Moller Magnet Power Supply Remote Communication with Siemens Programmable Logic Controller

Brian Eng, Mary Ann Antonioli, Peter Bonneau, Aaron Brown, Pablo Campero, George Jacobs, Mindy Leffel, Tyler Lemon, Marc McMullen, and Amrit Yegneswaran
Physics Division, Thomas Jefferson National Accelerator Facility, Newport News, VA 23606
February 20, 2024

This note presents the ladder (LAD) and the structured control language (SCL) used for communication between the Siemens programmable logic controller (PLC) with the Moller magnet power supply (MPS)—a product of the Officine Costruzioni Elettro Meccaniche (OCEM) Power Electronics company.

The OCEM Power Electronics' MPS for the Moller experiment uses TCP/IP over Ethernet to remotely control and monitor the unit.

A way of limiting access to the TCP/IP interface is to put the interface on a private network, connected to one port on the PLC controller; the other PLC port is connected to the Jefferson Lab network to communicate with devices not needed on the private network, e.g. computers running PLC software and EPICS IOCs.

Siemens provides libraries for communication (Support Entry ID: 109780503) with various protocols; the LCom library is for TCP/IP-based communication. After configuring the tags to the LCom_Communication function block, the PLC was able to send and receive commands from the MPS.

The PLC is programmed using the Totally Integrated Automation (TIA) Portal, which has four languages available to create a programming block: LAD, function block diagram, statement list, and SCL.

The default main organizational block when creating a new PLC project is LAD, which was carried over from when the PLC was first developed. To maintain consistency, LAD was selected for any additional blocks created, one of which was parsing the response from the MPS to commands from the PLC, Fig. 1.

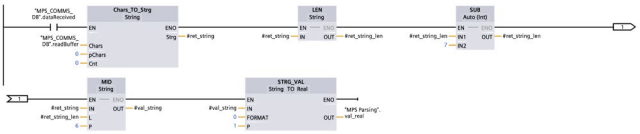

FIG. 1. LAD implementation of MPS response parsing.

The LAD code strips away the beginning part of the response, which echoes back the command issued, removes the trailing carriage return and new line characters, and finally converts the value into a floating-point number. This code spans more than a screen width and requires either scrolling or zooming out to view the whole screen.

Figure 2 shows the SCL code using the same calling conventions.

It is possible to further reduce the SCL code by removing some of the temporary tags, Fig. 3. These types of reductions are not possible in LAD.

```
0001 IF "MPS_COMMS_DB".dataReceived THEN
0002
0003   Chars_TO_Strg(Chars:="MPS_COMMS_DB".readBuffer,
0004            pChars:=0,
0005            Cnt:=0,
0006            Strg=>#ret_string);
0007
0008   #ret_string_len := LEN(#ret_string) - 7;
0009
0010   #val_string := MID(IN := #ret_string, L := #ret_string_len, P := 6);
0011
0012   STRG_VAL(IN:=#val_string,
0013         FORMAT:=0,
0014         P:=1,
0015         OUT=>"MPS Parsing".val_real);
0016
0017  END_IF;
```

FIG. 2. SCL code equivalent to LAD code.

```
0001 IF "MPS_COMMS_DB".dataReceived THEN
0002
0003    Chars_TO_Strg(Chars:="MPS_COMMS_DB".readBuffer,
0004             pChars:=0,
0005             Cnt:=0,
0006             Strg=>#ret_string);
0007
0008    STRG_VAL(IN:=MID(IN:=#ret_string, L:=LEN(#ret_string) - 7, P:=6),
0009         FORMAT:=0,
0010         P:=1,
0011         OUT=>"MPS Parsing".val_real);
0012
0013  END_IF;
```

FIG. 3. SCL code with reduced temporary tags.

Though it is not ideal to mix different languages in a single project handling string manipulation in particular, this approach seems to be the best at hand.