# Flex – BESS Control Software

## About

|  |  |
| --- | --- |
| Author: | S. Coates |
| Revision: | 1 |
| Date: | 20-09-24 |
| Audience | MSP and Stakeholders |

## History

* Flex 1 – Python 3 + Tkinter application deployed on one site. Single threaded and manually operated, it controls a small Tesvolt stack and early EPC Inverter.
* Flex 2 – Python 3 + Web HMI to separate tasks and allow for more detailed HMI creation and graphical elements. Deployed on only two remaining sites and not scalable due to single-threaded and customer specific coding methodology.
* Flex 3 – Python 3 + Web, solves the scalability issues of Flex 2 by modularising both the HMI and Python elements into separate entities, allowing different developers to independently interface with external hardware without affecting core operations. Multi-threaded to improve system performance.
* Flex3.5 – As per Flex 3 but with improved timing to solve concurrency issues discovered while deploying on complex sites. Local threads have been replaced with independent processes to allow for true concurrency and mitigate the limitations imposed by the Python Global Interpreter Lock (GIL).   
    
  Local and Remote data logging are now spawned to a process which can have functionality added to it while systems are running. Most stable version.

## File Structure

logs

Folder containing locally stored system log data in excel format

static/images

Folder containing image data for the web-based HMI

templates

Folder containing the HTML / Javascript / CSS component of each module.

Flex3.5.py

The entry point to the system, double-clicking this file or launching from shell:startup will initialise the BESS. It loads a HMI configuration from db.json and performs a round-robin interrogation of each loaded module (peripheral) collecting data, which is then processed by the Control, Logging, SCADA and Client modules to facilitate predefined system behaviour and Remote Control.

FlexDB.py  
Helper functions for accessing the system configuration database through the tinydb library.

BACnetServer.py

BACnet to Modbus converter, allows us to split different data into their own modules for easier representation on the HMI and presentation in the SCADA output data under their respective sections. Currently reads AC Meter and AC Solar data presented by Edmonton EcoPark, CXP-00884.

FlexLogger.py

A process launched by FlexMod\_MspLogging.py to handle remote logging to the server and local logging to file, and email reporting of alerts and faults to designated recipients. Can be side-loaded during runtime to add more functionality.

FlexMod\_AccuACMeter.py

AC Meter Module for data retrieval from an Accuenergy Acuvim II Power Meter.

FlexMod\_BACNetACMeter.py

AC Meter module for data retrieval of AC Meter data from a designated BACnet register populated by the client from a pool of devices on CXP-00884. We use a proxy application, BACnetServer.py as a BACnet to Modbus-TCP bridge and separate the data used by FlexMod\_BACnetACSolar.py.

FlexMod\_AdamAnaIO.py

Analogue IO module for data retrieval from an Advantech ADAM-6015 AIO device.

FlexMod\_AdamDigIO.py

Digital IO module for data retrieval from an Advantech ADAM-6060 AIO device.

FlexMod\_BACNetACSolar.py

AC Solar module for data retrieval of AC Solar data from a designated BACnet register populated by the client from a pool of devices on CXP-00884. We use a proxy application, BACnetServer.py as a BACnet to Modbus-TCP bridge and separate the data used by FlexMod\_BACnetACMeter.py.

FlexMod\_AmptDCSolar.py

DCSolar module for data retrieval and configuration of an AMPT Communications Unit (CU).

FlexMod\_BenderACefm.py

AC Earth Fault Monitor module to read from the Bender ISO685 device.

FlexMod\_BenderDCefm.py

DC Earth Fault Monitor module to read from the Bender ISO685 device.

FlexMod\_Client.py

Client module which takes overall control of a system. Once enabled, it will close the battery contacts, start the inverter(s) and begibn operating as requested by the client, whether that be base operations such as Time Of Us (TOU) / Aggregation / Peak Shaving, Solar Import or specific functionality that is tailored to the client to support automated power movement and new telemetry.

FlexMod\_CXP-00780\_Master.py

Client module code specifically for Heron Lynas’ site comprising of a master and two slave systems. As part of the system enable procedure, this module enables two Flex1000 BESS systems over SCADA. All other systems use modified versions of FlexMod\_Client.py for control.

FlexMod\_CXP-00780\_Slave.py

Client module complimentary to FlexMod\_CXP-00780\_Master.py, it enables power to the Kore RBMS systems when enabled and reports performance telemetry back to the Master over SCADA.

FlexMod\_KoreBattery.py

Battery module code written to interface with the Kore KP-MC BMS, to enable and and monitor rack state.

FlexMod\_MideaAircon.py

Aircon module for communication over Modbus-RTU over the Intesis-Midea communications interface.

FlexMod\_MspControl.py

Control module code which monitors system health and passes data to the client module.

FlexMod\_MspLogging.py

Logging module which used to primarily compile system-wide data for transmission to the remote server and log locally. Now it serves as a reference point for the HMI and pass-through for FlexLogger.py to enable side-loading of new functionality.

FlexMod\_MspLogo.py

HMI decoration, redirects to the MSP homepage.

FlexMod\_MspSCADA.py

SCADA module for remote control and monitoring of the BESS over Modbus-TCP. May in future support other remote interfaces such as MQTT.

FlexMod\_NexcerisLiIonTamer.py

Li-Ion Tamer module for direct monitoring of the system’s off-gas detection system.