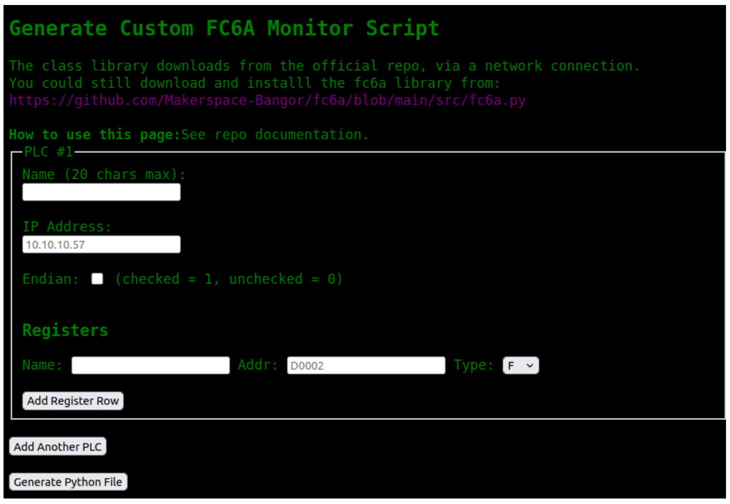

Generate Custom FC6A Monitor Script

The class library downloads from the official repo, via a network connection.
You could still download and installl the fc6a library from:
https://github.com/Makerspace-Bangor/fc6a/blob/main/src/fc6a.py

How to use this page: See repo documentation.

PLC #1

Name (20 chars max):

IP Address: 10.10.10.57

Endian: ☐ (checked = 1, unchecked = 0)

Registers

Name: Addr: D0002 Type: F

Figure 1: The Custom FC6A Monitoring Code Generation Page

Custom Monitor Program Generation Web Page

November 8, 2025

How to Use the Custom PLC Monitor Web Form

To use this form, you must give your monitor a name; this can be any name you want. The intent is to create a clearly labeled plot.

You need to know the IP addresses of every PLC you wish to monitor, and Maintenance Protocol must be running on the PLC, on port 2101. By whatever means, you need to have a stable network connection to the PLCs you wish to monitor.

You need to know the endian, else your data will not be correct while monitoring floats.

Name fields can be named whatever you want. Naming registers the values assigned on the PLC makes sense to us, but is not required in this application.

Addr: The registers need to be in `DNNNN` or `MNNNN` format. **Datatype:** This tells the program how to interpret the registers.

Add Another PLC: Adds a field to include another PLC and populate registers you want to plot.

When you are finished, click *Generate Python File.* This generates a script to your specification and allows for plotting of all registers you have previously defined.

What This Page Does

The Generate FC6A Python Logger web form is a PHP program that allows you, the user of IDEC PLCs, to create your very own custom monitoring applications using Python3 and our custom `fc6a` library.
The library uses our interpretation of Maintenance Protocol to read and write Bits, Words, and Floats to your compatible FC6A PLC. The script this page generates assigns different colors to different PLC plots. Registers of the same PLC will have the same plot color. At its core, this page populates an array (list) and then generates a script.

