PyDA – CLX DAQ

PYTHON FOR DATA ACQUISITION

Section 1 – Configuration

Author: Alejandro Cadena
Github: https://github.com/acadena-repo

History

Revision	Date	Change Sections	Reason
0	May 2021		Initial issue

Table of Contents

1 OVERVIEW

Python for Data Acquisition is a set of functions developed in Python that are used to capture and analyze data sent by a PLC controller thru a data server function.

Taking advantage of the capability of new PLC controllers which implement direct calls to a socket interface, such as Rockwell Automation controllers, that it is used to expose process data.

The application is suitable during commissioning of equipment that needs to be monitored on-line in order to achieve proper tuning and setup with respect to its control and operation.

In this document, the use of the data server and a set of tools, is exposed through an example as a tutorial and in turn emphasizes the points where the reader can modify the application to obtain a better performance of it in other situations.

2 REFERENCES

Next documentation contain additional information that can be used as additional resources:

EIP Socket Interface - ENET-AT002D-EN-P - October 2020.pdf	Ethernet/IP Socket Interface Logix 5000 Controllers
https://docs.python.org/3/library/socket.html	Python Standard Library – Socket Module Documentation
https://scapy.readthedocs.io/en/latest/	Scapy Documentation
https://pandas.pydata.org/docs/	Pandas Documentation
https://matplotlib.org/stable/contents.html	Matplotlib Documentation

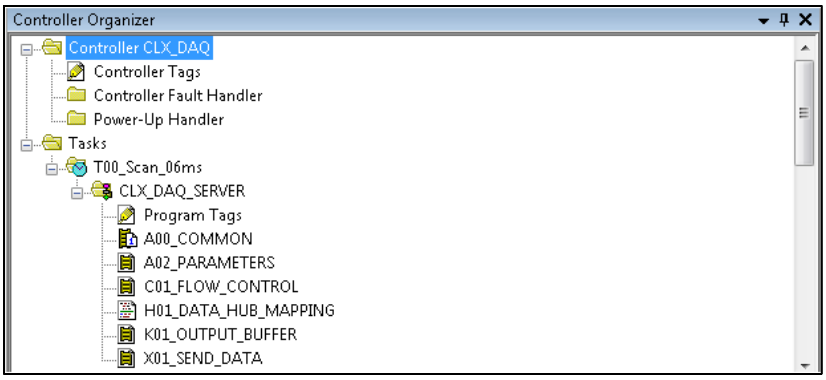
3 CLX-DAQ DATA SERVER

3.1 Data Server Structure

A Data Server program which can be fit on any Logix 5000 controller with Socket Interface capability was developed. The communication mechanism used to expose data was thru a UDP datagram under the “unconnected” configuration with the use of Message Instructions.

The reason of use UDP for the Data Server is to avoid service interruptions between the Data Server and the Client and also to improve speed on the data throughput. Each Data Server program works as a “Module” or “Data Package” which can send up to 450 bytes at a time.

The next figure shows the structure of the Data Server.



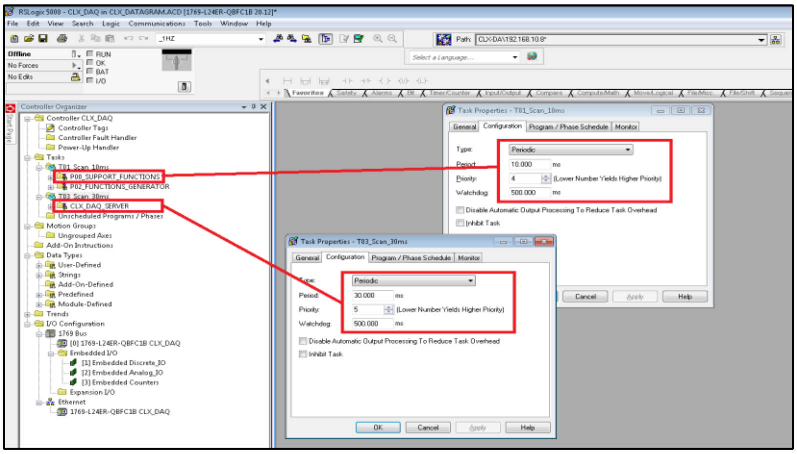
The structure of the Data Server is splitted in several subroutines to allow the user a better understanding of the program and to facilitate future modifications[1].

