

APPLICATION NOTE PQMAN01: EVENT RECORDER APPLICATION

The Event Recorder stores all online data in a section of non-volatile memory when triggered by an event. The PQM defines any of the following situations as an event:

- Analog Input Alternate Alarm
- Analog Input Alternate Alarm Clear
- Analog Input Main Alarm
- Analog Input Main Alarm Clear
- Clear Event Record
- Clock Not Set Alarm
- Clock Not Set Alarm Clear
- COM1 Fail Alarm
- COM1 Fail Alarm Clear
- COM2 Fail Alarm
- COM2 Fail Alarm Clear
- Current THD Alarm
- Current THD Alarm Clear
- Current Unbalance Alarm
- Current Unbalance Alarm Clear
- Data Log 1 Alarm
- Data Log 1 Alarm Clear
- Data Log 2 Alarm
- Data Log 2 Alarm Clear
- kVA Demand Alarm
- kVA Demand Alarm Clear
- Negative kvar Alarm
- Negative kvar Alarm Clear
- Negative kvar Demand Alarm
- Negative kvar Demand Alarm Clear
- Negative kW Alarm
- Negative kW Alarm Clear
- Negative kW Demand Alarm
- Negative kW Demand Alarm Clear
- Neutral Current Demand Alarm
- Neutral Current Demand Alarm Clear
- Neutral Overcurrent Alarm
- Neutral Overcurrent Alarm Clear
- Overcurrent Alarm
- Overcurrent Alarm Clear
- Overfrequency Alarm
- Overfrequency Alarm Clear
- Overvoltage Alarm
- Overvoltage Alarm Clear
- Parameters Not Set Alarm
- Parameters Not Set Alarm Clear
- Phase A Current Demand Alarm
- Phase A Current Demand Alarm Clear
- Phase B Current Demand Alarm
- Phase B Current Demand Alarm Clear
- Phase C Current Demand Alarm
- Phase C Current Demand Alarm Clear
- Phase Reversal Alarm
- Phase Reversal Alarm Clear
- Positive kvar Alarm
- Positive kvar Alarm Clear
- Positive kvar Demand Alarm
- Positive kvar Demand Alarm Clear
- Positive kW Alarm
- Positive kW Alarm Clear
- Positive kW Demand Alarm
- Positive kW Demand Alarm Clear
- Power Factor Lag 1 Alarm
- Power Factor Lag 1 Alarm Clear
- Power Factor Lag 2 Alarm
- Power Factor Lag 2 Alarm Clear
- Power Factor Lead 1 Alarm
- Power Factor Lead 1 Alarm Clear
- Power Factor Lead 2 Alarm
- Power Factor Lead 2 Alarm Clear
- Power Off
- Power On
- Pulse Count Total Alarm
- Pulse Input 1 Alarm
- Pulse Input 1 Alarm Clear
- Pulse Input 2 Alarm
- Pulse Input 2 Alarm Clear
- Pulse Input 3 Alarm
- Pulse Input 3 Alarm Clear
- Pulse Input 4 Alarm
- Pulse Input 4 Alarm Clear
- Pulse Input Total Alarm Clear
- Reset
- Self Test Alarm
- Self Test Alarm Clear
- Setpoint Access Enabled
- Switch A Alarm
- Switch A Alarm Clear
- Switch B Alarm
- Switch B Alarm Clear
- Switch C Alarm
- Switch C Alarm Clear
- Switch D Alarm

A

- Switch D Alarm Clear
- Time Alarm
- Time Alarm Clear
- Trace Memory Trigger
- Undercurrent Alarm
- Undercurrent Alarm Clear
- Underfrequency Alarm

Up to 40 events can be stored in non-volatile memory for retrieval and review. The Event Recorder can be enabled, disabled, or cleared via the keypad or serial port. The following data is saved when an event occurs:

- Analog Input (high)
- Analog Input (low)
- Date - Month/Day
- Date - Year
- Event Cause
- Event Number
- Frequency
- I Unbalance
- Ia
- Ia Demand
- Ia THD
- Ib
- Ib Demand
- Ib THD
- Ic
- Ic Demand
- Ic THD
- In
- In Demand
- In THD
- kVAh (high)
- kVAh (low)
- Negative kvarh (high)
- Negative kvarh (low)
- Negative kWh (low)
- Negative kWh (high)
- P3 (high)
- P3 (low)
- P3 Demand (high)
- P3 Demand (low)
- Pa (high)
- Pa (low)
- Pb (high)
- Pb (low)
- Pc (high)
- Pc (low)
- PF3
- PFa
- PFb
- PFC
- Positive kvarh (high)
- Positive kvarh (low)
- Positive kWh (high)
- Positive kWh (low)
- Q3 (high)
- Q3 (low)
- Q3 Demand (high)
- Q3 Demand (low)
- Qa (high)
- Qa (low)
- Qb (high)
- Qb (low)
- Qc (high)
- Qc (low)
- S3 (high)
- S3 (low)
- S3 Demand (high)
- S3 Demand (low)
- Sa (low)
- Sa (high)
- Sb (high)
- Sb (low)
- Sc (high)
- Sc (low)
- Switches and Relays States
- Time - Hours/Minutes
- Time - Seconds
- V Unbalance
- Vab (high)
- Vab (low)
- Vab THD
- Van (high)
- Van (low)
- Van THD
- Vbc (high)
- Vbc (low)
- Vbc THD
- Vbn (high)
- Vbn (low)
- Vbn THD
- Vca (high)
- Vca (low)
- Vcn (high)
- Vcn (low)
- Vcn THD

a) ACCESS TO EVENT RECORDER INFORMATION:

There are two ways to access Event Recorder Information:

- Access only the Records and data you wish to view
- Access the entire Event Record.

The Event Recorder is indexed by Event Number (1 to 40). To access a specific Event, the Event Number must be written to the PQM memory map location 12C0h. The data specific to that Event can be read starting at memory map location 0AE0h. The specific Event Number must be known to read the Event Recorder in this fashion. However, this Event Number is usually not known and the entire Event Record must be read. The easiest way to do this is to read the PQM Memory Map location 0AD0h (Total Number of Events Since Last Clear) and loop through each Event Number indicated by the value from 0AD0h, reading the associated data pertaining to each Event. This requires from 1 to 40 serial reads of 170 bytes each. Once this data is obtained, it can be interpreted based upon the format of each value as described in Section 7.3: MODBUS MEMORY MAP. It is important to note that some memory map parameters are 32 bits (4 bytes) long and require 2 registers to contain their value, one for the two high bytes and one for the two low bytes.

A.1.2 INTERFACING USING HYPERTERMINAL

APPLICATION NOTE PQMAN02: INTERFACING WITH THE PQM USING HYPERTERMINAL

When upgrading firmware, the PQM may appear to lockup if there is an interruption on the communication port during the upload process. If the PQM does not receive the necessary control signals for configuration during firmware upload, it could remain in a halted situation until reinitialized. The steps used by PQMPC to upload firmware to the PQM are as follows:

1. Prepare the PQM for firmware upgrade by saving setpoints to a file.
2. Erase the flash memory and verify it to be blank.
3. Upload the new firmware.
4. Verify the CRC when upload is complete.
5. Run the new code.

