Microwave Offset Requirements

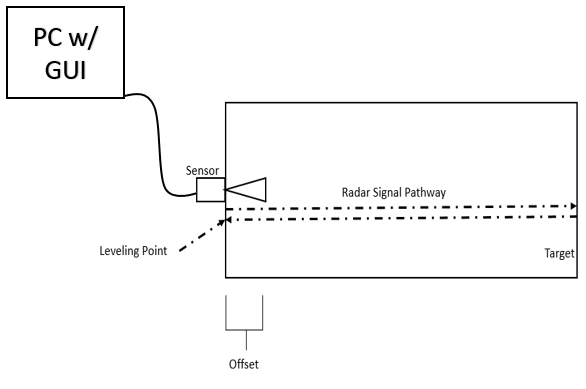
# Section A—System Fundamentals

Microwave radar range sensors are being used to measure water levels in the USA in various locations. Some sensors used at the Center for Operational Products and Services (COOPS) uses time-of-flight principles to determine range while others use FFT. Due to manufacturing tolerances and the implementation on the National Water Level Observation Network (NWLON), the range measured by the sensor is offset from the virtual zero point.

The sensor is affixed to a metal structure with a known distance-to-target and high frequency range data is averaged, recorded and analyzed to compute the “sensor offset.” Each sensor’s offset is then applied in the computation of water level during post data processing prior to being provided to users.

Because these sensors have serial output which can be output at 1Hz, we are able to generate raw data for comparisons and determination of sensor resolution, accuracy and consistency. We are also able to compute the offset distance from the sensor’s ‘zero point’ to the ‘leveling point’.

To ensure that every sensor goes through the same validation process, we use standardized methods of data quality assurance. Each sensor’s offset measurement is then recorded and identified by the sensor serial number. Offsets may be recorded in a database in Silver Spring by the field installation team and also on a label printed and affixed directly to the sensor.

Figure 1: Offset System Diagram

Power is provided to the sensor via an SDI-12 to RS-232 conversion device which also allows the sensor to be easily integrated to a desktop PC and our smart data collection platforms (DCPs). Field applications use the SDI-12 interface for direct data retrieval, trimmed means and other basic configuration functionality. The RS-232 interface is used to generate signal quality curves and interface with lower level hardware settings.

Software for validation will reside on an industrial grade computing platform which will be used for the sole purpose of performing sensor offset verification testing. COOPS needs to procure the embedded computing platform, monitors and associated hardware to move forward.

# Section B—General Application Requirements

The application should fulfill the following requirements.

* Function as specified in *Section C: Application Requirements* of this document.
* Conform to LabVIEW coding style and documentation standards found in LabVIEW documentation. Refer to the *Development Guidelines* section of the *LabVIEW Help.*
* Be hierarchical in nature. All major functions should be performed in SubVIs.
* Be scalable to add more states or features without complete refactoring.
* Minimize the use of excessive structures, local or global variables, and Property Nodes.
* Respond to front panel controls within 100 ms and never use 100% of CPU time.
* Close all opened references and handles where used.
* Be well documented and include the following documentation features.
  + Labels on appropriate wires within the Main VI and SubVIs
  + Descriptions for each algorithm
  + Documentation in **VI Properties » Documentation** for both main VI and SubVIs
  + Tip strips and descriptions for front panel controls and indicators
  + Labels for constants

# Section C—Application Requirements

## Definitions

This section defines the terminology for the project.

* MWWL — Microwave water level sensor, typically of the H3611i or Nile variety from YSI.
* Legacy Offset Mount — The original AS II offset mount is an aluminum frame where the top of the mount is 1 meter from the sensor’s target area, allowing for a computation of a sensor offset.
* New Offset Mount — Because the minimum range and blanking distance for the long range Nile version MWWL is 2 meters, a larger frame was designed and constructed.
* SDI-12 — Serial Digital Interface at 1200 baud. An asynchronous serial communications protocol for intelligent sensors that monitor environment data.

## Task:

## Develop a sensor offset tool using LabVIEW. The front panel of the controller should look similar to the Front Panel shown in Figure 2.

Figure 2. Front Panel GUI

## General System Workflow

Each MWWL sensor is attached to the legacy or new offset mount with ¼”- 20 bolts. Sensors are then connected to a Waterlog SDI-12 to RS-232 converter which allows a serial interface to the instrument. The operator selects their name and location from the drop-down menus and then pushes the Start button. Five minutes of data are then collected, averaged and recorded before the sensor offset is computed and displayed to the user. Raw serial output is continuously displayed to the user in a String Indicator box for troubleshooting purposes and if any errors occur with the data or sensor a Red Fault Indicator LED will light up, the operator will be notified and the test will cease.

