# VPX

# Operator's Guide
## VPX 6U I2C

## MILITARY GRADE VITA 62 COMPLIANT POWER SUPPLY


VPX 6U I2C

Made in USA

SynQor®
*Advancing The Power Curve®*

*1.888.567.9596 (USA only)* | **SynQor Headquarters** *155 Swanson Road, Boxborough, MA 01719-1316 USA* | **www.synqor.com**

# Table of Contents

# 1 Overview

The SynQor 6U VPX power supply modules have optional $I^2C$ capability that provides the ability for the user to query the status of the power supply. The VPX power supply allows the user to monitor input voltage, current and power status; output voltage for each output rails; output current and power status for VS1, VS2, VS3 and 3.3V_AUX outputs; temperature readings at card edges.

The VPX $I^2C$ communication supports both PMBus and IPMI interfaces and complies with VITA 46.11 specification. The VPX dynamically identifies which interface is being used based on the incoming data structure without the need of any additional setup.

# 2 $I^2C$ Configurations

The system management buses SM[3:0] on the 6U P1 connector represent the physical lines used for $I^2C$ communications. The VPX uses SM[1:0] for the primary $I^2C$ bus, and SM[3:2] for the redundant $I^2C$ bus. The pin definitions are shown below.

| Pin Number | Pin Name | Description |
|---|---|---|
| C5 | SM0 | Primary $I^2C$ Clock Line |
| D5 | SM1 | Primary $I^2C$ Data Line |
| A6 | SM2 | Redundant $I^2C$ Clock Line |
| B6 | SM3 | Redundant $I^2C$ Data Line |

**Table A: VPX 6U $I^2C$ Pin Definitions**

The primary $I^2C$ and redundant $I^2C$ buses are identical in their functionalities, but operate independently. For example, the primary $I^2C$ bus can be configured to run PMBus interface, and redundant $I^2C$ bus configured for IPMI interface at the same time. Both primary and redundant $I^2C$ buses have the same $I^2C$ address described below. External pull-up resistors on the backplane are required. Pull-up voltage should be 3.3V. The $I^2C$ communication is configured for 100 kHz operation.

# 3 $I^2C$ Addressing

The $I^2C$ address is 7-bit and configured using Geographical Addressing pins defined by VITA 46.11. It is derived by using the state of the GA[4:0]* pins as the LSBs of a 7-bit numeric field, and adding 40h. GA[4:0] and GAP* bits are grounded to indicate a logic “1” or left floating to indicate a logic “0”. They have internal pull-up resistors of 10 kΩ to 3.3V.

The GAP* bit provides odd parity protection over the GA[4:0]* bits. The sum of all the grounded geographical address pins, GA[4:0]* and GAP*, must be odd. The IPMB address is defined by VITA 46.11 for communication over IPMI interface. The IPMB address is determined by multiplying its 7-bit $I^2C$ address by 2, to derive an 8-bit IPMB address. The full range of assignable address is shown in the table below. If the geographical address is invalid, the VPX module will not be assigned an $I^2C$ address and its $I^2C$ interface will not be active.

| Slot ID | Geographical Pins | | | | | | I²C Address | IPMB Address |
|---|---|---|---|---|---|---|---|---|
| | GAP* | GA4* | GA3* | GA2* | GA1* | GA0* | | |
| 1 | U | U | U | U | U | G | 41h | 82h |
| 2 | U | U | U | U | G | U | 42h | 84h |
| 3 | G | U | U | U | G | G | 43h | 86h |
| 4 | U | U | U | G | U | U | 44h | 88h |
| 5 | G | U | U | G | U | G | 45h | 8Ah |
| 6 | G | U | U | G | G | U | 46h | 8Ch |
| 7 | U | U | U | G | G | G | 47h | 8Eh |
| 8 | U | U | G | U | U | U | 48h | 90h |
| 9 | G | U | G | U | U | G | 49h | 92h |
| 10 | G | U | G | U | G | U | 4Ah | 94h |
| 11 | U | U | G | U | G | G | 4Bh | 96h |
| 12 | G | U | G | G | U | U | 4Ch | 98h |
| 13 | U | U | G | G | U | G | 4Dh | 9Ah |
| 14 | U | U | G | G | G | U | 4Eh | 9Ch |
| 15 | G | U | G | G | G | G | 4Fh | 9Eh |
| 16 | U | G | U | U | U | U | 50h | A0h |
| 17 | G | G | U | U | U | G | 51h | A2h |
| 18 | G | G | U | U | G | U | 52h | A4h |
| 19 | U | G | U | U | G | G | 53h | A6h |
| 20 | G | G | U | G | U | U | 54h | A8h |
| 21 | U | G | U | G | U | G | 55h | AAh |
| 22 | U | G | U | G | G | U | 56h | ACh |
| 23 | G | G | U | G | G | G | 57h | AEh |
| 24 | G | G | G | U | U | U | 58h | B0h |
| 25 | U | G | G | U | U | G | 59h | B2h |
| 26 | U | G | G | U | G | U | 5Ah | B4h |
| 27 | G | G | G | U | G | G | 5Bh | B6h |
| 28 | U | G | G | G | U | U | 5Ch | B8h |
| 29 | G | G | G | G | U | G | 5Dh | BAh |
| 30 | G | G | G | G | G | U | 5Eh | BCh |
| 31 | U | G | G | G | G | G | 5Fh | BEh |

