The University of Alabama in Huntsville

~ 08/29/2019 ~

Testing Method – OpenPLC

A list and description of how to test Thurst 2: Open PLC Project.

- **Unconnected**
  - *Future Work - Develop tool for testing this type*
- **Connected**
  - *Analog*
    - Using AdvancedHMI, we are creating a simulated blinking LED and changing the frequency of the blinking LED using buttons on AdvancedHMI.
  - *Digital*
    - Using AdvancedHMI, we are creating a simulated blinking LED (0 – OFF 1 – ON)
- **Unknown**
  - *Ethersploit -> send_raw_pccc [data]*
    - Using Cygwin, run this exploit tool just sends raw data to the PLC.
      - *./ethersploit*
      - *Connect 127.0.0.1*
      - *Send_raw_pcc [data]*

***Programs and Tools Used for Testing:**

- *OpenPLC Runtime – Virtual PLC*
- *Cygwin – Application that allows Linux native apps on Windows*
- *Wireshark – Packet capture tool to verify contents are sending and receiving*
- *AdvancedHMI – HMI that was used to test usability and application*
  - *Replace the folder "AdvancedHMI" with the folder from Google Drive*
  - https://drive.google.com/drive/folders/1tPYGuvh1GNW0lMSPyFzmMyou3HzI3V4q
  - Select the form in AdvancedHMI from the "AllTest" folder

The University of Alabama in Huntsville

~ 08/29/2019 ~

AdvancedHMI Tests

A description of the successfully tested functions and how to set up AdvancedHMI in order to test those functions. All testing and verification can be done with the ladder logic file "alltest3.st" and with the single Advanced HMI setup provided. Wireshark is also used to verify the correct format of the packets.

Functions

PCCC supports three different overarching functions. Protected Logical Read is used to read data from memory. Protected Logical Write is used to write byte size data into memory. Protected Logical Write with Mask is used to write bitwise data into memory.

File Types & Numbers

The PCCC protocol stores data in sections of memory blocked off in "files". Each of these sections has a corresponding File Type and File Number. The four file types implemented in OpenPLC are Input Logical by Slot, Output Logical by Slot, Integer, and Float.

Read Digital Output

Description of Test: The PLC returns a binary value from the digital output memory.

Function Code 0xA2 **File Type** 0x8B **File Number** 0x00

Ladder Logic File: alltest.st

AdvancedHMI Interface Components List:

- Driver – EthernetIPforSLCMicroCom1
- Basic Indicator – Reads from the Address either a 1 or 0
- Basic Button ON – Writes a 1
- Basic Button OFF – Writes a 0

Drive Location: https://drive.google.com/drive/folders/1ouMFUaN8jH8vUa8VN8xtN5qIWeaL2NJB

Read Digital Input

Description of Test: The PLC returns a binary value from the digital input memory.

Function Code 0xA2 **File Type** 0x8C **File Number** 0x01

Ladder Logic File: alltest.st

AdvancedHMI Interface Components List:

- Driver – EthernetIPforSLCMicroCom1
- Basic Indicator – Reads from the Address either a 1 or 0
- Basic Button ON – Writes a 1
- Basic Button OFF – Writes a 0

Drive Location: https://drive.google.com/drive/folders/1ouMFUaN8jH8vUa8VN8xtN5qIWeaL2NJB

Read Integer

Description of Test for Analog: The PLC return an integer value from memory.

Function Code 0xA2 **File Type** 0x89 **File Number** 0x07

Ladder Logic File: alltest.st

AdvancedHMI Interface Components List: Integer (Bottom Four Labels)

- Driver – EthernetIPforSLCMicroCom1
- Basic Label – Keypad -> Double Click Label "Int1" (Writes)
- Basic Label – Keypad -> Double Click Label "Int2" (Writes)
- Basic Label – Number Display Above Int1 -> Int1 Value (Reads)
- Basic Label – Number Display Above Int2-> Int2 Value (Reads)

Drive Location: https://drive.google.com/drive/folders/1ouMFUaN8jH8vUa8VN8xtN5qIWeaL2NJB

Read Float

Description of Test for Analog: The PLC return a floatr value from memory.