If the PQM is interrupted prior to erasing the flash memory, it could be halted in a mode where the display will read:

```
PQM FLASH LOADER
ENTER TEXT "LOAD"
```

If the computer being used to upload firmware has a screen saver enabled, and the screen saver operates during the upload process, the communication port will be interrupted during the launch of the screen saver. *It is recommended to disable any screen saver prior to firmware upload.*

There are two ways to alleviate this condition: one is to cycle power to the PQM; the second is to interface with the PQM using a terminal program, such as Hyperterminal, and perform the upload process manually.

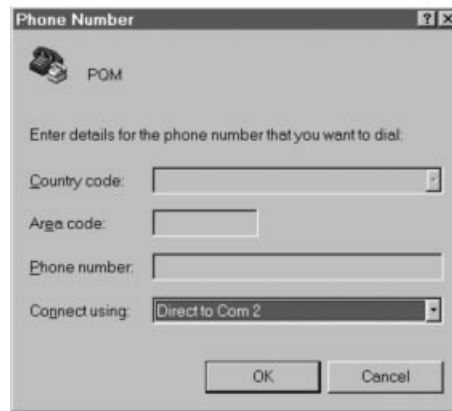
a) CYCLING POWER

Remove and then re-apply control power to the PQM. The PQM should then run the existing firmware in its flash memory. If the PQM does not run the firmware in flash memory, attempt the second method using Hyperterminal.

b) HYPERTERMINAL

1. Hyperterminal is a terminal interface program supplied with Microsoft Windows. Run the program "hypertrm.exe" which is usually located in the Accessories folder of your PC.
2. A Connection Description window will appear asking for a name, use a name such as PQM for the connection and click on OK.

- The following window appears.

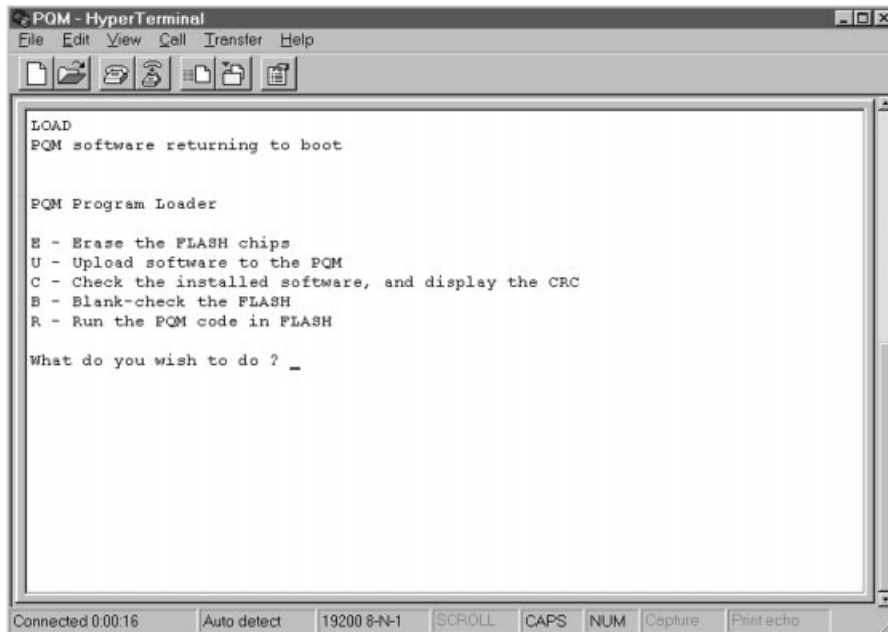


- Select the communications port of your PC that is connected to the PQM and click on OK.
- The following window appears next.

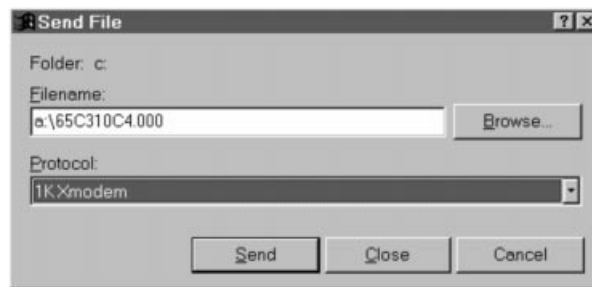


- Change the settings in the Properties window to match those shown above, and click on OK. You should now have a link to the PQM. Enter the text LOAD in uppercase in the text window of Hyperterminal.

7. The PQM Boot Menu should appear in the text window.



8. Type “E” to Erase the PQM flash memory.
9. The PQM verify that you wish to erase the flash memory; enter “Y” for yes. The Boot Menu appears again when complete.
10. Now select “B” to blank check the flash memory; the PQM Boot Menu will appear again when complete.
11. Type “U” to upload software to the PQM. The PQM is now waiting for you to send a firmware file. Select “Transfer” and then “Send File” on the Hyperterminal task bar and the following window will appear.



12. Enter the location and the name of the firmware file you wish to send to the PQM, and ensure the Protocol is **1KXmodem**, and click on **Send**. The PQM will now proceed to receive the firmware file, this usually takes 3 to 4 minutes. When complete the Boot Menu will again appear.
13. Type “C” to check the installed firmware, and then type “R” to run the flash. If the CRC check is bad, erase the flash and re-install the firmware. If numerous bad CRC checks are encountered, it is likely that the file you are attempting to load is corrupted. Obtain a new file and try again. If attempts to use Hyperterminal are unsuccessful, consult the factory.

APPLICATION NOTE PQMAN03: PQM PHASORS IMPLEMENTATION

The purpose of the function Calc_Phasors within the PQM firmware is to take a digitally sampled periodic signal and generate the equivalent phasor representation of the signal. In the conventional sense, a phasor depicts a purely sinusoidal signal which is what we're interested in here; we wish to calculate the phasor for a given signal at the fundamental power system frequency. The following Discrete Fourier Series equations calculate the phasor in rectangular co-ordinates for an arbitrary digitally sampled signal. The justification for the equations is beyond the *scope* of this document but can be found in some form in any text on signal analysis.

$$\text{Re}(g) = \frac{2}{n} \sum_{n=0}^{N-1} g_n \cdot \cos(\omega_0 n T) ; \quad \text{Im}(g) = \frac{2}{n} \sum_{n=0}^{N-1} g_n \cdot \sin(\omega_0 n T)$$

where: $\text{Re}(g)$ = Real component of phasor
 $\text{Im}(g)$ = Imaginary component of phasor
 g = Set of N digital samples = $\{g_0, g_1, \dots, g_{N-1}\}$
 g_n = nth sample from g
 N = Number of samples
 f_0 = Fundamental frequency in Hertz
 $\omega_0 = 2\pi f_0$ = Angular frequency in radians
 $T = 1 / (f_0 N)$ = Time between samples

