# iPM Software Interface Requirements

|  |  |
| --- | --- |
| v20221024 J.Carnes | Initial Draft |
| v20221028 J.Carnes | Added Application Utilities |
| V20230509 J.Carnes | Moved OFF/RESET cycle to end of initialization |
| V20230512 J.Aquino | Remove AC Phase Angle from RECORD? Variable list. |
| v20230628 J.Carnes | Updated supported query rate ranges. Added fault case for bad data received. Added information about query response timing. |
| V20230706 J.Aquino | Update XML spec to match code |

# iPM Device Summary

The information in this section summarizes functionality detailed in the iPM product datasheet and programming guide.



## Physical Interface

* RS-485, up to 115.2 kbaud, 8n1, half-duplex, 2-wire.
* Multi-drop connections with multiple devices accessing same 2-wire data bus

## Queries and Addressing

* All device data is queried with ASCII string lines, terminated with ASCII linefeed (0x0A). No information is output on a free-running period. Linefeeds are indicated below as <LF>.
* Devices are addressed by broadcasting the “ADR <address>” command to the bus to set the active device listener prior to sending a query or command. The value of <address> can be 0 through 255 integer values. The device with the corresponding address will become active but will not immediately respond, therefore, an unsuccessful addressing attempt (and subsequent attempts to query that address) will have no response. A device remains active until an “ADR” command is received with a non-matching address. An active device holds the physical bus while transmitting data, and releases bus after completing transmission.

## Device Responses

* Responses from normal running queries are composed of a short ASCII string header terminated with <LF>, immediately followed by the byte data payload. The header is composed of only the data payload size, an integer.
* The byte data payload is NOT appended with a linefeed.
* The ASCII header uniquely identifies the type of query/response but does not identify the device address.

# Application Requirements

Information in this section specifies software interfacing requirements to be satisfied to meet the application of power monitoring in the NCAR RAF GV and C-130 aircraft and integration with the ADS data acquisition system.

## Physical Interface

* Multiple devices shall share the 2-wire half-duplex bus as a multi-drop application. Bus contention must be handled at the application layer.
* Baud rate = 115.2 kbaud
* The RS-485 bus shall interface to a DSM serial port, either a serial card port or native CPU port
* Four (4) devices will be installed in the GV, with two (2) separate multi-drop busses.

## Device Configuration

The following device parameters shall be configured prior to installation using the specified Configuration Commands. Modification of these parameters after installation shall be performed manually via a standard terminal command line. The NIDAS sensor profile shall not implement any Configuration Commands.

The “SETCONFIG ON” command must be sent prior to performing other configuration commands.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Configuration Command (ASCII)** |
| Baud Rate | 115.2 kbaud | “SETBAUD 0<LF>” |
| Multi-Drop Addressability | ON | “SETMULTIDROP ON<LF>” |
| Address | [0,1,2,3,4,5,6,7] | “SETADDRESS <address><LF>” |
| Performance Limits | TBD | TBD |

## Queries and Addressing

* Normal operation consists of the following queries and commands at the given sample period.
* The NIDAS sensor profile shall accommodate up to 8 device address (0-7) on one physical bus. Addresses outside this range shall be ignored.
* Addressing commands are queued prior to queries ad hoc. They can be pre-pended to all queries or only as needed.
* All queries to all addresses must be queued by a single state machine to avoid race conditions and bus conflicts. As a result, query/response delays from one device may delay samples from other devices.

|  |  |  |
| --- | --- | --- |
| **Query Type** | **Query ASCII String** | **Sample Period** |
| Device Address Selection | “ADR<SPACE><address><LF>” | As needed |
| Device Measurement | “MEASURE?<LF>” | 0.2 – 1sec range, 1 second typical |
| Device Status | “STATUS?<LF>” | 0.2 – 1sec range, 1 second typical |
| Device Statistics | “RECORD?<LF>” | 1 – 30 min range, 10 minutes typ |