3.2 Configuration and Setup

To configure the server only two subroutines must be adjusted 1) A02_PARAMETERS: Inside the subroutine the user must set the IP address and the port number of the client to whom the data is intended to be received. 2) H01_DATA_HUB_MAPPING: All the data which are transferred to the client must be mapped into the server buffer. To facilitate this process an automated tool is used. ***Referes to - 4.2 Process Data Map ST-Code***

Two basic programs that need to be imported into the PLC application are supplied. The Data Server program and a support functions program which contains mainly time management functions. It is recommended that the support functions run in a periodic task at 10 milliseconds rate. The Data Server should be set in a periodic task multiple of the scan rate to which the user wants to send data to the client.

The next figure shows both programs configured in a PLC application.



[1] **The user is encouraged to encapsulate the Data Server in an Add-On instruction**

3.3 PyDA Package Header

The Data Server sends a UDP datagram of 462 bytes[2] of which only 450 bytes are used as a payload and 12 bytes are used as a header for the application protocol.

Header	Byte Start	Byte End	Byte Length	Description
Quality	0	0	1	Quality of service of each packet
Reserved	1	1	1	Reserved for future implementations
Timestamp	2	6	5	Packet timestamp: Month/Day/Hour/Minute/Second
Rate Code[3]	7	7	1	Code to determine the data flow rate
Packet Number	8	11	4	Number of packet

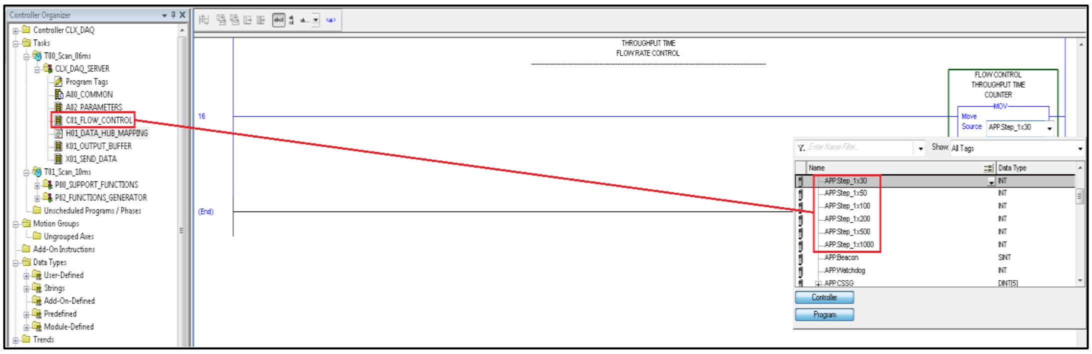
NB: The timestamp header is determined by the PLC internal clock. It is advisable to sichronize the PLC clock before any packet is sent.

3.4 Data Throughput Time

To allow the most flexibility during the setup of the packets data rate, the support functions count with a series of time step base counters which can be used to specify a data throughput rate for the Data Server.

This configuration allow to setup the Data Server program in a fix task and at the same time send a package at a different rate. Also, this configuration allow to modify the package throughput time in run time.

The data throughput time can be adjusted inside the Flow Control subroutine of the Data Server as shown in the next figure.