The PQM Trace Memory feature is employed to calculate the phasors. The Trace Memory feature samples 16 times per cycle for two cycles for all current and voltage inputs. Substituting $N = 16$ (samples/cycle) into the equations yields the following for the real and imaginary components of the phasor:

$$\text{Re}(g) = \frac{1}{8} \left(g_0 \cos 0 + g_1 \cos \frac{\pi}{8} + g_2 \cos \frac{2\pi}{8} + \dots + g_{31} \cos \frac{31\pi}{8} \right)$$

$$\text{Im}(g) = \frac{1}{8} \left(g_0 \sin 0 + g_1 \sin \frac{\pi}{8} + g_2 \sin \frac{2\pi}{8} + \dots + g_{31} \sin \frac{31\pi}{8} \right)$$

The number of multiples in the above equation can be reduced by using the symmetry inherent in the sine and cosine functions which is illustrated as follows:

$$\cos \phi = -\cos(\pi - \phi) = -\cos(\pi + \phi) = \cos(2\pi - \phi)$$

$$\sin \phi = \sin(\pi - \phi) = -\sin(\pi + \phi) = -\sin(2\pi - \phi)$$

$$\cos \phi = \sin\left(\frac{\pi}{2} - \phi\right)$$

Let $k_1 = \cos(\pi/8)$, $k_2 = \cos(\pi/4)$, $k_3 = \cos(3\pi/8)$; the equations for the real and imaginary components are reduced to:

$$\text{Re}(g) = \frac{1}{8} (k_1(g_1 - g_7 - g_9 + g_{15} + g_{17} - g_{23} - g_{25} + g_{31}) + k_2(g_2 - g_6 - g_{10} + g_{14} + g_{18} - g_{22} - g_{26} + g_{30}) + k_3(g_3 - g_5 - g_{11} + g_{13} + g_{19} - g_{21} - g_{27} + g_{29}) + (g_0 - g_8 + g_{16} - g_{24}))$$

$$\text{Im}(g) = \frac{1}{8} (k_1(g_3 + g_5 - g_{11} - g_{13} + g_{19} + g_{21} - g_{27} - g_{29}) + k_2(g_2 + g_6 - g_{10} - g_{14} + g_{18} + g_{22} - g_{26} - g_{30}) + k_3(g_1 + g_7 - g_9 - g_{15} + g_{17} + g_{23} - g_{25} - g_{31}) + (g_4 - g_{12} + g_{20} - g_{28}))$$

A

The number of subtractions can be reduced between the calculations of real and imaginary components by not repeating the same subtraction twice. The following subtractions are repeated:

$$\begin{array}{llll} \Delta_0 = g_0 - g_8 & \Delta_1 = g_1 - g_9 & \Delta_2 = g_2 - g_{10} & \Delta_3 = g_3 - g_{11} \\ \Delta_4 = g_4 - g_{12} & \Delta_5 = g_5 - g_{13} & \Delta_6 = g_6 - g_{14} & \Delta_7 = g_7 - g_{15} \\ \Delta_8 = g_{16} - g_{24} & \Delta_9 = g_{17} - g_{25} & \Delta_{10} = g_{18} - g_{26} & \Delta_{11} = g_{19} - g_{27} \\ \Delta_{12} = g_{20} - g_{28} & \Delta_{13} = g_{21} - g_{29} & \Delta_{14} = g_{22} - g_{30} & \Delta_{15} = g_{23} - g_{31} \end{array}$$

Substituting in the above 'delta' values results in the form of the equations that will be used to calculate the phasors:

$$\text{Re}(g) = \frac{1}{8}(\Delta_0 + \Delta_8 + k_1(\Delta_1 - \Delta_7 + \Delta_9 - \Delta_{15}) + k_3(\Delta_3 - \Delta_5 + \Delta_{11} - \Delta_{13}))$$

$$\text{Im}(g) = \frac{1}{8}(\Delta_4 + \Delta_{12} + k_1(\Delta_3 + \Delta_5 + \Delta_{11} + \Delta_{13}) + k_2(\Delta_1 + \Delta_7 + \Delta_9 + \Delta_{15}))$$

A.1.4 TRIGGERED TRACE MEMORY RESOLUTION

A

APPLICATION NOTE PQMAN04: TRIGGERED TRACE MEMORY RESOLUTION

The Triggered Trace Memory can be used to detect and record system disturbances. The PQM uses a dedicated continuous sampling rate of 16 samples per cycle to record fluctuations in voltage or current as per user defined levels. The PQM calculates the true RMS value of one consecutive cycle, or 16 samples, and compares this value with the user-defined trigger levels to determine if it will record all sampled waveforms. The sampled waveforms include Ia, Ib, Ic, In, Va, Vb and Vc.

The PQM uses the following method to calculate the true RMS value of all sampled waveforms

$$\text{TrueRMS} = \sqrt{\frac{1}{n} \sum_{1 \rightarrow n} V_n^2}$$

where: n = the number of samples
 V = the sampled waveform

To find the minimum disturbance that the PQM can detect, we first solve for the magnitude of any one sample. Deriving from the RMS calculation above, we have:

$$V_k = \sqrt{n \cdot \text{TrueRMS}^2 - \left(\sum_{1 \rightarrow k-1} V_{k-1}^2 + \sum_{1 \rightarrow k+1} V_{k+1}^2 \right)}$$

where: n = the number of samples
 V = the sampled waveform
 k = the sample to be extracted

The PQM Triggered Trace Memory has a minimum step value of 1% on the user defined level. Therefore, we find the minimum magnitude of any sample that can be detected. Using the individual sample equation above:

$$V_{k2} = \sqrt{n \cdot (0.99 \cdot \text{TrueRMS})^2 - \left(\sum_{1 \rightarrow k-1} V_{k-1}^2 + \sum_{1 \rightarrow k+1} V_{k+1}^2 \right)}$$

where: n = the number of samples
 k = the sample number to be extracted
 V_{k2} = the calculated value of the extracted sample that deflects the overall value of the RMS calculation by 1%

Knowing this, we can conclude that any individual sample that is equal to V_{k2} in the array of the 16 samples evaluated by the PQM is the minimum disturbance that can be detected by the Triggered Trace Memory. The PQM uses the fundamental RMS value based on the VT Secondary setpoint as the reference for the user-defined Trace Memory trigger levels.

The minimum duration of a disturbance is determined by the sample rate. At 16 samples per cycle, the time between samples is based upon the system frequency and is determined as follows:

$$t_s = \frac{1/f}{16}$$

where: t_s = the time between samples
 f = the system frequency as determined by the PQM metering functions

The PQM can determine frequencies from 20 to 80 Hz. Therefore, the minimum duration of a detectable disturbance is 0.783 ms at 80 Hz. The minimum duration of a detectable disturbance is 3.125 ms at 20 Hz.