## Device Responses

* Normal operation consists of the following query responses

|  |  |  |
| --- | --- | --- |
| **Query String** | **Response** | **Query + Response**  **Min. Transfer Time** |
| “MEASURE?<LF>” | “34<LF>”{MeasureData\_34Bytes} | 46 / 11520 = 4.0 ms |
| “STATUS?<LF>” | “12<LF>”{StatusData\_12Bytes} | 23 / 11520 = 2.0 ms |
| “RECORD?<LF>” | “68<LF>”{RecordData\_64Bytes}{CRC\_4Bytes} | 79 / 11520 = 6.9 ms |
| “ADR<SPACE>  <address\_int><LF>” | None |  |

* Evaluation of the iPM measured data query responses with the following timing
  + ‘MEASURE?’ responses fully transferred 25 +/-5 ms (typical) after the query completed transfer. The iPM sends the response ASCII header and then a ~5ms gap before sending the byte payload.
* Handling Responses (Options for discussion)
  + Queries are blocking. Response timeouts (XXX ms) result in NaN data. Query response headers are parsed to verify the data payload size, and a delay counter is set to allow full transmission of the byte payload before sending the next query.
  + All queries are strictly queued in time (non-blocking), and response latency is expected by design with XXX ms additional buffer. Bus conflict may result in corrupt data that is passed on to post-processing.
* The NIDAS sensor profile shall re-package the response to include the device address and a breaking character sequence to uniquely separate queries. These are required to aid post-processing.

## Initialization

Device initialization shall be performed upon each execution of the NIDAS sensor profile application at the specified serial port and shall accomplish the following tasks. These tasks are performed only for addresses specified in the NIDAS XML.

* Verify device existence at all addresses
* Query device serial numbers at all addresses
* Query device firmware version at all addresses
* Perform built-in self-test on all devices
* Turn the device OFF, wait >100 ms, and then ON to reset the state

The above tasks are accomplished with the following commands.

|  |  |  |
| --- | --- | --- |
| **Command/Query Type** | **Command/Query String** | **Response** |
| Set Active address | “ADR <address\_int><LF>” | None |
| Turn Device OFF | “OFF<LF>” | “OK<LF>” |
| Turn Device ON (reset) | “RESET<LF>” | “OK<LF>” |
| Query Serial number | “SERNO?<LF>” | “<serial\_number><LF>” |
| Query firmware version | “VER?<LF>” | “<firmware\_version><LF>” |
| Execute built-in self test | “TEST<LF>” | “OK<LF>” |
| Query self test results | “BITRESULT?<LF>” | “24<LF>”{BitData\_24Bytes} |

The Serial Number query must be validated before continuing. If the first query with an expected response, so OFF, has no response, an error shall be logged and no further queries to that address shall be attempted. Other responses are intended only to be recorded in the raw data file (.ads).

## Fault Identification and Response

* Initialization serial number queries at addresses that have no response or bad data response shall result in a logged error. No further queries shall be made to that address. If no device initialization is successful, then the application waits 5 seconds and then re-attempts initialization on all devices.
* RECORD? Query responses shall have their CRC values validated. Responses with an incorrect value shall be stored in the raw data (i.e. .ads file) but set as NaN in processed data (i.e. .nc file).
* A “heartbeat” should be implemented. If no data is received for more than TIMEOUT, then the application controlling queries shall wait 5 seconds and then be restarted. The timeout restart shall be logged.
* A “Bad Data” counter should be implemented. If bad data is received (e.g. header error, size error, CRC error, query timeout) then the counter is incremented. If counter reaches 10 errors, then application shall wait 5 seconds and then be restarted. Bad data packets shall be stored in the .ads file and shall not be discarded. The bad data restart shall be logged.
* Query timeouts … TBD

## Timing Stamps

Query responses shall be time-stamped by NIDAS at the time they are received. Accuracy adjustments to the time stamp in processing is not required.

## Sampling Sequence, Bandwidth

* Devices shall be sampled in ascending order by address (i.e. 0, 1, 2, 3, 0, 1, 2, 3)
* All queries are completed for one address before advancing to next address
* For a given address, MEASURE? Is queried first, followed by STATUS?, followed by RECORD?
* A complete set of queries (MEASURE?, STATUS?, RECORD?) for one address is estimated to require < 40ms including all query transmissions, responses, and dead time. A complete set of queries for four (4) addresses is therefore estimated to be < 160ms. Therefore, 5-10 Hz MEASURE? query sampling may be possible.

