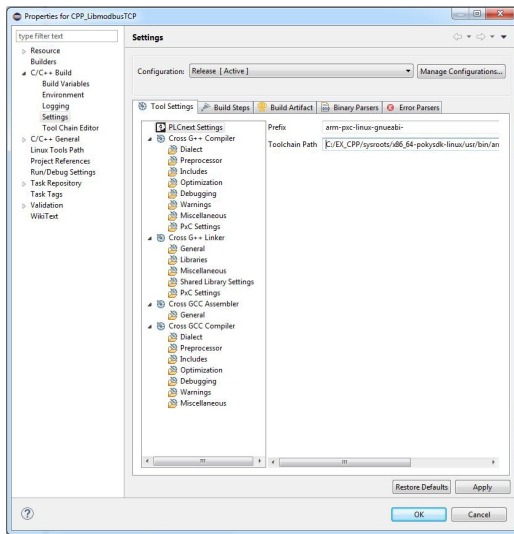
...

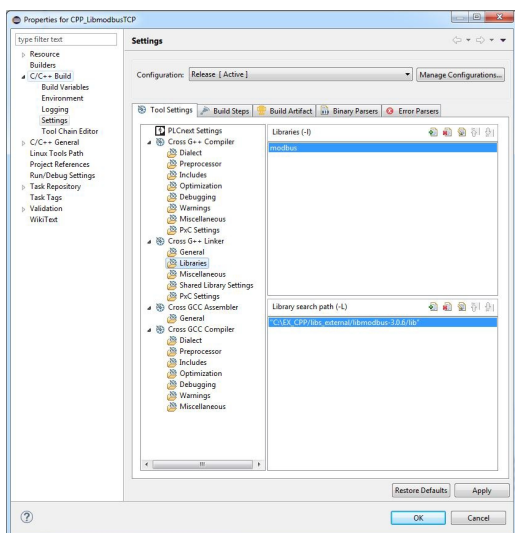
...

...

Start Eclipse and load the project CPP_LibmodbusTCP.

Setup according to your environment (here: our environment mentioned above):





Build your project.

```
15:51:48 **** Rebuild of configuration Release for project CPP_LibmodbusTCP ****  
Info: Internal Builder is used for build  
arm-pxc-linux-gnueabi-g++ -std=c++11 -IC:/EX_CPP/sysroots/cortexa9t2hf-neon-pxc-linux-gnueabi/usr/incl  
arm-pxc-linux-gnueabi-g++ -std=c++11 -IC:/EX_CPP/sysroots/cortexa9t2hf-neon-pxc-linux-gnueabi/usr/incl  
arm-pxc-linux-gnueabi-g++ -std=c++11 -IC:/EX_CPP/sysroots/cortexa9t2hf-neon-pxc-linux-gnueabi/usr/incl  
arm-pxc-linux-gnueabi-g++ -std=c++11 -IC:/EX_CPP/sysroots/cortexa9t2hf-neon-pxc-linux-gnueabi/usr/incl  
arm-pxc-linux-gnueabi-g++ "-LC:\\EX_CPP\\libs_external\\libmodbus-3.0.6/lib" -shared -march=armv7-a ...  
15:52:33 Build Finished (took 44s.848ms)
```

Check if the **libCPP LibmodbusTCP.so** file is created.

```
C:\EX_CPP\CPP_Libmodbus\Release>dir
.
.
dd.mm.yy  15:52                106.488 libCPP_LibmodbusTCP.so
.
.
```

Create the PC Worx Engineer library.

Check if the ***CPP_LibmodbusTCP.pcwlx*** file is created.

```
C:\EX_CPP\libs_pcwe>dir
...
dd.mm.yy  16:22                47.870 CPP_LibmodbusTCP.pcw1x
...
```

5. Prepare your AXC F 2152

5.1. Check your controller firmware

```
admin@axcf2152:/etc/plcnext$ cat arpversion
Arpversion: 1.0.1.8
Svn Revision: 7585
Build Job: "jenkins-Arp-AXCF2152_1.0.1-8"
```

5.2. Transfer the lib.tar file

Transfer the **lib.tar** file you have created in your Linux environment to the AXC F 2152 to the directory **/opt/plcnext/projects/Default/Libs**.