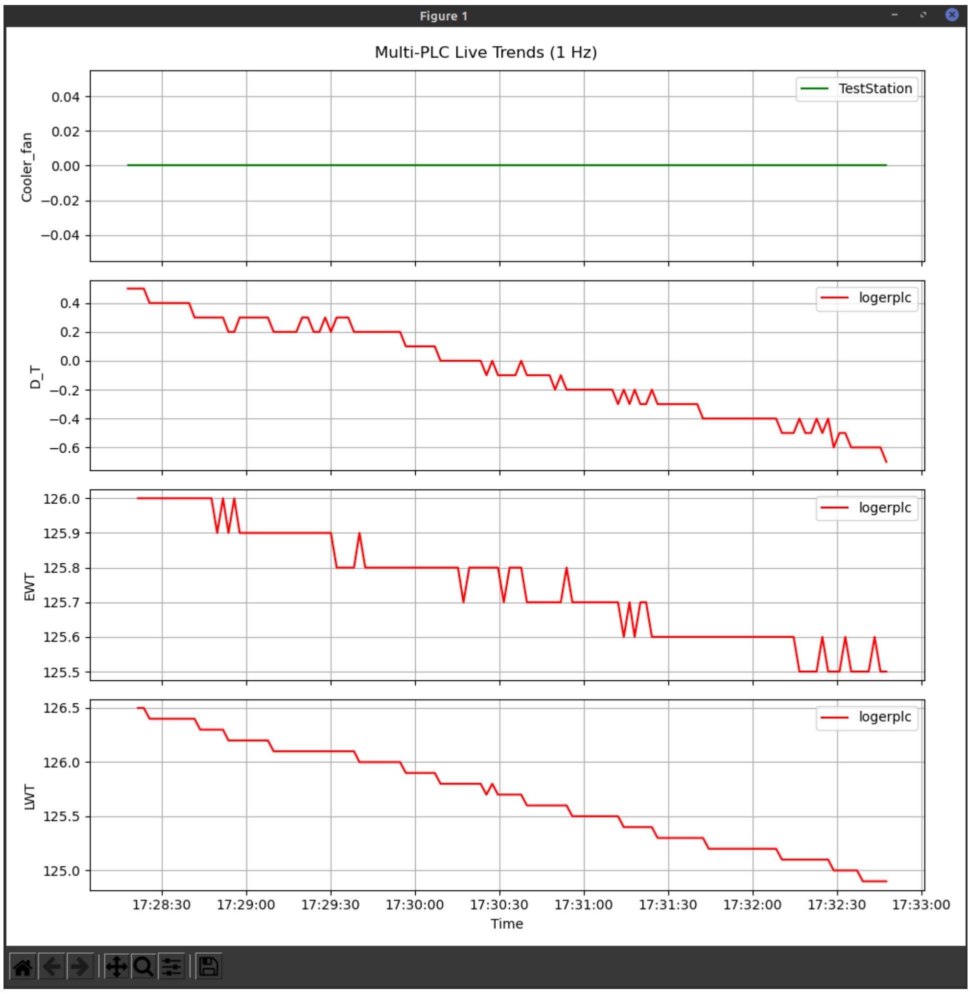
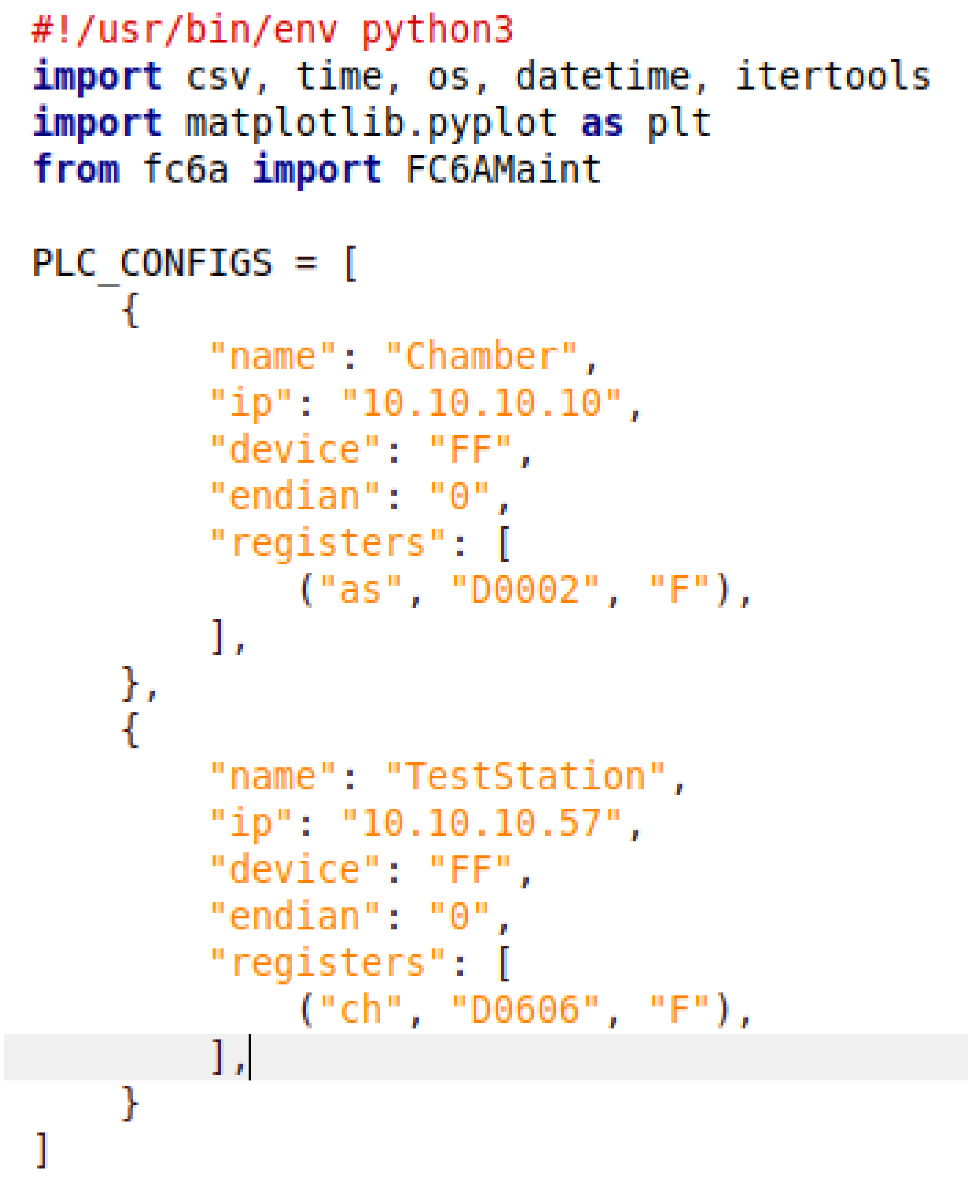