## NIDAS XML Device Example

|  |
| --- |
| <!-- In the sensor catalog -->  <sensor ID=”iPM\_1phase” class=”UDPSocketSensor” devicename=”usock::30100 suffix=”\_iPM1”>  <sample id=”1” rate=”1” scanfFormat=”MEASURE,%f,%\*f,%f,%\*f,%\*f,%f,%\*f,%\*f,%f,%\*f,%\*f,%f,%\*f,%\*f,%f,%\*f,%\*f,%d”>  <variable longname="AC Power Frequency" name="FREQ" units="Hz"/>  <variable longname="AC Voltage, RMS Phase A" name="VRMSA" units="V"/>  <variable longname="AC Voltage, Peak Phase A" name="VPKA" units="V"/>  <variable longname="AC Voltage, DC Component Phase A" name="VDCA" units="V"/>  <variable longname="AC Phase Angle Phase A" name="PHA" units="deg\_C"/>  <variable longname="AC Voltage THD Phase A" name="THDA" units="%"/>  <variable longname="Power OK" name="POWEROK" units=""/>  </sample>  <sample id="2" rate="1" scanfFormat="STATUS,%d,%\*d,%l,%l,%\*d">  <variable longname="Operational State" name="OPSTATE" units=""/>  <variable longname="Power Trip Flags, performance exceeds limits" name="TRIPFLAGS" units=""/>  <variable longname="Power Caution Flags, marginal performance" name="CAUTIONFLAGS" units=""/>  </sample>  <sample id="3" rate="1" scanfFormat="RECORD,%d,%\*d,%\*l,%l,%\*l,%\*l,%f,%f,%\*f,%\*f,%\*f,%\*f,%f,%f,%f,%f,%\*f,%\*f,%\*f,%\*f,%f,%f,%\*f,%\*f,%\*f,%\*f,%f,%f,%\*f,%\*f,%\*f,%\*f,%\*f">  <variable longname="Event Type" name="EVTYPE" units=""/>  <variable longname="Elapsed Time since power-up" name="TIME" units="ms"/>  <variable longname="Minimum AC Voltage, RMS Phase A" name="VRMSMINA" units="V"/>  <variable longname="Maximum AC Voltage, RMS Phase A" name="VRMSMAXA" units="V"/>  <variable longname="Minimum AC Power Frequency" name="FREQMIN" units="Hz"/>  <variable longname="Maximum AC Power Frequency" name="FREQMAX" units="Hz"/>  <variable longname="Minimum AC Voltage, DC Component Phase A" name="VDCMINA" units="V"/>  <variable longname="Maximum AC Voltage, DC Component Phase A" name="VDCMAXA" units="V"/>  <variable longname="Minimum AC Voltage THD Phase A" name="THDMINA" units="%"/>  <variable longname="Maximum AC Voltage THD Phase A" name="THDMAXA" units="%"/>  <variable longname="Minimum AC Voltage, Peak Phase A" name="VPKMINA" units="V"/>  <variable longname="Maximum AC Voltage, Peak Phase A" name="VPKMAXA" units="V"/> </sample>  </sensor>  <sensor ID="iPM\_3phase" class="UDPSocketSensor" devicename="usock::30101" suffix="\_iPM3">  <sample id="1" rate="1" scanfFormat="MEASURE,%f,%\*f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%d">  <variable longname="AC Power Frequency" name="FREQ" units="Hz"/>  <variable longname="AC Voltage, RMS Phase A" name="VRMSA" units="V"/>  <variable longname="AC Voltage, RMS Phase B" name="VRMSB" units="V"/>  <variable longname="AC Voltage, RMS Phase C" name="VRMSC" units="V"/>  <variable longname="AC Voltage, Peak Phase A" name="VPKA" units="V"/>  <variable longname="AC Voltage, Peak Phase B" name="VPKB" units="V"/>  <variable longname="AC Voltage, Peak Phase C" name="VPKC" units="V"/>  <variable longname="AC Voltage, DC Component Phase A" name="VDCA" units="V"/>  <variable longname="AC Voltage, DC Component Phase B" name="VDCB" units="V"/>  <variable longname="AC Voltage, DC Component Phase C" name="VDCC" units="V"/>  <variable longname="AC Phase Angle Phase A" name="PHA" units="deg\_C"/>  <variable longname="AC Phase Angle Phase B" name="PHB" units="deg\_C"/>  <variable longname="AC Phase Angle Phase C" name="PHC" units="deg\_C"/>  <variable longname="AC Voltage THD Phase A" name="THDA" units="%"/>  <variable longname="AC Voltage THD Phase B" name="THDB" units="%"/>  <variable longname="AC Voltage THD Phase C" name="THDC" units="%"/>  <variable longname="Power OK" name="POWEROK" units=""/>  </sample>  <sample id="2" rate="1" scanfFormat="STATUS,%d,%\*d,%l,%l,%\*d">  <variable