Previously used Control values will be stored in a configuration file (defaults.ini) and loaded as defaults on the next run iteration.

## Sequence of Operation

This section describes the sequence the operator follows to operate the software for computing sensor offsets.

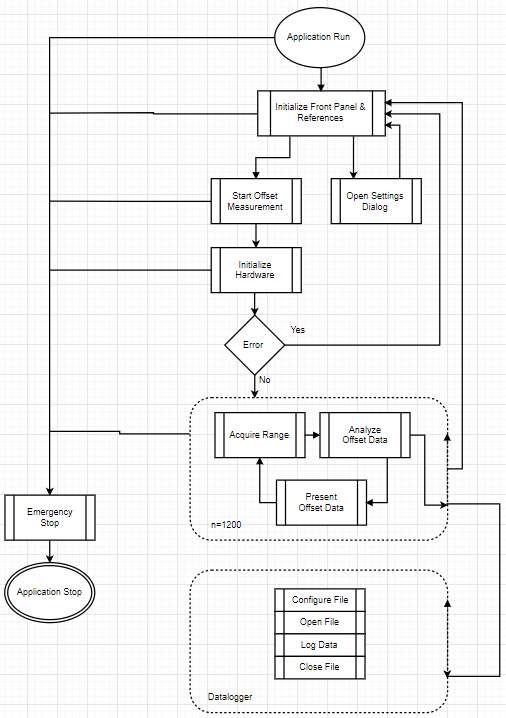


Figure 3. Offset Computation - Sequence of Operation

### Application Run

When the application starts, the **Serial Output** string Indicator will be clear and **Fault LED** will be OFF. Operator Name and Operator Location will be initialized to the last used value, saved in the defaults.ini file. The Initialize controls as indicated in Table 1.

Table 1. Initial Control Values for the User Interface

|  |  |
| --- | --- |
| Control | Initial Value |
| Operator Name | Operator Name |
| Operator Location | Operator Location |
| Start Button | OFF |
| Stop Button | OFF |
| Reset Button | OFF |

Table 2. Initial Indicator Values for the User Interface

|  |  |
| --- | --- |
| Indicator | Initial Value |
| Serial Output | (blank) |
| Fault LED | OFF |
| Sensor Offset (m) | 0.000 |

The **Operator Name** and **Operator Location** indicators are enabled at application start. All other controls are disabled to ensure that the operator starts the process properly. Once an operator and location have been chosen, the **Start Button** control will be enabled.

The application reads sensor configuration data from an INI file. These values initialize the controller and the data logging mechanism. Different sensors and/or firmware versions use different INI files.

The following data is sent to the status log file:

* Time string: Absolute date and time at event in GMT
* Event string: **System Initialized**
* Event data string: **-**  
  (There is no relevant data to log for this process)

**Note:** Refer to the *File Specifications* sectionfor file format and update policies for the status log file and the INI file.

### Start

When Start Button is clicked, the VISA resources are initialized and the sensor is polled to validate communication and determine sensor serial number.

When this step begins, log the following data to the status log file.

* Time string: **NA**
* Event string: **Operator: %s**
* Event data string: **Operator’s name**
* Time string: NA
* Event string: **Location: %s**
* Event data string: **Location’s name**
* Time string: Absolute date and time at event **Example 12/29/2017 10:35AM**
* Event string: **None**
* Event data string: **None**

Disable the **Start** button so that the operator does not click it a second time during the measurement cycle.

The sensor is then polled for its Serial Number using the ‘YI!’ command in the SDI-12 protocol. The Front Panel Controls for Operator & Location are inserted in the header information of the log file prior to the start time, along with the sensor SN.

After the initialization completes, the **Status** string indicates **Acquiring and Logging.** Log the following data to the data log file.

* Time string: mm/DD/YYYY HH:MM:SS
* Event string: **NA**
* Event data string: The calculated offset values from sensor, in meters, to 3 digits of precision.
* Delimiter: Tab

Update the StatusBox with the ASCII output from the sensor.

Decrement the Time Indicator from 10 minutes to zero, 1 second at a time.

Update the Scope chart with each calculated offset value.