Figure 2: A Custom Monitoring Application generated via the web page

You may post-process the array if you are comfortable writing Python code. We think the code is fairly easy to read, but understand not everyone is a programmer, which is why we wrote this application.

Shown below is a sample of the PLC_CONFIGS list which dictates how the generated FC6A program operates. You may edit the downloaded files, as you see fit, by opening the file with a text editor, and modifying the PLC_CONFIGS section to meet new monitoring requirements. Correct typos, change endians, data types, etcetra.



```
#!/usr/bin/env python3
import csv, time, os, datetime, itertools
import matplotlib.pyplot as plt
from fc6a import FC6AMaint

PLC_CONFIGS = [
    {
        "name": "Chamber",
        "ip": "10.10.10.10",
        "device": "FF",
        "endian": "0",
        "registers": [
            ("as", "D0002", "F"),
        ],
    },
    {
        "name": "TestStation",
        "ip": "10.10.10.57",
        "device": "FF",
        "endian": "0",
        "registers": [
            ("ch", "D0606", "F"),
        ],
    }
]
```

Figure 3: Configuration section of a generated plotter

What You Need

The program generated by this page requires a network connection to port 2101 with Maintenance Protocol enabled on all PLCs you will communicate with. To run your custom application, you will need the `fc6a.py` library, or an internet connection. The script is capable of getting the necessary library from Github, provided you can access the internet at the time of running your script. You will also need Python3 and `matplotlib` installed.

Notes

While you do not need WindLDR to use the programs generated by this web form, you may need to know information about the PLCs you want to monitor. This may include, but is not limited to:

- Registers and their data types.
- Endianness of the PLC program.

To gather this information, your best option is to use windLDR.

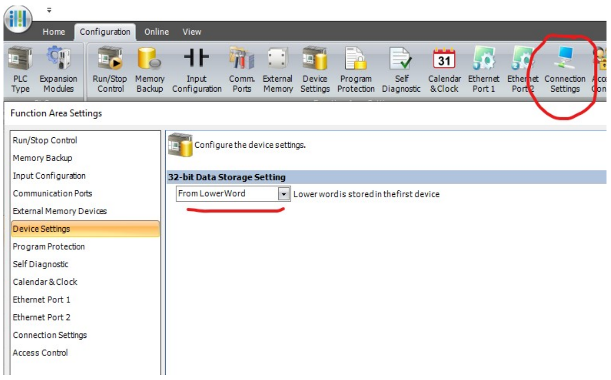
Endian

Endian is the computer science term for how bytes are packed into memory. IDEC uses different terminology, but it's the same principle.