longname="Operational State" name="OPSTATE" units=""/>  <variable longname="Power Trip Flags, performance exceeds limits" name="TRIPFLAGS" units=""/>  <variable longname="Power Caution Flags, marginal performance" name="CAUTIONFLAGS" units=""/>  </sample>  <sample id="3" rate="1" scanfFormat="RECORD, %d,%\*d,%\*l,%l,%\*l,%\*l,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%\*f">  <variable longname="Event Type" name="EVTYPE" units=""/>  <variable longname="Elapsed Time since power-up" name="TIME" units="ms"/>  <variable longname="Minimum AC Voltage, RMS Phase A" name="VRMSMINA" units="V"/>  <variable longname="Maximum AC Voltage, RMS Phase A" name="VRMSMAXA" units="V"/>  <variable longname="Minimum AC Voltage, RMS Phase B" name="VRMSMINB" units="V"/>  <variable longname="Maximum AC Voltage, RMS Phase B" name="VRMSMAXB" units="V"/>  <variable longname="Minimum AC Voltage, RMS Phase C" name="VRMSMINC" units="V"/>  <variable longname="Maximum AC Voltage, RMS Phase C" name="VRMSMAXC" units="V"/>  <variable longname="Maximum AC Power Frequency" name="FREQMAX" units="Hz"/>  <variable longname="Minimum AC Power Frequency" name="FREQMIN" units="Hz"/>  <variable longname="Minimum AC Voltage, Peak Phase A" name="VPKMINA" units="V"/>  <variable longname="Maximum AC Voltage, Peak Phase A" name="VPKMAXA" units="V"/>  <variable longname="Minimum AC Voltage, Peak Phase B" name="VPKMINB" units="V"/>  <variable longname="Maximum AC Voltage, Peak Phase B" name="VPKMAXB" units="V"/>  <variable longname="Minimum AC Voltage, Peak Phase C" name="VPKMINC" units="V"/>  <variable longname="Maximum AC Voltage, Peak Phase C" name="VPKMAXC" units="V"/>  <variable longname="Minimum AC Voltage, DC Component Phase A" name="VDCMINA" units="V"/>  <variable longname="Maximum AC Voltage, DC Component Phase A" name="VDCMAXA" units="V"/>  <variable longname="Maximum AC Voltage, DC Component Phase B" name="VDCMAXB" units="V"/>  <variable longname="Minimum AC Voltage, DC Component Phase B" name="VDCMINB" units="V"/>  <variable longname="Minimum AC Voltage, DC Component Phase C" name="VDCMINC" units="V"/>  <variable longname="Maximum AC Voltage, DC Component Phase C" name="VDCMAXC" units="V"/>  <variable longname="Minimum AC Voltage THD Phase A" name="THDMINA" units="%"/>  <variable longname="Maximum AC Voltage THD Phase A" name="THDMAXA" units="%"/>  <variable longname="Maximum AC Voltage THD Phase B" name="THDMAXB" units="%"/>  <variable longname="Minimum AC Voltage THD Phase B" name="THDMINB" units="%"/>  <variable longname="Minimum AC Voltage THD Phase C" name="THDMINC" units="%"/>  <variable longname="Maximum AC Voltage THD Phase C" name="THDMAXC" units="%"/> </sample>  </sensor>  <!-- Under acserver DSM entry -->  <sensor IDREF=”iPM\_1phase” devicename=”usock::30101” id=”850” suffix=”\_iPM1”/>  <sensor IDREF=”iPM\_1phase” devicename=”usock::30102” id=”860” suffix=”\_iPM2”/>  <sensor IDREF=”iPM\_3phase” devicename=”usock::30103” id=”870” suffix=”\_iPM3”/>  <sensor IDREF=”iPM\_1phase” devicename=”usock::30104” id=”880” suffix=”\_iPM4”/>  <!-- Under the DSM to which the iPM is connected -->  <sensor ID=”NAIiPM” class=”raf.UDPiPMSensor” devicename=”sock:30222” id=”6200” timeout=”10”>  <parameter name=”device” type=”string” value=”/dev/ttyS0”/>  <parameter name=”status\_port” type=”int” value=”30222”/>  <parameter name=”measurerate” type=”int” value=”1”/>  <parameter name=”recordperiod” type=”int” value=”10”/>    <parameter name=”num\_addr” type=”int” value=”4”/>  <parameter name=”dev0” type=”string” value=”0,1,5,30101”/> <!-- addr, numphases, procqueries, port -->  <parameter name=”dev1” type=”string” value=”1,1,5,30102”/>  <parameter name=”dev2” type=”string” value=”2,3,7,30103”/>  <parameter name=”dev3” type=”string” value=”3,3,5,30104”/>  </sensor> |