Function Code 0xA2 **File Type** 0x8A **File Number** 0x08

Ladder Logic File: alltest.st

AdvancedHMI Interface Components List: Floating Point (Top Three Labels)

- Driver – EthernetIPforSLCMicroCom1
- Basic Label – Keypad -> Double Click Label "Float1" (Writes)
- Basic Label – Number Display Top Left -> Float1 Value (Reads)
- Basic Label – Number Display Top Right -> Float1 Value + 10.0 (Reads)

Drive Location: https://drive.google.com/drive/folders/1ouMFUaN8jH8vUa8VN8xtN5qIWeaL2NJB

Write Digital Output

Description of Test: The PLC stores a binary value into digital output memory.

Function Code 0xAB **File Type** 0xAB **File Number** 0x00

Ladder Logic File: alltest.st

AdvancedHMI Interface Components List: analogint.st

- Driver – EthernetIPforSLCMicroCom1
- Basic Indicator – Reads from the Address either a 1 or 0
- Basic Button ON – Writes a 1
- Basic Button OFF – Writes a 0

Drive Location: https://drive.google.com/drive/folders/1ouMFUaN8jH8vUa8VN8xtN5qIWeaL2NJB

Write Integer

Description of Test for Analog: The PLC stores an integer value into memory.

Function Code 0xAA **File Type** 0x89 **File Number** 0x07

Ladder Logic File: alltest.st

AdvancedHMI Interface Components List: Integer (Bottom Four Labels)

- Driver – EthernetIPforSLCMicroCom1
- Basic Label – Keypad -> Double Click Label "Int1" (Writes)
- Basic Label – Keypad -> Double Click Label "Int2" (Writes)
- Basic Label – Number Display Above Int1 -> Int1 Value (Reads)
- Basic Label – Number Display Above Int2-> Int2 Value (Reads)

Drive Location: https://drive.google.com/drive/folders/1ouMFUaN8jH8vUa8VN8xtN5qIWeaL2NJB

Write Float

Description of Test for Analog: The PLC stores a float value into memory.

Function Code 0xA2 **File Type** 0x8A **File Number** 0x08

Ladder Logic File: alltest.st

AdvancedHMI Interface Components List: Floating Point (Top Three Labels)

- Driver – EthernetIPforSLCMicroCom1
- Basic Label – Keypad -> Double Click Label "Float1" (Writes)
- Basic Label – Number Display Top Left -> Float1 Value (Reads)
- Basic Label – Number Display Top Right -> Float1 Value + 10.0 (Reads)

Drive Location: https://drive.google.com/drive/folders/1ouMFUaN8jH8vUa8VN8xtN5qIWeaL2NJB

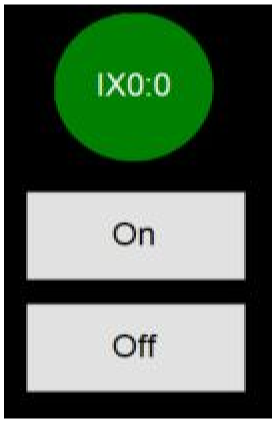
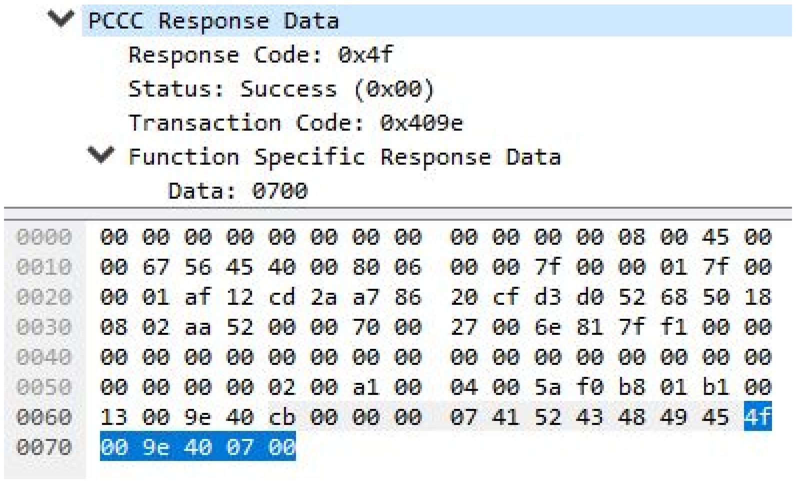


Figure 1: Reading Digital Inputs & Outputs with LED Indicators. Writing Digital Outputs with push buttons



```
PCCC Response Data
    Response Code: 0x4f
    Status: Success (0x00)
    Transaction Code: 0x409e
  Function Specific Response Data
      Data: 0700

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00
0010  00 67 56 45 40 00 80 06  00 00 7f 00 00 01 7f 00
0020  00 01 af 12 cd 2a a7 86  20 cf d3 d0 52 68 50 18
0030  08 02 aa 52 00 00 70 00  27 00 6e 81 7f f1 00 00
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0050  00 00 00 00 02 00 a1 00  04 00 5a f0 b8 01 b1 00
0060  13 00 9e 40 cb 00 00 00  07 41 52 43 48 49 45 4f
0070  00 9e 40 07 00
```