A

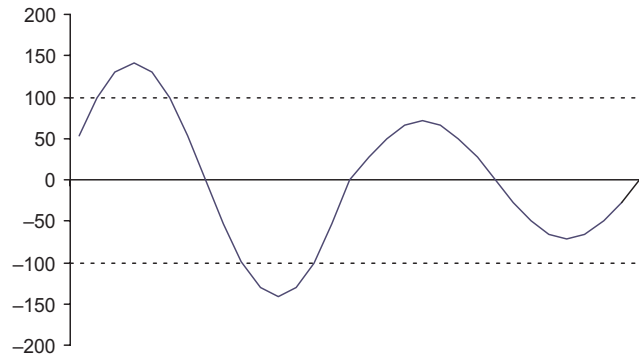
Any sample that deviates from its corresponding reference sample by the amount V_{k2} for a minimum duration as calculated above is detectable by the PQM.

EXAMPLE:

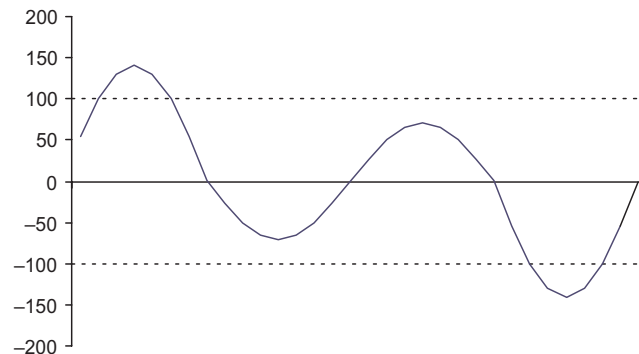
Consider a PQM with a nominal VT secondary voltage of 100 V into the VT inputs. The trace memory trigger for a Phase A undervoltage level of 90 V is enabled with the following setpoints:

S2 SYSTEM SETUP \ CURRENT/VOLTAGE CONFIGURATION \ VT NOMINAL SECONDARY VOLTAGE: 100 V
S1 PQM SETUP \ TRACE MEMORY \ VA UNDERVOLTAGE TRIG LEVEL: 90%

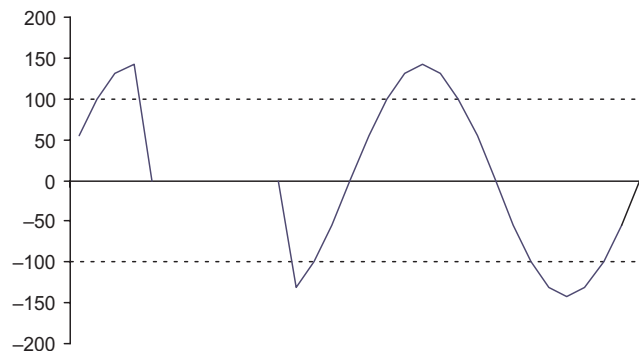
In the waveform below, an undervoltage fault occurs in the second cycle. The first cycle RMS voltage is 100 V; the second cycle RMS voltage is 50 V, triggering the trace memory feature for the settings above.



A centered one cycle undervoltage fault is shown below. In this case, the first cycle RMS voltage is 79.05 V and the second cycle RMS voltage is 79.05 V, triggering the trace memory feature for the settings above.



A half cycle undervoltage fault is shown below. In this case, the first cycle RMS voltage is 70.07 V and the second cycle RMS voltage is 100 V, triggering the trace memory feature for the settings above.



A.1.5 PULSE OUTPUT APPLICATION

A

APPLICATION NOTE PQMAN05: PULSE OUTPUT APPLICATION

Up to 4 SPDT Form C output relays are configurable as Pulse Initiators based on energy quantities calculated by the PQM. Variables to consider when using the PQM as a Pulse Initiator are:

- PQM Pulse Output Parameter
 - PQM Pulse Output Interval
 - PQM Pulse Output Width
 - PQM Output Relay Operation
 - Pulse Acceptance Capability of the end receiver
 - Type of Pulse Receiver
 - Maximum Energy Consumed over a defined interval
1. **PQM Pulse Output Parameter:** The PQM activates the assigned output relay based upon the energy quantity used as the base unit for pulse initiation. These energy quantities include \pm kWhr, \pm kVARh, and kVAh.
 2. **PQM Pulse Output Interval:** The PQM activates the assigned output relay at the accumulation of each Pulse Output Interval as defined by the user. This interval is based upon system parameters such that the PQM pulse output activates at a rate not exceeding the Pulse Acceptance Capability of the end receiver.
 3. **PQM Pulse Output Width:** This user defined parameter defines the duration of the pulse initiated by the PQM when a quantity of energy equal to the Pulse Output Interval has accumulated. It is based upon system parameters such that the PQM pulse output will activate for a duration that is within the operating parameters of the end receiver.
 4. **PQM Output Relay Operation:** This user defined parameter defines the normal state of the PQM output relay contacts, i.e. Fail-safe or Non-Failsafe.
 5. **Pulse Acceptance Capability of the end Receiver:** This parameter is normally expressed as any one of the following: (a) Pulses per Demand Interval; (b) Pulses per second, minute or hour; (c) Minimum time between successive closures of the contacts.
 6. **Type of Pulse Receiver:** There are 4 basic types of Pulse receivers: a) Three-wire, every pulse counting; b) Three-wire, every other pulse counting; c) Two-wire, Form A normally open, counts only each contact closure; d) Two-wire, counts every state change, i.e. recognizes both contact closure and contact opening.
 7. **Maximum Energy Consumed over a defined interval:** This is based upon system parameters and defines the maximum amount of energy that may be accumulated over a specific time.

Application of the PQM Pulse Output Relays to an End Receiver Using KYZ Terminals:

Typical end receivers require a contact closure between KY or KZ based upon the type of receiver. The PQM Pulse Output feature can be used with either two- or three-wire connections. The PQM activates the designated Output Relay at each accumulation of the defined Pulse Output Interval for the defined Pulse Output Width. Therefore, each PQM contact operation represents one interval. For end receivers that count each closure and opening of the output contacts, the PQM Pulse Output Interval should be adjusted to match the registration of the end receiver. For example, if the end receiver counts each closure as 100 kWh and each opening as 100 kWh, the PQM Pulse Output Interval should be set to 50 kWh.

The PQM Output Relays can be configured as Failsafe or Non-Failsafe to match the normally open/closed configuration of the KY and KZ connections at the end receiver. The K connection is always made to the COM connection of the designated PQM output relay, and the Y and Z connections can be made to the N/O or N/C connections based upon the type of end receiver.