Suggestion: Use WinSCP.

Now you can open Putty and connect to the controller. Navigate to **/opt/plcnext/projects/Default/Libs** and execute the command **tar -xf lib.tar**.

Check:

```
admin@axcf2152:/opt/plcnext/projects/Default/Libs$ ls -l
total 212
-rw-rw-r-- 1 admin plcnext 92160 Jan xx 12:17 lib.tar
-rw-r--r-- 1 admin plcnext 39896 Jan xx 13:17 libmodbus.a
-rwxr-xr-x 1 admin plcnext  960 Jan xx 13:17 libmodbus.la
lrwxrwxrwx 1 admin plcnext  18 Jan xx 13:17 libmodbus.so -> libmodbus.so.5.0.5
lrwxrwxrwx 1 admin plcnext  18 Jan xx 13:17 libmodbus.so.5 -> libmodbus.so.5.0.5
-rwxr-xr-x 1 admin plcnext 40296 Jan xx 13:17 libmodbus.so.5.0.5
drwxrwxr-x 2 admin plcnext  4096 Jan xx 13:17 pkgconfig
```

5.3. Set your library path with ldconfig.

Login and change to root

```
login as: admin
admin@10.250.65.10's password:
Last login: Fri Jan xx 10:41:59 2018 from 10.250.65.1
admin@axcf2152:~$ cd /etc
admin@axcf2152:/etc$ sudo passwd root
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
Password:
Changing password for root
Enter the new password (minimum of 5 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
Re-enter new password:
passwd: password changed.
admin@axcf2152:/etc$ su root
Password:
```

Add **/opt/plcnext/projects/Default/Libs** to your library search path.

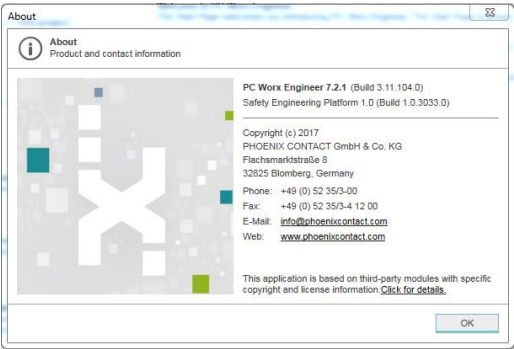
```
root@axcf2152:/etc# vi ld.so.conf
```

Check

```
root@axcf2152:/etc# cat ld.so.conf
/usr/local/lib
/opt/plcnext/projects/Default/Libs
root@axcf2152:/etc# ldconfig
root@axcf2152:/etc# ldconfig -v | grep modbus
libmodbus.so.5 -> libmodbus.so.5.0.5
```

6. PC Worx Engineer

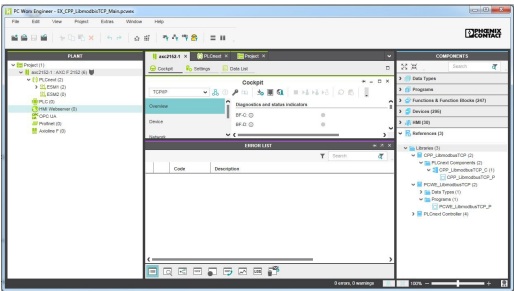
Check your PC Worx Engineer version:



Copy **EX_CPP_LibmodbusTCP.pcwex** to the PC Worx Engineer project directory,
e.g. **c:\Users\Public\Documents\PC Worx Engineer\Projects**

Copy **CPP_LibmodbusTCP.pcwlx** and **PCWE_LibmodbusTCP.pcwlx** to the PC Worx Engineer library directory,
e.g. **c:\Users\Public\Documents\PC Worx Engineer\Libraries**

Open **EX_CPP_LibmodbusTCP.pcwex** in PC Worx Engineer.