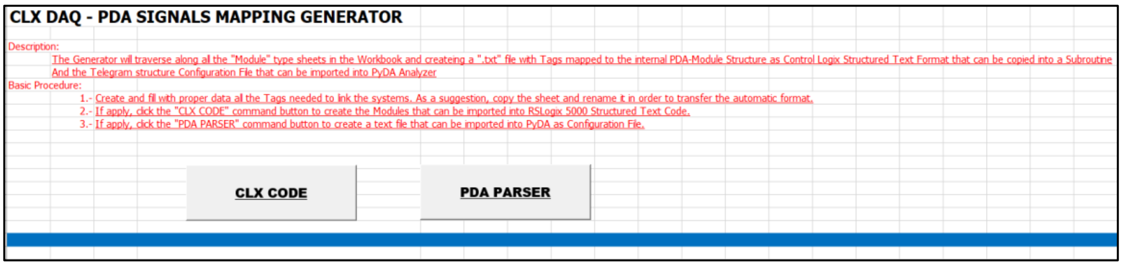
[2] **Refers to EIP Socket Interface - Table 3 - Maximum Packet Sizes on page 15**

[3] **See Appendix 1**

4 CLX-DAQ SIGNALS MAPPING GENERATOR

The Signals Mapping Generator is a tool that creates two files that are used by PyDA application. It generates a file with code that can be imported into the Data Hub Mapping subroutine as a process data mapping. Also, it generates a configuration file which specify the way the data captured by PyDA should be parsed.

The next figure shows the interface to generate both files.

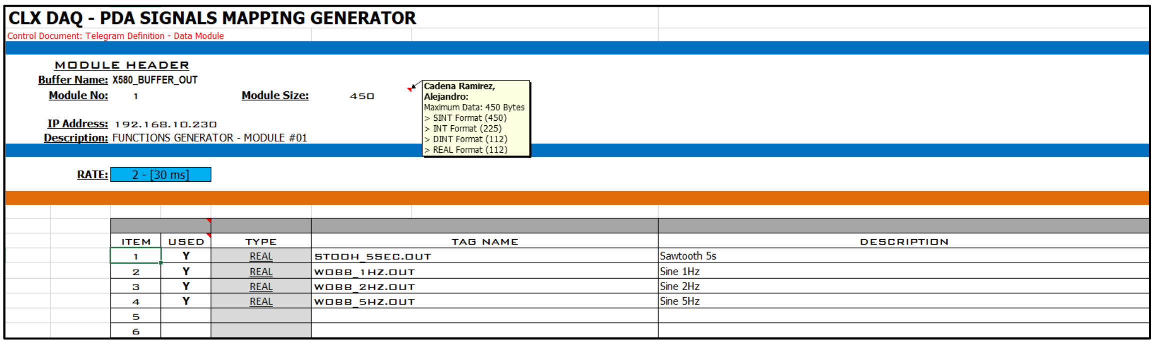


4.1 Data Module

Each data module handled by the Data Server is captured by PyDA as raw data, namely the data is stored as a string of bytes in hexadecimal notation. The reason to use raw data during the capture phase is to use no time to decode the data sent by the endpoint and allow the user to send the relevant information on its native data type (from the PLC controller point of view).

The Data Server buffer is declared as a byte array so, in order to send the process data thru the Data Server a mapping process should be performed and also the raw data captured need to be decoded to be analyzed.

To support both operations the Signals Mapping Generator provides a template that is used by the code generator and the parser. The next figure shows the main view of the data module template.



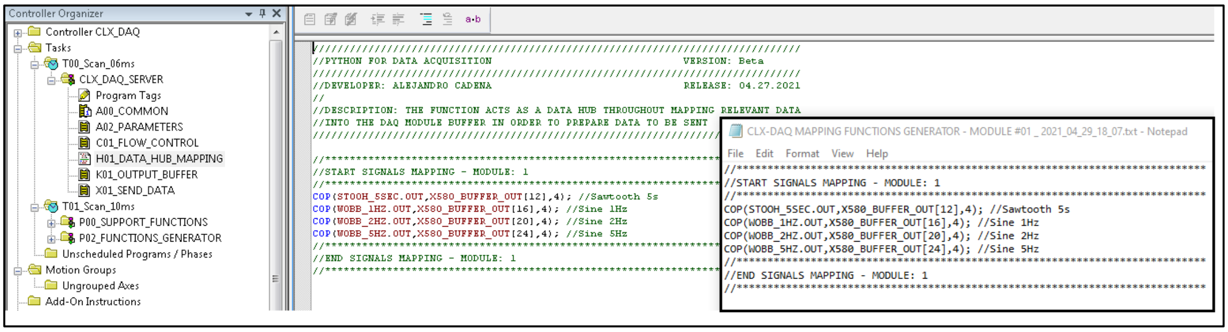
The user must provide the next information which is used by the generator:

- Buffer Name: Tag name in the PLC that is used as the module's buffer
- Type: PLC data type of the value that is sent. It can be SINT, INT, DINT or REAL
- Tag Name: Name of the tag which contains the value that is sent.
- Description: Description of the value that is sent.

4.2 Process Data Map ST-Code

Once the information needed by the Generator has been filled in the data module template, the Generator tool can be run (*CLX CODE* command button). The Generator tool creates a Structured Text code that can be copy and paste directly into the PLC subroutine. This process can be done even if the PLC is running, so no need to stop the PLC process to performe on line changes in the data module's configuration.

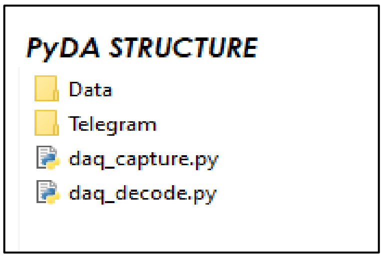
The next figure shows the output code from the Generator and its version in the PLC code.



5 PYDA CAPTURING & ANALYZING DATA EXAMPLE

5.1 PyDA Files Structure

PyDA is composed of two functions. 1) daq_capture.py which uses a network interface of the client machine to capture data sent by the PLC Data Server and save it in a text file as a raw data and 2) daq_decode.py which uses a telegram definition configuration file and the file generated by daq_capture.py to parse the raw data into its original format for further analysis into Pandas.



Both functions save the data processed into the Data folder. daq_capture.py creates a text file and daq_decode.py creates a .csv file. Also, daq_decode.py can returns a Pandas Dataframe so the analysis can be performed directly into a Jupyter Notebook.

The user must copy the configuration file generated with the Signals Mapping Generator tool (*PDA PARSER* command button), into the Telegram folder, previous to decode the capture file.

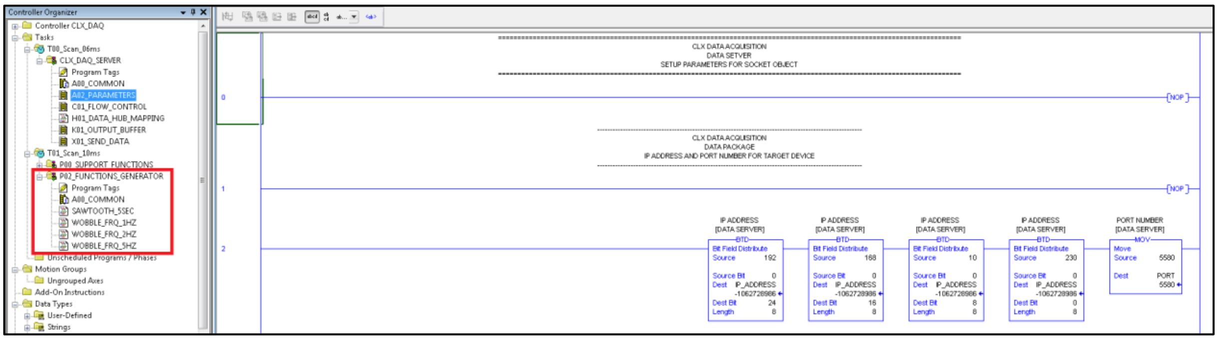
5.2 Use Case: Periodic Functions

A use case is described in order to show the complete usage of the application.

A Data Server has been setup into the PLC program where 4 periodic functions need to be captured. The Data Server is configured to send the data to a Client with IP address 192.168.10.230 and port number 5580.