A.1.6 DATA LOGGER IMPLEMENTATION

APPLICATION NOTE PQMAN06: DATA LOGGER IMPLEMENTATION

The Data Logger allows various user defined parameters to be continually recorded at a user defined rate. The Data Logger uses 64 samples/cycle data. The PQM has allocated 65536 bytes of memory for Data Log storage. The memory structure is partitioned into 512 blocks containing 64×2 byte registers as shown below:

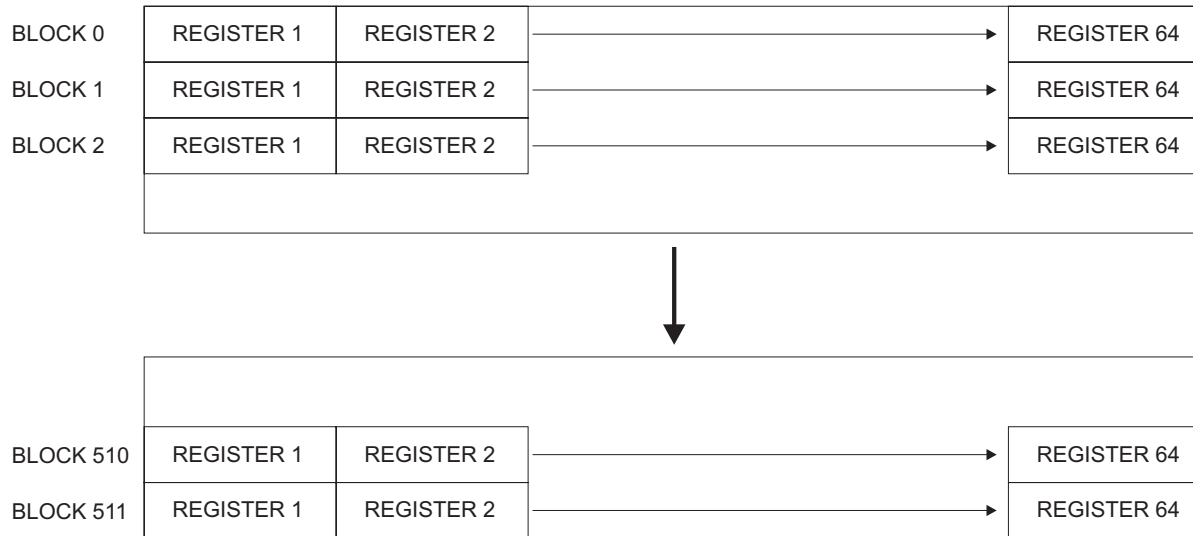


Figure A-1: DATA LOGGER MEMORY STRUCTURE

Each entry into the Data Log is called a Record. The Record can vary in size depending upon the parameters the user wishes to log. The memory structure can also be partitioned into 2 separate Data Logs. The 2 logs can be user defined in size, or can be optimized by PQMPC. The top of each Data Log contains what is called the Header. Each Data Log Header contains the following information...

- **Log Time Interval:** This is the user defined interval at which the data log will store entries
- **Present Log Time and Date:** This is the time and date of the most recent Record
- **Log Start Address:** This is the start address for the beginning of the logged data
- **Log Record Size:** This is the size of each Record entry into the Data Log based upon the user defined Data Log structure
- **Log Total Records:** This is the total number of records available based upon the user defined Data Log parameter structure
- **Log Pointer to First Item of First Record:** This is a pointer to the first record in the Data Log
- **Log Pointer to First Item of Record After Last:** This is a pointer to the next record to be written into the Data Log
- **Log Status:** This reports the current status of the Data Log; i.e.: Running or Stopped
- **Log Records Used:** This is the number of records that have been written into the Data Log
- **Log Time Remaining Until Next Reading:** This is a counter showing how much time remains until the next record is to be written into the Data Log

The Data Logger has 2 modes of operation, **Run to Fill** and **Circulate**. In the Run to Fill mode, the Data Log will stop writing records into the memory structure when there is not enough memory to add another record. Depending on the size of each record, the Data Log may not necessarily use the entire 65536 bytes of storage available. In the Circulate mode, the Data Log will continue to write new Records into the Log beyond the last

available Record space. The Log will overwrite the first Record after the Header and continue to overwrite the Records to follow until the user wishes to stop logging data. The Log will act as a rolling window of data in time, going back in time as far as the amount of records times the Log Time Interval will allow in the total space of memory available.

a) ACCESSING THE DATA LOG INFORMATION

The Data Log can be accessed in two ways: using PQMPC via the serial port. Access via PQMPC is as described in Chapter 6: SOFTWARE. Access is manually via the serial port as follows:

- Set the Block of data you wish to access at 1268h in the PQM Memory Map
- Read the required amount of data from the 64 Registers in the Block

Accessing the Data Log in this manner assumes that the user knows which Block they wish to access, and knows the size of each Record based upon the parameters they have selected to log.

The easiest way to access the data in the Data Log is to read the entire log and export this data into a spreadsheet for analysis (this is the method incorporated by PQMPC). This requires defining the Block to be read, starting at Block 0, and reading all 128 bytes of data in each of the 64 Registers within the Block. You would then define Blocks 1,2,3 etc. and repeat the reading of the 64 Registers for each block, until Block 511. This requires 512 reads of 128 bytes each. The data can then be interpreted based upon the parameter configuration.

b) INTERPRETING THE DATA LOG INFORMATION

Using 2 Data Logs in the Run to Fill mode, the Data Log is configured as follows:

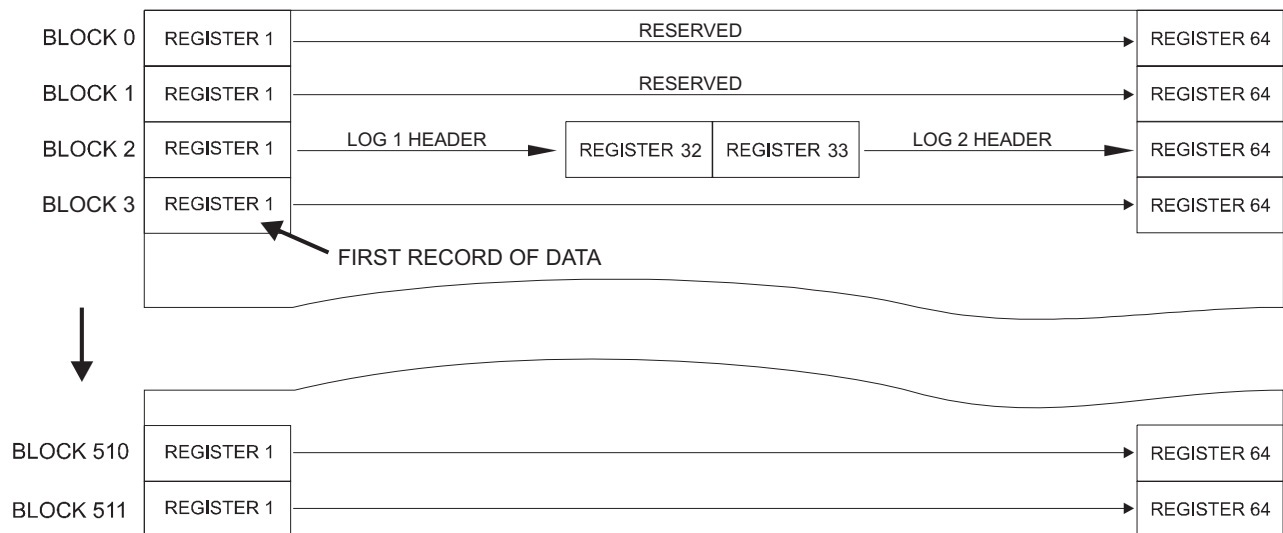


Figure A-2: DATA LOG CONFIGURATION

Blocks 0 and 1 are reserved for Data Logger Data Interval information. Block 2 contains the Header information for both Data Logs. The first 32 registers of Block 2 are reserved for Data Log 1 Header Information, and the remaining 32 registers of Block 2 are reserved for Data Log 2 Header Information. The first register of Data Log information resides at Register 1 of Block 3. This leaves 65152 bytes of storage for data.