Connect the ports:

OUT Port	IN Port
Arp.Plc.Eclr / PCWE_LibmodbusTCP : OP_xExecute	CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP : IP_Execute
Arp.Plc.Eclr / PCWE_LibmodbusTCP : OP_usiOperationMode	CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP : IP_OperationMode
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP : OP_RegArray	Arp.Plc.Eclr / PCWE_LibmodbusTCP : IP_arrData
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP : OP_State	Arp.Plc.Eclr / PCWE_LibmodbusTCP : IP_uiState
Select OUT Port here	

Set the operation mode:

Then you can set the operationmode via the port OP_usiOperationMode.

OP_usiOperationMode	Effect
1	read first 32 registers.
2	write first 32 registers (values 0 to 31)
all others	no operation

Start the program:

After setting OP_usiOperationMode, you can start the routine by setting OP_xExecute to TRUE. To start the program again, you have to set OP_xExecute first to FALSE, then to TRUE again.

Read registers

OUT Port	Value	Type
Arp.Plc.Eclr / PCWE_LibmodbusTCP_P1 : OP_usiOperationMode	1	USINT
Arp.Plc.Eclr / PCWE_LibmodbusTCP_P1 : OP_xExecute	TRUE	BOOL
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_RegArray		uint16[32]
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_State	510	uint16
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_Version	2	uint16

In our case we have set the register values as follows:

Name	Value	Data Type
OP_RegArray		uint16[32]
OP_RegArray[0]	0	uint16
OP_RegArray[1]	1	uint16
OP_RegArray[2]	2	uint16
OP_RegArray[3]	3	uint16
OP_RegArray[4]	4	uint16
OP_RegArray[5]	5	uint16
OP_RegArray[6]	6	uint16
OP_RegArray[7]	7	uint16
OP_RegArray[8]	8	uint16

Output.log:

```
dd.mm.yy 17:06:23.828 root INFO - EX_CPP: 0: CPP_LibmodbusTCP_2.
dd.mm.yy 17:06:24.328 root INFO - EX_CPP: 100: Initialize Modbus.
dd.mm.yy 17:06:24.328 root INFO - EX_CPP: 100: Valid Modbus.
dd.mm.yy 17:06:24.828 root INFO - EX_CPP: 200: Modbus connect.
dd.mm.yy 17:06:24.834 root INFO - EX_CPP: 200: modbus_connect = 0.
dd.mm.yy 17:06:24.834 root INFO - EX_CPP: 200: Valid Connection.
dd.mm.yy 17:06:25.328 root INFO - EX_CPP: 300: Read Registers.
dd.mm.yy 17:06:25.828 root INFO - EX_CPP: 500: Connection closed.
```

Write registers

OUT Port	Value	Type
Arp.Plc.Eclr / PCWE_LibmodbusTCP_P1 : OP_usiOperationMode	2	USINT
Arp.Plc.Eclr / PCWE_LibmodbusTCP_P1 : OP_xExecute	TRUE	BOOL
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_RegArray		uint16[32]
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_State	510	uint16
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_Version	2	uint16

Output.log:

```
dd.mm.yy 17:03:20.328 root INFO - EX_CPP: 0: CPP_LibmodbusTCP_2.
dd.mm.yy 17:03:20.828 root INFO - EX_CPP: 100: Initialize Modbus.
dd.mm.yy 17:03:20.829 root INFO - EX_CPP: 100: Valid Modbus.
dd.mm.yy 17:03:21.328 root INFO - EX_CPP: 200: Modbus connect.
dd.mm.yy 17:03:21.333 root INFO - EX_CPP: 200: modbus_connect = 0.
dd.mm.yy 17:03:21.333 root INFO - EX_CPP: 200: Valid Connection.
dd.mm.yy 17:03:21.828 root INFO - EX_CPP: 400: Write Registers.
dd.mm.yy 17:03:22.328 root INFO - EX_CPP: 500: Connection closed.
```

Init error

If you do not have a modbus server, you will get an connect error.