The periodic functions are:

- A Sawtooth function of 5 seconds period
- A Sine functions with 1Hz frequency
- A Sine functions with 2Hz frequency
- A Sine functions with 5Hz frequency



The Signals Mapping Generator is used to link the periodic functions to the Data Server Buffer and to create the configuration file to decode the telegram. The two files created are:

- CLX-DAQ MAPPING FUNCTIONS GENERATOR - MODULE #01 _ 2021_04_29_18_07.txt
- CLX-DAQ TELEGRAM FUNCTIONS GENERATOR - MODULE #01 _ 2021_04_29_20_25.txt

Once all the configurations have been done, the capture function is started.

```
============================================================================================
INDEX  IFACE                                           IPv4             IPv6  MAC
19     Intel(R) Dual Band Wireless-AC 8260             [redacted]
1      Software Loopback Interface 1
22     Zscaler Network Adapter 1.0.2.0
23     Microsoft Wi-Fi Direct Virtual Adapter
14     Microsoft Wi-Fi Direct Virtual Adapter #2
25     VMware Virtual Ethernet Adapter for VMnet8
15     Intel(R) Ethernet Connection (2) I219-LM
20     Juniper Network Connect Virtual Adapter
16     VirtualBox Host-Only Ethernet Adapter
10     VMware Virtual Ethernet Adapter for VMnet1
-1     NdisWan Adapter
-2     NdisWan Adapter
-3     NdisWan Adapter
None
Select the index for the interface to be used:15
Port number that Data Server is connected:5580
Description for data file:PERIODIC FUNCTIONS
```

The file with the data capture is saved in the Data folder under the name:

- CLX-DAQ PERIODIC FUNCTIONS 30-Apr-2021_215939.txt

With this file and the configuration file, the function daq_decode.py is executed.

```
(FLASK) C:\NETWORKING\TCP-IP SOCKET PROGRAMMING\CLX SERVER - SOCKET COMMINICATION\CODE\PyDA>python daq_decode.py
Python for Data Acquisition                                                    Version: Beta
=============================================================================================

Name of the file with the captured data: CLX-DAQ PERIODIC FUNCTIONS 30-Apr-2021_215939
Name of the file with the Telegram Definition: CLX-DAQ TELEGRAM FUNCTIONS GENERATOR - MODULE #01 _ 2021_04_29_20_25
```

The function creates a .csv file with the data decoded which is saved in the Data folder under the name:

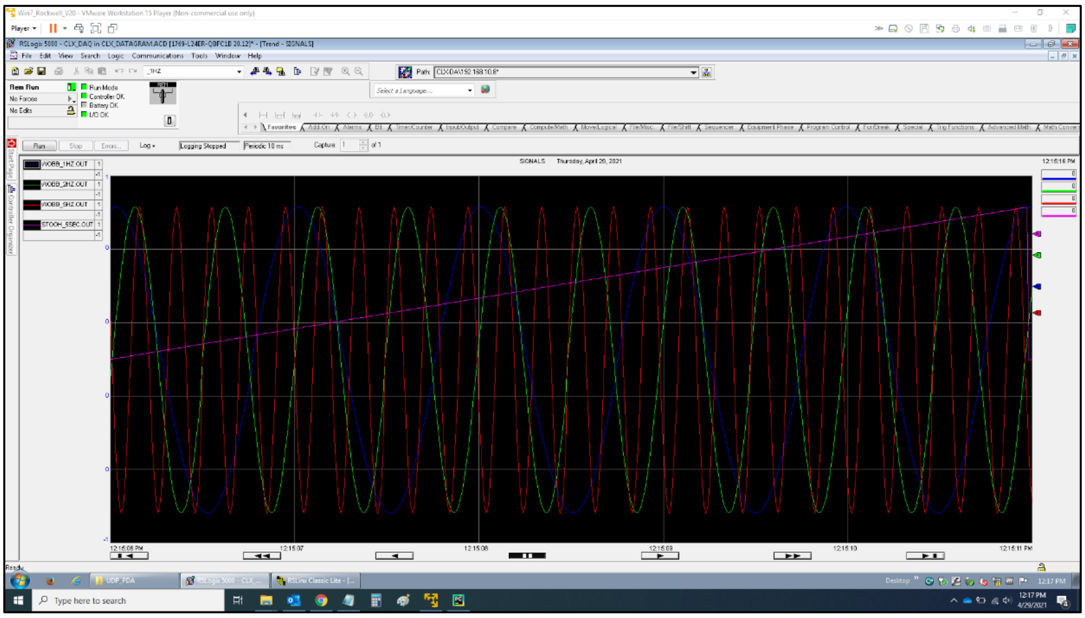
- CLX-DAQ DATAFRAME 30-Apr-2021_220316.csv