### Stop

Click the **Stop** button to cancel the current offset calculation that is in progress. This action will free up the COM port and release the data file references but will save the data in the \*.log file for future references.

When the test completes, the **Status** string indicates **Ready, the UI loop computes the offset** which is printed to the StatusBox and the log file is closed.

Enable the **Start** button and disable the **Reset** button to ensure that the operator starts the boiler properly.

* Time string: Absolute date and time at event
* Event string: **Forced Draft Fan ON**
* Event data string: **True**

Now that the boiler is running, close the pilot gas valve (turn off the **Pilot Gas Valve** LED), which turns off the pilot flame (turn off the **Pilot** LED). The boiler now begins monitoring its **Simulate Failure** button.Log the following data to the status log file.

* Time string: Absolute date and time at event
* Event string: **Boiler Running**
* Event data string: Value of Fuel Control Valve Position

### Stop Boiler

Stop the flow of natural gas to the combustion chamber to stop combustion and initiate a purge cycle to clear any remaining combustible gasses from the combustion chamber.

Either of the following conditions can shut down the software when it is running.

* Click the **Stop** button.
* Click the **Exit** button.

* Time string: Absolute date and time at event
* Event string: **Start Shutdown Purge**
* Event data string: **0**

Log the following data to the status log file.

* Time string: Absolute date and time at event
* Event string: **Shutdown Purge Complete**
* Event data string: Purge elapsed time

### Emergency Stop

This button shuts down the application completely. Click this button to halt execution of all parallel loops, close all open references, and exit.

## Handle Errors

Depending on the type of error, the system should respond differently.

* Critical errors — Result from problems that prevent the application from executing safely. These errors halt the application in the same manner as the Emergency Stop button. Log critical errors to disk.  
  **Note:** Refer to the *File Specifications* section for file format and update policies.
* Non-critical errors — Glitches or minor problems that do not affect safety and do not prevent normal execution. Handle these errors locally instead of shutting down the system.

## Description of Controls and Indicators

Table 2. Description of Controls

|  |  |  |
| --- | --- | --- |
| Control Name | Description | Function |
| **Reset** | Button | Resets the boiler |
| **Start** | Button | Initiates the pre-purge step |
| **Light Pilot** | Button | Initiates the ignition step |
| **Fuel Control Valve** | Numeric | Controls amount of fuel flowing into the boiler |
| **Stop Boiler** | Button | Initiates the boiler shutdown |
| **Emergency Stop** | Button | Stops the application within the specified time frame |

Table 3. Description of Indicators

|  |  |  |
| --- | --- | --- |
| Indicator Name | Description | Function |
| **Status** | String | Indicates boiler status (current step) |
| **Primary Fan** | LED | Indicates ON/OFF status of the primary fan |
| **Pilot Gas Valve** | LED | Indicates ON/OFF status of the natural gas valve |
| **Pilot** | LED | Indicates ON/OFF status of the pilot |
| **Forced Draft Fan** | LED | Indicates ON/OFF status of the forced draft fan |

## File Specifications

This section describes the three files the boiler controller uses to initialize configuration data, log status information, and log error information.

### INI File Specification

* **File Name**: Boiler Init.ini
* **File Location**: Relative – same location as the main VI
* **Format**: Configuration file
  + Sections — Message Handling Loop (MHL), Controller, Boiler
  + Keys:
    - MHL
      * Fuel Control Valve Maximum
      * Fuel Control Valve Minimum
    - Controller
      * Purge Time
      * Flame Threshold
    - Boiler
      * Pilot Increment (ms)
      * Pilot Decrement (ms)
* **File Creation and Modification**: Create the file and add configuration data as needed for the application.

### Status Log File Specification

* **File Name**: Status Log.txt
* **File Location**: Relative – same location as the main VI
* **Format**: Tab-delimited text file
* **File Header**: Timestamp, Event, Event Data
* **Event Data string format**:
  + Absolute date and time string
  + Event string
  + Event data string
* **File Creation and Modification**: If file does not exist, create new, write header and log status data. If file exists, append status data to the end of the file.

### Error Log File Specification

* **File Name**: Error Log.txt
* **File Location**: Relative – same location as the main VI
* **Format**: Tab-delimited text file
* **Error Data string format**:
  + Absolute date and time string
  + Error Code
  + Error Source (the full call chain information is not needed)
* **File Creation and Modification**: If file does not exist, create new and log error data. If file exists, append error data to the end of the file.