Output.log:

```
dd.mm.yy 17:07:51.328 root INFO - EX_CPP: 0: CPP_LibmodbusTCP_2.
dd.mm.yy 17:07:51.828 root INFO - EX_CPP: 100: Initialize Modbus.
dd.mm.yy 17:07:51.828 root INFO - EX_CPP: 100: Valid Modbus.
dd.mm.yy 17:07:52.328 root INFO - EX_CPP: 200: Modbus connect.
dd.mm.yy 17:07:52.333 root INFO - EX_CPP: 200: modbus_connect = -1.
dd.mm.yy 17:07:52.333 root INFO - EX_CPP: 200: Invalid Connection.
dd.mm.yy 17:07:52.828 root INFO - EX_CPP: 298: Connect error.
```

Read error

If your modbus server does not answer fast enough during read you will get a read timeout error.

OUT Port	Value	Type
Arp.Plc.Eclr / PCWE_LibmodbusTCP_P1 : OP_usiOperationMode	1	USINT
Arp.Plc.Eclr / PCWE_LibmodbusTCP_P1 : OP_xExecute	TRUE	BOOL
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_RegArray		uint16[32]
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_State	399	uint16
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_Version	2	uint16

Output.log:

```
dd.mm.yy 17:11:35.328 root INFO - EX_CPP: 0: CPP_LibmodbusTCP_2.
dd.mm.yy 17:11:35.828 root INFO - EX_CPP: 100: Initialize Modbus.
dd.mm.yy 17:11:35.828 root INFO - EX_CPP: 100: Valid Modbus.
dd.mm.yy 17:11:36.328 root INFO - EX_CPP: 200: Modbus connect.
dd.mm.yy 17:11:36.333 root INFO - EX_CPP: 200: modbus_connect = 0.
dd.mm.yy 17:11:36.333 root INFO - EX_CPP: 200: Valid Connection.
dd.mm.yy 17:11:36.828 root INFO - EX_CPP: 300: Read Registers.
dd.mm.yy 17:11:36.879 root INFO - EX_CPP: 300: Read failed = Connection timed out.
dd.mm.yy 17:11:37.328 root INFO - EX_CPP: 398: Read error.
```

Write error

If your modbus server does not answer fast enough during write you will get a write timeout error.

OUT Port	Value	Type
Arp.Plc.Eclr / PCWE_LibmodbusTCP_P1 : OP_usiOperationMode	2	USINT
Arp.Plc.Eclr / PCWE_LibmodbusTCP_P1 : OP_xExecute	TRUE	BOOL
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_RegArray		uint16[32]
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_State	499	uint16
CPP_LibmodbusTCP.CPP_LibmodbusTCP_C-1 / CPP_LibmodbusTCP_P1 : OP_Version	2	uint16

Output.log:

```
dd.mm.yy 17:12:50.828 root INFO - EX_CPP: 0: CPP_LibmodbusTCP_2.
dd.mm.yy 17:12:51.328 root INFO - EX_CPP: 100: Initialize Modbus.
dd.mm.yy 17:12:51.328 root INFO - EX_CPP: 100: Valid Modbus.
dd.mm.yy 17:12:51.828 root INFO - EX_CPP: 200: Modbus connect.
dd.mm.yy 17:12:51.832 root INFO - EX_CPP: 200: modbus_connect = 0.
dd.mm.yy 17:12:51.832 root INFO - EX_CPP: 200: Valid Connection.
dd.mm.yy 17:12:52.328 root INFO - EX_CPP: 400: Write Registers.
dd.mm.yy 17:12:52.379 root INFO - EX_CPP: 400: Write failed = Connection timed out.
dd.mm.yy 17:12:52.828 root INFO - EX_CPP: 498: Write error.
```