* ‘measurerate’ (Hz) indicates the rate of data collection for the STATUS and MEASURE queries. This applies to all devices on the same physical bus. Typically = 1 (Hz).
* ‘recordperiod’ (minutes) indicates the period of RECORD queries. Typically = 10 (minutes).
* ‘numphases’ (integer) indicates whether 1 phase, or 3-phases of data are to be capture. Should only be =1 or =3
* ‘procqueries’ is an integer representation of 3-bit Boolean field (similar to Linux permissions values), indicating whether query responses [RECORD,MEASURE,STATUS] should be processed and variables included in a processed data file.
  + A value of d’3 (b’011) indicates that MEASURE+STATUS are processed.
  + A value of d’5 (b’101) indicates that RECORD+STATUS are processed.

## Data Variables to Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Description** | **Units** | **Corresponding**  **Query** |
| OPSTATE | Operational State | Integer, 8-bit | STATUS? |
| TRIPFLAGS | Power Trip Flags, performance exceeds limits | Integer, 32-bit | STATUS? |
| CAUTIONFLAGS | Power Caution Flags, marginal performance | Integer, 32-bit | STATUS? |
|  | | | |
| **Variable** | **Description** | **Units** | **Corresponding**  **Query** |
| FREQ | AC Power Frequency | Report in Hz  Raw data in uint, 16-bit  1-LSB = 0.1Hz | MEASURE? |
| VRMSA,  VRMSB,  VRMSC | AC Voltage, RMS  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.1V | MEASURE? |
| VPKA,  VPKB,  VPKC | AC Voltage, Peak  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.1V | MEASURE? |
| VDCA,  VDCB,  VDCC | AC Voltage, DC Component  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.001V | MEASURE? |
| PHA,  PHB,  PHC | AC Phase Angle  Phases A, B, C | Report in Degrees  Raw data in uint, 16-bit  1-LSB = 0.1 degree | MEASURE? |
| THDA,  THDB,  THDC | AC Voltage THD  Phases A, B, C | Reported in %  Raw data in uint, 8-bit  1-LSB = 0.1% | MEASURE? |
| POWEROK | Power OK, All phases | Integer, 8-bit | MEASURE? |
|  | | | |
| **Variable** | **Description** | **Units** | **Corresponding**  **Query** |
| FREQMAX | Maximum AC Power Frequency | Report in Hz  Raw data in uint, 16-bit  1-LSB = 0.1Hz | RECORD? |
| FREQMIN | Minimum AC Power Frequency | Report in Hz  Raw data in uint, 16-bit  1-LSB = 0.1Hz | RECORD? |
| VRMSMAXA  VRMSMAXB  VRMSMAXC | Maximum AC Voltage, RMS  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.1V | RECORD? |
| VRMSMINA  VRMSMINB  VRMSMINC | Minimum AC Voltage, RMS  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.1V | RECORD? |
| VPKMAXA  VPKMAXB  VPKMAXC | Maximum AC Voltage, Peak  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.1V | RECORD? |
| VPKMINA  VPKMINB  VPKMINC | Minimum AC Voltage, Peak  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.1V | RECORD? |
| VDCMAXA  VDCMAXB  VDCMAXC | Maximum AC Voltage, DC Component  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.001V | RECORD? |
| VDCMINA  VDCMINB  VDCMINC | Minimum AC Voltage, DC Component  Phases A, B, C | Report in Volts  Raw data in uint, 16-bit  1-LSB = 0.001V | RECORD? |
| THDMAXA,  THDMAXB,  THDMAXC | Maximum AC Voltage THD  Phases A, B, C | Reported in %  Raw data in uint, 8-bit  1-LSB = 0.1% | RECORD? |