Figure 2: Response to a Read Input Logical by Slot request for 2 bytes. The first 3 least significant bits are all set to 1. (Notice the little endian format of the two bytes)

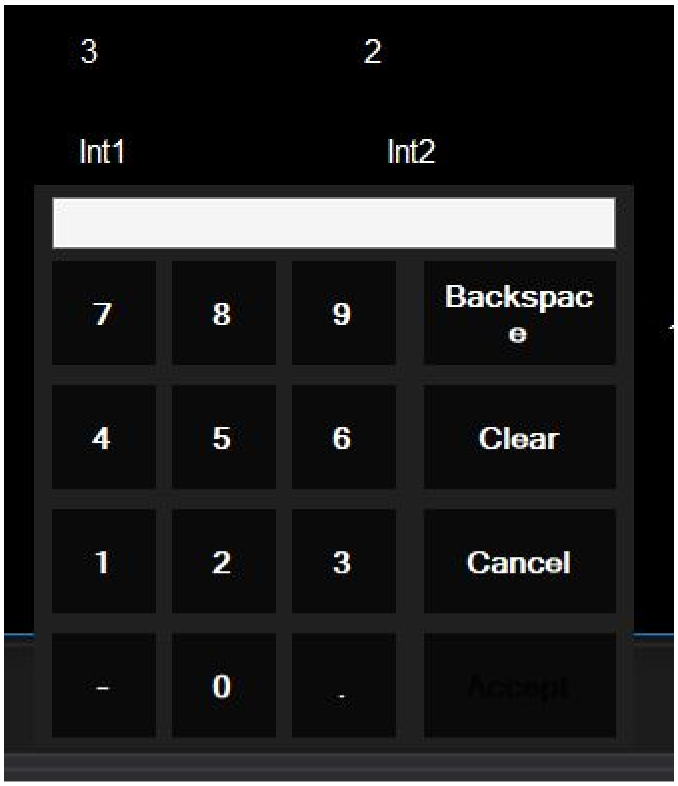
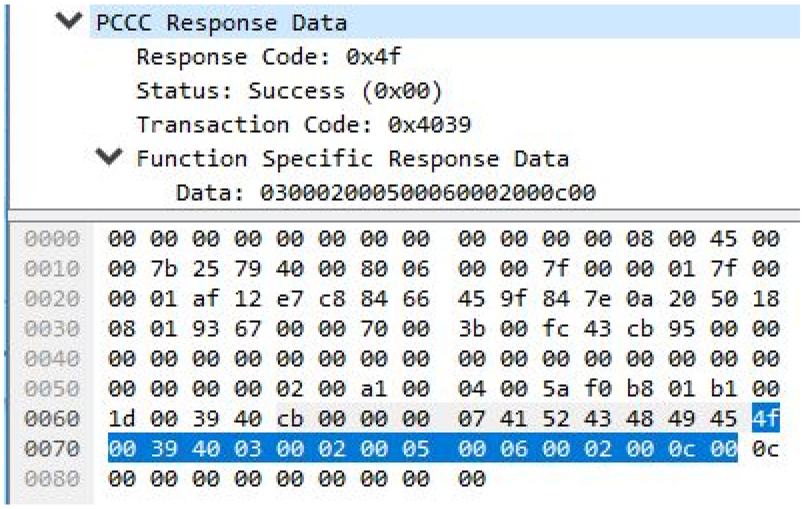


Figure 3: Reading Integers along the top row. Writing an integer value with the keypad.



```
  ∨ PCCC Response Data
      Response Code: 0x4f
      Status: Success (0x00)
      Transaction Code: 0x4039
    ∨ Function Specific Response Data
        Data: 030002000500060002000c00
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00
0010  00 7b 25 79 40 00 80 06  00 00 7f 00 00 01 7f 00
0020  00 01 af 12 e7 c8 84 66  45 9f 84 7e 0a 20 50 18
0030  08 01 93 67 00 00 70 00  3b 00 fc 43 cb 95 00 00
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0050  00 00 00 00 02 00 a1 00  04 00 5a f0 b8 01 b1 00
0060  1d 00 39 40 cb 00 00 00  07 41 52 43 48 49 45 4f
0070  00 39 40 03 00 02 00 05  00 06 00 02 00 0c 00 0c
0080  00 00 00 00 00 00 00 00  00
```

Figure 4: Response to Read Integer with for 12 Bytes.