The location of the first Record in Log 2 will depend upon the Log configuration. Its location is determined by reading the Log 2 Header value for Log Start Address at location 0AAEh in the memory map and performing a calculation. The Log Start Address is a value from 0 to 65535 representing the first byte of the first Record within the Data Log memory structure. Add 1 to this number and then divide this number by 64 (number of Records in a Block). Then divide this number by 2 (number of bytes in a record), and truncate the remainder of

the division to determine the Block number. Multiplying the remainder of the division by 64 will determine the Register number. For example, if the Log Start Address was 34235, then the Block and Register numbers containing the first record of Log 2 are:

$$\text{Block Number} = (34235 + 1) / 64 / 2 = 267.46875$$

Therefore, Block Number 267 contains the starting record.

$$\text{Record Number} = 0.46875 \times 64 = 30$$

Therefore, Register Number 30 contains the first byte of Log 2 data

This location will always be the starting address for Data Log 2 for the given configuration. Adding or deleting parameters to the configuration will change the Log 2 Starting Address.

The Data Logs will use the maximum amount of memory available, minus a 1 record buffer, based upon the user configuration. For Example, if the Record Size for a given configuration was 24 bytes, and there were 40 bytes of memory left in the memory structure, the Data Logger will not use that last 40 bytes, regardless of the mode of operation. The Data Logger uses the following formula to determine the total record space available...

$$\text{Total Space} = (65152 / \text{Record Size}) - 1$$

As in the example, the total space calculated would be $65152 / 24 - 1 = 2713.67$. This equates to 2713 records with 40 bytes of unused memory at the end of Block 511. The total amount of space used in the structure can also be found in the Log Header in the Log Total Records field.

When the Data Log is configured for **Circulate** mode of operation, the memory structure is the same as for Run to Fill mode. To read the Log data, you must use the Log Starting Address, Log Record Size, and Log Total Records information from each of the Log Headers. The Log Starting Address for Log 2 can be determined as shown in the previous calculation for Run to Fill mode. The total space occupied in the memory structure by either log is determined by multiplying the Log Total Records by the Log Record Size and adding this value to the Log Starting Address. It is important to note that the Log Starting Address is always referenced to the first Register of Block 0, or the first byte of the Data Log memory structure.

Once you have separated the Data Log data from the Headers, you will then need to interpret the data into a structured format. Each Record is comprised of user-defined parameters. These parameters are implemented into the user-defined structure in a specific order based upon selection into either, or both, Data Log(s). Address 1270h in the PQM Memory Map is the Holding Register for the first available parameter for use by the Data Logs. The Data Logs will place the user-selected parameters into their respective Record structures based upon their respective order in the PQM Memory Map. For example, if Positive kWh, Frequency and Current Unbalance were selected to be measured parameters, they would be placed into the Record structure in the following order:

Unbalance	2 bytes	(16 bit value)
Frequency	2 bytes	(16 bit value)
Positive kWh	4 bytes	(32 bit value)

The DATA LOG PARAMETERS table on the following page illustrates the order of parameters and their size.

Therefore, the Record size would be 8 bytes. To put a time value associated with each Record, you must read the Log Time and Date from the Header. This is the time of the last Record in the Log. To time stamp the first Record used, multiply the Log Time Interval by the Log Records Used and subtract this number from the time associated with the last Record. To determine the time associated with any Record, add the Log Time Interval times the Record to be read to the time associated with the first Record in the Log.

For example:

Log Time Interval:	3600
Log Time - Hours/Minutes:	02 30
Log Time - Seconds:	30300

Log Date - Month: 06 15
 Log Date - Year: 1997
 Log Records Used: 1600

The last Record entry time is interpreted as 2:30 AM, 30.300 seconds, June 15, 1997. The Log Time Interval is 3600 seconds, or 1 hour. Taking the Log Records Used (1600) and multiplying this by the Log Time Interval(3600) gives 5760000 seconds. This translates into 66 days and 16 hours. Subtracting backwards on a calendar from the time for the last Record gives a time and date of 10:30:30.000AM, April 9, 1997. This is the time stamp for the first Record. Time stamping the remaining Records requires adding 3600 seconds for each Record starting from the time associated with the first Record. It is important to note that when in the Circulate mode, and the Data Log fills the available memory, the Log wraps around the first available Register of the memory structure and the Log Pointer to First Item of First Record will float along in time with each additional entry into the Log. For example, if the Data Log has wrapped around the available memory more than once, the Log Pointer to First Item of First Record will always be preceded in memory by the Log Pointer to First Item of Record After Last. As each new entry is written into the Log, these two pointers move down to the next record space in memory, overwriting the first entry into the log as of the Present Log Time and Date.

c) DATA LOG PARAMETERS

Listed below are the parameters available for capturing data via the Data Logger. Note that these parameters will be placed within the Record structure of the Data Log in the order and size that they appear in this table.

Table A-1: DATA LOG PARAMETERS

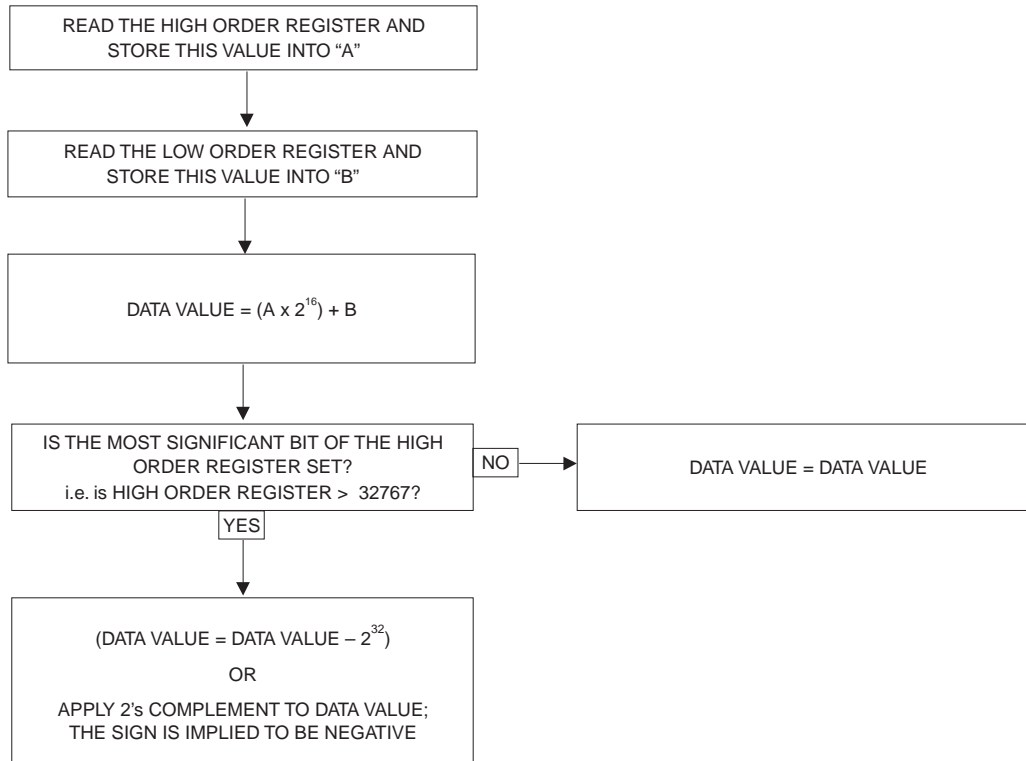
DATA LOG PARAMETER	SIZE (BYTES)	DATA LOG PARAMETER	SIZE (BYTES)	DATA LOG PARAMETER	SIZE (BYTES)
Ia	2	PFa	2	kVAh	4
Ib	2	Pb	4	Ia Demand	2
Ic	2	Qb	4	Ib Demand	2
Iavg	2	Sb	4	Ic Demand	2
In	2	PFb	2	In Demand	2
I Unbalance	2	Pc	4	P3 Demand	4
Van	4	Qc	4	Q3 Demand	4
Vbn	4	Sc	4	S3 Demand	4
Vcn	4	PFc	2	Ia THD	2
Vpavg	4	P3	4	Ib THD	2
Vab	4	Q3	4	Ic THD	2
Vbc	4	S3	4	In THD	2
Vca	4	PF3	2	Van THD	2
Vlavg	4	Frequency	2	Vbn THD	2
V Unbalance	2	Positive kWh	4	Vcn THD	2
Pa	4	Negative kWh	4	Vab THD	2
Qa	4	Positive kvarh	4	Vbc THD	2
Sa	4	Negative kvarh	4	Analog Input	4