Endian relates to multi-register data types (only floats are supported at this time) and how the data is stored in the respective registers. The FC6A library reads floats in succession. To place the data in the appropriate order, you need to know how it is stored.

We assert that "From Lower Word" is little endian, and that "From Upper Word" is big endian.
To find out the endianness of your PLC program:

> Load WindLDR, go to *Connection Settings* → *Device Settings...*, then view the "32-bit Data Storage" settings drop-down menu.



Configure the device settings.

32-bit Data Storage Setting

From Lower Word — Lower word is stored in the first device

Figure 4: Endian settings in WindLDR

We suggest not changing your PLC program. The FC6A library can cope with PLCs of different endians, you simply have to specify how the PLC is set up to read it correctly. If you happen to set it incorrectly, Python scripts are open-source text files, so you may open and modify them in any text editor. Or you could run the online code generator again!

Compatibility

The programs have been tested on Windows 10+ and Linux. We see no reason for them not to work on other platforms we simply have not tested them. The FC6A library may be compatible with other PLCs, including those of the FC4A series, but we only have access to FC6A PLCs at this time.

Security

When you configure your Custom Monitor and click “Generate Python File,” your browser may warn you about the file being a potential security risk. Ignore this, and download the program. You may view and review the source code as you wish.

PLC Security in General

PLCs are not quite computers, and practically speaking, have no security. If you are using PLCs, it is our advice and sincere hope that you do so behind a firewall or VPN.

Resources

Further reading:

- Primary source: IDEC FC4A Protocol Implementation Manual
- Secondary source: IDEC FC6A MicroSmart Communication User Manual
- WindO/I-NV4 User Manual

Bugs / Features / Limitations / Requests

Make the webpage look better. Add doubles and strings!

- We cannot write special registers at this time.
- We only support Bits, Words, and Floats at this time, but are open to expanding this in the future.
- Add support for timers, counters, and other objects you are welcome to contribute.

What happened to the logging output? IDK man, I’m making programs that make programs, I took it out to make a shorter program. I might put it back in.

The plotting screen causes other windows to flash. That’s a known bug, and I assume it can be fixed later pretty easily. This program is all of one week old.

To our surprise, Maintenance Protocol has been remarkably stable and reliable. This web form is intended to monitor and plot a limited set of key values to determine operational trends.

Loading dozens of registers won’t overload Maintenance Protocol but may make the plotting screen squished.

When the computer loses network connection, “Error reading Name from Device timed out” will print to the console. A date-time on this would be nice.

Make clicking the X on the window close the app instead of requiring `CTRL+C` from the console.

Can we make the plots an online service? Sure! Are you going to pay the hosting fees?

How to Break things

Monitor everything. Literally everything. Load every register on the PLC. You can do it, and mayhem may ensue. Although, to be honest, we have done this, and the PLCs never crashed out. The HMI screens started flashing though. The plotting window will probably crash before the PLCs will. That's why you have to love a PLC, they're stable and resilient.

While we have made code to read blocks of data, (see example code) the Custom Program Monitoring webpage, and generated scripts are single read operations. Loading hundreds of registers successively, will reduce networking performance. The PLCS will respond, but you wont get 1 second response times. We have no fixed figure to how much is too much. Performance depends on more than the PLCs, including network configurations, and network performance. The more traffic (read registers) the more handshaking goes on.

Legal

The FC6A.py library is open-source, as is Python3 and all the materials in this project. IDEC PLCs, however, are proprietary. We do not assert any claim to software or hardware licenses, or intellectual properties owned by IDEC. The `FC6A.py` library only provides a means to access and interpret resources already available to users of IDEC PLC systems. We assert that the use of this web page, and affiliated project files are for educational purposes only. Use at your own risk.

Final Notes

The `fc6a` library is a work in progress and not a complete implementation of the Maintenance Protocol.

While writing the aforementioned datatype is possible, programming the PLC is not possible, or even intended with this tool, we are only offering a means to monitor and plot data related to your PLCs. To program your IDEC PLCs, you will need WindLDR, available from IDEC through their Automation Organizer suite.

The Custom PLC Monitor Code Generator and subsequently generated codes are also a work in progress.

We are open to suggestions or contributions to the codebase. If you have questions, concerns, or thoughts, feel free to raise them on GitHub or contact us at:

contact@bangormakerspace.org