To analyze the data a Jupyter Notebook is used.

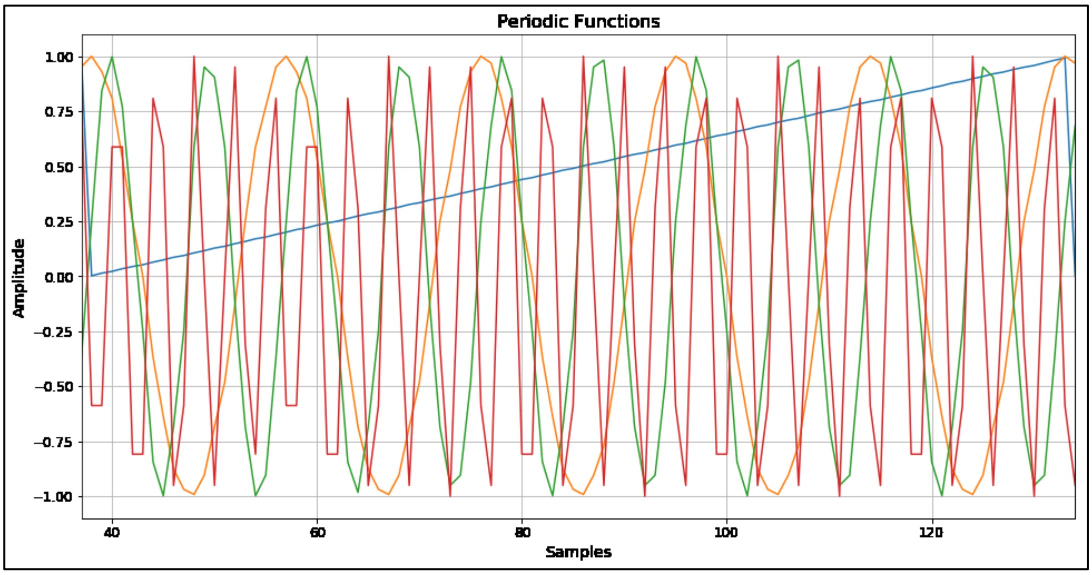
```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: df = pd.read_csv('.\Data\CLX-DAQ DATAFRAME  30-Apr-2021_220316.csv')
        df_func = df[['Sawtooth 5s','Sine 1Hz','Sine 2Hz','Sine 5Hz']][37:135]
```

As a comparative the periodic functions where captured with the use of the trend functionality of the PLC. A capture with a rate of 10ms is used.



And the data recorder with PyDA at a rate of 30 ms and plotted with Matplotlib.



1 APPENDIX

1.1 Rate Codes – Table

CODE	RATE	CODE	RATE	CODE	RATE	CODE	RATE
0	10 ms	7	80 ms	14	600 ms	21	3000 ms
1	20 ms	8	90 ms	15	700 ms	22	4000 ms
2	30 ms	9	100 ms	16	800 ms	23	5000 ms
3	40 ms	10	200 ms	17	900 ms	24	6000 ms
4	50 ms	11	300 ms	18	1000 ms	25	7000 ms
5	60 ms	12	400 ms	19	1500 ms	26	8000 ms
6	70 ms	13	500 ms	20	2000 ms	27	9000 ms