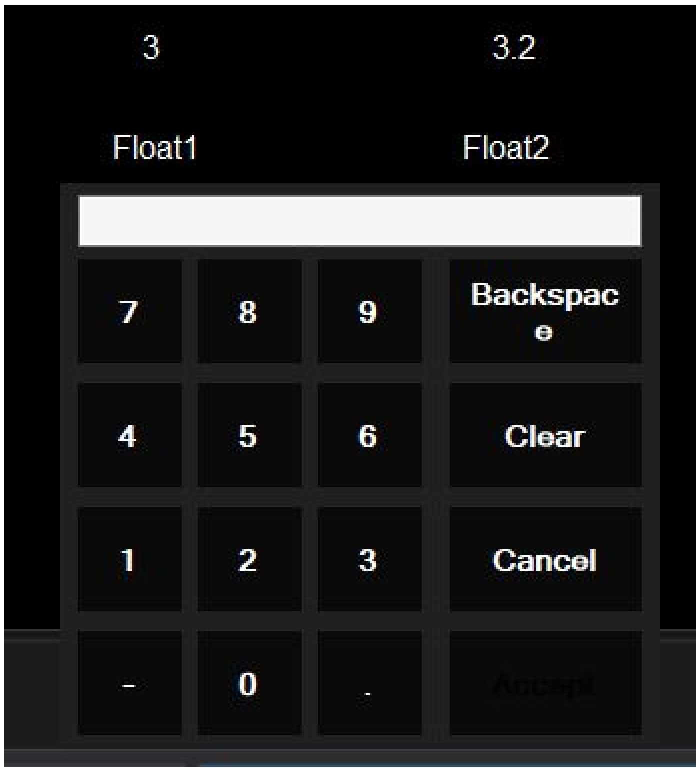
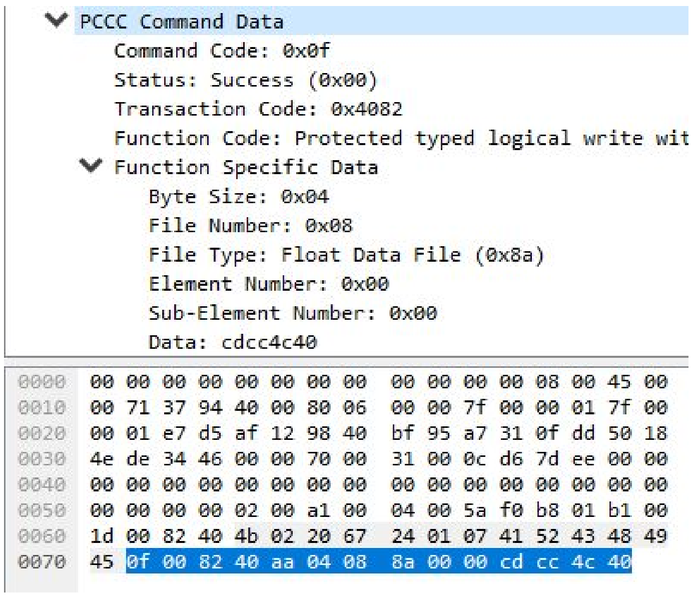


Figure 5: Reading Floats along the top row. Writing a float value with the keypad.



```
PCCC Command Data
    Command Code: 0x0f
    Status: Success (0x00)
    Transaction Code: 0x4082
    Function Code: Protected typed logical write wit
  Function Specific Data
        Byte Size: 0x04
        File Number: 0x08
        File Type: Float Data File (0x8a)
        Element Number: 0x00
        Sub-Element Number: 0x00
        Data: cdcc4c40

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00
0010  00 71 37 94 40 00 80 06  00 00 7f 00 00 01 7f 00
0020  00 01 e7 d5 af 12 98 40  bf 95 a7 31 0f dd 50 18
0030  4e de 34 46 00 00 70 00  31 00 0c d6 7d ee 00 00
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0050  00 00 00 00 02 00 a1 00  04 00 5a f0 b8 01 b1 00
0060  1d 00 82 40 4b 02 20 67  24 01 07 41 52 43 48 49
0070  45 0f 00 82 40 aa 04 08  8a 00 00 cd cc 4c 40
```

Figure 6: Writing a Float value.