where: I = current; V = Voltage; P = Real Power; Q = Reactive Power; S = Apparent Power; PF = Power Factor
 THD = Total Harmonic Distortion

A.1.7 READING LONG INTEGERS FROM MEMORY MAP

APPLICATION NOTE PQMAN08: READING LONG INTEGER VALUES FROM THE MEMORY MAP

The PQM memory map contains some data which is formatted as a long integer type, or 32 bits. Because the Modbus Protocol maximum register size is 16 bits, the PQM stores long integers in 2 consecutive register locations, 2 high order bytes, and 2 low order bytes. The data can be retrieved by the following logic:

**EXAMPLE:**

1. Reading a positive 3 Phase Real Power actual value from the PQM:

REGISTER	ACTUAL VALUE (hex)	DESCRIPTION	UNITS & SCALE	FORMAT
02F0	004F	3 Phase Real Power (high)	0.01 x kW	F4
02F1	35D1	3 Phase Real Power (low)	0.01 x kW	F4

2. Following the method described above:

$$\begin{aligned}
 \text{DATA VALUE} &= (004F \times 2^{16}) + 35D1 && \text{hexadecimal} \\
 &= 5177344 + 13777 && \text{converted to decimal} \\
 &= 5191121 && \text{decimal}
 \end{aligned}$$

The most significant bit of the High Order register is not set, therefore the DATA VALUE is as calculated. Applying the Units & Scale parameters to the DATA VALUE, we multiply the DATA VALUE by 0.01 kW. Therefore the resultant value of 3 Phase Real Power as read from the memory map is 51911.21 kW.

3. Reading a negative 3 Phase Real Power actual value from the PQM:

REGISTER	ACTUAL VALUE (hex)	DESCRIPTION	UNITS & SCALE	FORMAT
02F0	FF3A	3 Phase Real Power (high)	0.01 x kW	F4
02F1	EA7B	3 Phase Real Power (low)	0.01 x kW	F4

4. Following the method described above:

$$\begin{aligned}
 \text{DATA VALUE} &= (\text{FF3A} \times 2^{16}) + \text{EA7B} && \text{hexadecimal} \\
 &= (65338 \times 2^{16}) + 60027 && \text{converted to decimal} \\
 &= 4282051195 && \text{decimal}
 \end{aligned}$$

5. The most significant bit of the High Order register is set, therefore the DATA VALUE is calculated as:

$$\begin{aligned}
 \text{DATA VALUE} &= \text{DATA VALUE} - 2^{32} \\
 &= 4282051195 - 4294967296 \\
 &= -12916101
 \end{aligned}$$

Applying the Units & Scale parameters to the DATA VALUE, we multiply the DATA VALUE by 0.01 kW. Therefore the resultant value of 3 Phase Real Power as read from the memory map is -129161.01 kW.

APPLICATION NOTE PQMAN09: PULSE INPUT APPLICATION

The PQM has up to 4 Logical Switch Inputs that can be configured as Pulse Input Counters. Variables to consider when using the PQM as a Pulse Input Counter are:

- PQM Switch Input A/B/C/D Function
 - PQM Switch Input A/B/C/D Activation
 - PQM Switch Input A/B/C/D Name
 - PQM Pulse Input (Units)
 - PQM Pulse Input A/B/C/D (Value)
 - PQM Totalized Pulse Input
1. PQM Switch Input A/B/C/D Function: This parameter defines the functionality to be provided by the PQM Switch Input. For use as a Pulse Input Counter, the PQM Switch Input to be used must be assigned as either Pulse Input 1/2/3 or 4.
 2. PQM Switch Input A/B/C/D Activation: This parameter is set to OPEN or CLOSED. The PQM will see the operation of the Switch Input in the state as defined by this parameter.
 3. PQM Switch Input A/B/C/D Name: This parameter defines the name given to each of the Switch Inputs used. It is used as a label only and has no bearing on the operation of the Switch Input.
 4. PQM Pulse Input (Units): This parameter is the name given to the base units that the PQM Pulse Input(s) will be counting. It is used as a label only and has no bearing on the operation of the Pulse Input.
 5. PQM Pulse Input A/B/C/D Value: This is the value assigned to each counting operation as determined by the Switch Input.
 6. PQM Totalized Pulse Input: This parameter creates a summing register of the various Pulse Inputs configured. It can be configured for any combination of the PQM Switch Inputs used as Pulse Inputs.

Application of the PQM Pulse Input(s) With a Pulse Initiator Using KYZ Terminals:

Typical end receivers require a contact closure between **KY** or **KZ** based upon the type of receiver. Because of the multi-functional parameters of the PQM Switch Inputs, the PQM Switch Inputs are not labeled with **KYZ** markings as a dedicated pulse input device. However, the PQM can still be used as a pulse counter. The PQM Switch Inputs require a signal from the PQM Switch Common terminal to be activated. The PQM configured as a Pulse Counter can be used with Two-Wire Pulse Initiators. The Pulse Initiator must provide a dry contact operation. The Switch Common terminal of the PQM is connected to the **K** terminal of the Pulse Initiator. The PQM Switch Input assigned to count pulses can be connected to the **Y** or the **Z** terminal of the Pulse Initiator, depending on the operation of the Pulse Initiator, i.e. OPEN or CLOSED. The PQM Pulse Input (Value) must be assigned to match the pulse value of the Pulse Initiator, i.e. if the Pulse Initiator delivers a dry contact closure for every 100kWh, the PQM Pulse Input (Value) must also be set to 100.