| THDMINA,  THDMINB,  THDMINC | Minimum AC Voltage THD  Phases A, B, C | Reported in %  Raw data in uint, 8-bit  1-LSB = 0.1% | RECORD? |
| TIME | Elapsed Time since power-up | 32-bit uint, 1-LSB=1ms | RECORD? |
| EVTYPE | Event Type | 8-bit uint | RECORD? |

Notes

* Variables in green indicate values of interest, but are typically excluded from data processing unless Trip and Caution Flags show excess activity. Processing options are configured in the NIDAS XML file.
* Inclusion of variables for different phases depends on the number of phases being collected, as specified in XML as 1 or 3.
* Variables should be appended or prepended with the sample name to differentiate between devices
* All data are made available in the UDP handoff to acserver. The XML specifies a reduced set of variables to be processed.

## Application Utilities

The following features shall be incorporated into the software design to enable lower-level control/querying for (automated)testing and diagnostics

* A ‘ipm\_query’ Linux command line utility shall have the following capabilities:
  + Utility sends a single query (or configuration command)
    - Some commands may contain a <SPACE> within the command string and all commands/queries must be terminated with <LF>, not <CR><LF>
  + Utility must specify the Linux device, baud rate and address value on the command line as options
    - Defaults device=/dev/ttyS0, baud=115200, addr=0
  + Utility receives and parses query response and repackages the byte payload into an ASCII comma-delimited string of values with appropriate units (V, Hz, degrees, etc.) for MEASURE?, STATUS?, and RECORD? queries.
  + The response data string must be prepended with a date/time stamp, query/command string and address. This is the “Sample String.” The Sample String shall be output to stdout.
    - Commands generally have no response, so their Sample String is composed of time-stamp, commandString, and address.
    - Sending a command (that does not expect a response) must block and wait until the command has been fully transmitted plus 10ms
    - An option shall be available that expands the Sample String stdout output of MEASURE?, STATUS? and RECORD? queries into a “report” format with fully labeled variable names, values and units
  + Utility must verify the CRC of the “RECORD?” query
  + Utility must return a status value of the success of sending queries
    - (0=success, -1=error, -2=timeout)
    - Commands without responses return Success.
  + Utility must allow data logging (such as piping response data to a log file)
  + Utility can run as a stand-alone application without the need for adjacent NIDAS features like XML, DSM process, ACSERVER processes
  + Utility must timeout and return within 100ms
  + Scaling factors for variables shall be hard-coded. A command line option will enable (default) or disable the scaling factors.
  + Example Linux Command:
    - % ipm\_query -b {baudRate} -d {device} -a {address} {queryString} > {logFile}
    - % ipm\_query -b 115200 -d /dev/ttyCTI0 -a 1 MEASURE? > log.txt