Various operating parameters with regard to the PQM Switch Inputs must be taken into account. The PQM Switch Inputs require a minimum 100ms operation time to be detected. The duration of the contact operation can be indefinite. The internal Switch Input circuit of the PQM is itself switched on and off at the times when the PQM is reading the status of the Switch Inputs. Monitoring the input to one of the PQM Switch Inputs will reveal a pulsed 24VDC waveform, not a constant signal. Standard wiring practice should be adhered to when making connections to the PQM Switch Inputs, i.e. avoiding long runs of cable along current carrying conductors or any other source of EMI. An induced voltage on the Switch Input can cause malfunction of the Switch Input.

A.1.9 PULSE TOTALIZER APPLICATION

A

APPLICATION NOTE PQMAN10: PULSE INPUT APPLICATION FOR USE AS A PULSE TOTALIZER

The PQM has up to 4 Logical Switch Inputs that can be configured as Pulse Input Counters. One common application of these Pulse Inputs is their use as an energy totalizer for more than one circuit. One PQM can totalize input from up to 4 different sources and sum these results into a single register. Variables to consider when using the PQM as a Pulse Input Counter are:

- PQM Switch Input A/B/C/D Function
 - PQM Switch Input A/B/C/D Activation
 - PQM Switch Input A/B/C/D Name
 - PQM Pulse Input (Units)
 - PQM Pulse Input A/B/C/D (Value)
 - PQM Totalized Pulse Input
1. PQM Switch Input A/B/C/D Function: This parameter defines the functionality to be provided by the PQM Switch Input. For use as a Pulse Input Counter, the PQM Switch Input to be used must be assigned as either Pulse Input 1/2/3 or 4.
 2. PQM Switch Input A/B/C/D Activation: This parameter is set to OPEN or CLOSED. The PQM will see the operation of the Switch Input in the state as defined by this parameter.
 3. PQM Switch Input A/B/C/D Name: This parameter defines the name given to each of the Switch Inputs used. It is used as a label only and has no bearing on the operation of the Switch Input.
 4. PQM Pulse Input (Units): This parameter is the name given to the base units that the PQM Pulse Input(s) will be counting. It is used as a label only and has no bearing on the operation of the Pulse Input.
 5. PQM Pulse Input A/B/C/D Value: This is the value assigned to each counting operation as determined by the Switch Input.
 6. PQM Totalized Pulse Input: This parameter creates a summing register of the various Pulse Inputs configured. It can be configured for any combination of the PQM Switch Inputs used as Pulse Inputs.

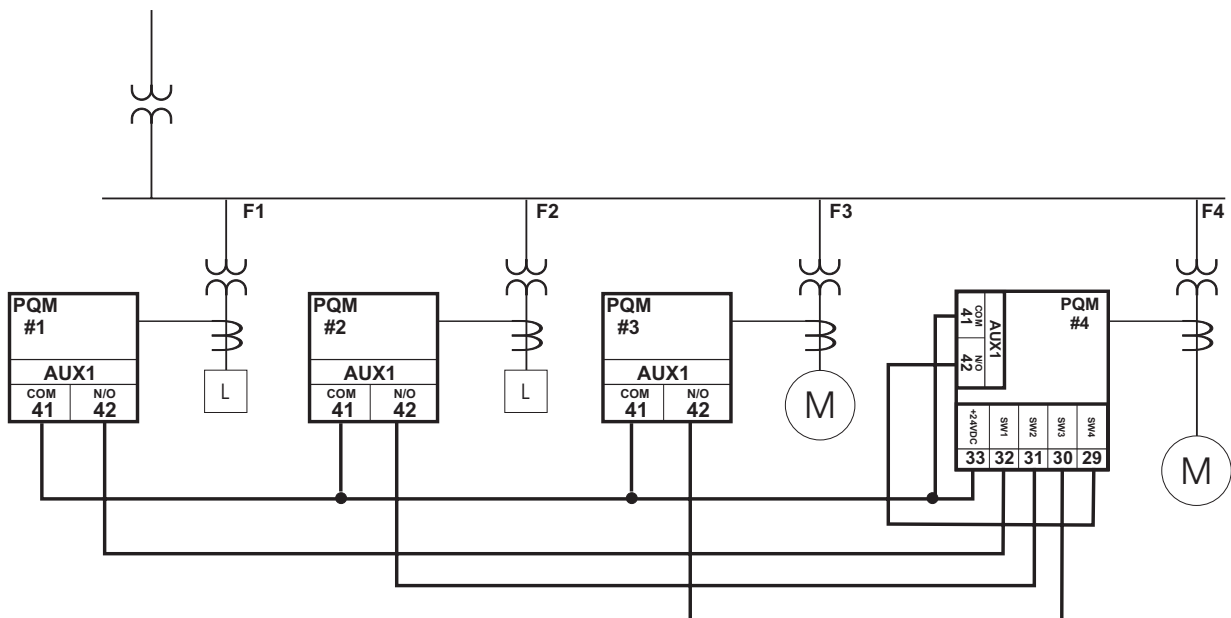
Configuring the PQM to Totalize Energy From Multiple Metering Locations:

Figure A-3: MULTIPLE METERING LOCATIONS

A

The diagram above shows an example of a PQM being used to totalize the energy from 4 other PQMs. PQMs 1 through 4 have each of their respective AUX1 relays configured for Pulse Output functionality (refer to the Pulse Output PQM Application note for details, or the PQM manual). The Switch Common output from PQM#4 is fed to the common contact of the AUX1 relays on PQMs 1 through 4. The N/O contact of AUX1 for PQMs 1 through 4 will operate based upon the setup as described in the Pulse Output functionality section of the PQM manual. The Totalized Pulse Input register of PQM#4 can be set to sum the counts from Switch Inputs 1 through 4, thus giving a total energy representation for the 4 metering locations. The count value for each Pulse Input on PQM#4 can be set to match the Pulse Output Interval as programmed on each PQM. For example, if PQM#1 had a Pulse Output Interval = 100 kWhr, and PQM#2 had a Pulse Output Interval = 10 kWhr, then Pulse Input 1 on PQM#4 would have the Pulse Input Value set for 100 and Pulse Input 2 on PQM#4 would have the Pulse Input Value set for 10.

Various operating parameters with regard to the PQM Switch Inputs must be taken into account. The PQM Switch Inputs require a minimum 100 ms operation time to be detected. Therefore the Pulse Output Width should be equal to or greater than 100 ms. The duration of the contact operation can be indefinite. The internal Switch Input circuit of the PQM is itself switched on and off at the times when the PQM is reading the status of the Switch Inputs. Monitoring the input to one of the PQM Switch Inputs will reveal a pulsed 24 V DC waveform, not a constant signal. Standard wiring practice should be adhered to when making connections to the PQM Switch Inputs, i.e. avoiding long runs of cable along current carrying conductors or any other source of EMI. An induced voltage on the Switch Input can cause malfunction of the Switch Input.