Notes:
U = Unconnected
G = Biased to Ground on the Backplane

**Table B: Valid I²C Address Assignment Table**

# 4 PMBus Interface

## 4.1 PMBus Data Format

Measurement data transmitted on the PMBus uses DIRECT data format defined by the PMBus specification. DIRECT format data is a two byte, two's complement binary integer. Low byte of data is transmitted first.

The user system should use the following equation to convert the value received from the VPX module into a reading of volts, amperes, watts, degrees Celcius:

$$X = \frac{1}{m}(Y * 10^{-R} - b)$$

Where:

X, is the calculated, "real world" value in the appropriate units (A, V, W, °C);

m, the slope coefficient, is a two byte, two's complement integer;

Y, is a two byte two's complement integer received from the VPX module;

b, the offset, is a two byte, two's complement integer;

R, the exponent, is a one byte, two's complement integer.

## 4.2 PMBus Coefficients

The table below provides the coefficients used for conversion and some examples.

| Parameter | | Conversion Coefficients | | | Examples | | | Note |
|---|---|---|---|---|---|---|---|---|
| | | m | b | R | X | Y | | |
| | | | | | | Decimal | Hex | |
| Input Voltage | 28V Model | 100 | 0 | 0 | 28 V | 2800 | 0AF0h | 10 mV/bit |
| | 270V Model | 10 | 0 | 0 | 270 V | 2700 | 0A8Ch | 0.1 V/bit |
| | AC Model | 1 | 0 | 0 | 115 Vrms | 115 | 0073h | 1 V/bit |
| Output Voltage | | 100 | 0 | 0 | 12 V | 1200 | 04B0h | 10 mV/bit |
| | | | | | -12 V | -1200 | FB50h | |
| Input Current | 28V Model | 100 | 0 | 0 | 40 A | 4000 | 0FA0h | 10 mA/bit |
| | 270V Model | 1000 | 0 | 0 | 3.0 A | 3000 | 0BB8h | 1 mA/bit |
| | AC Model | 1000 | 0 | 0 | 6.0 Arms | 6000 | 1770h | 1 mArms/bit |
| Output Current | | 100 | 0 | 0 | 30 A | 3000 | 0BB8h | 10 mA/bit |
| Power | | 1 | 0 | 0 | 500 W | 500 | 01F4h | 1 W/bit |
| Temperature | | 10 | 0 | 0 | 85 °C | 850 | 0352h | 0.1 °C/bit |
| | | | | | -40 °C | -400 | FE70h | |

**Table C: PMBus Coefficients Table**

The above **m**, **b** and **R** coefficients can also be retrieved using the *COEFFICIENTS* command. This command uses Block Write-Block Read Process Call as described in the SMBus specification.

## 4.3 PMBus Packet Error Checking

The PMBus interface implemented on SynQor VPX modules supports the standard packet error checking scheme defined by the PMBus specification. The response data generated by the VPX module will contain a Packet Error Code (PEC) if requested from the host. For any Write command from the host, the VPX module will check PEC to validate the message first. A command message from the host will be disregarded if it contains a PEC that does not match the data received. A Write command will be processed if the message from the host does not contain a PEC.

The PEC uses an 8-bit cyclic redundancy check (CRC-8) represented by the polynomial, $C(x) = x^8 + x^2 + x^1 +1$ and must be calculated in the order of bits as received. The PEC calculation includes all bytes in the transmission, including address, command and data. The PEC calculation does not include ACK, NACK, START, STOP nor Repeated START bits. This means that the PEC is computed over the entire message from the first START condition.

## 4.4 PMBus Page Access

The write PAGE command allows the user to select a specific output to query. When a read command is transmitted to the VPX module such as READ_VOUT and READ_IOUT, the VPX module will return the measurement information based on the selected page. The module will return FFFFh if a valid page is not selected or if the command is not supported by the current page. The user could also use read PAGE command to retrieve the current page information.

| PAGE | Description |
|---|---|
| 00h | Not used, default value on power on |
| 01h | Main +12V voltage, current, power (Note 1) |
| 02h | Main +5V voltage, current, power |
| 03h | +3.3V auxiliary voltage, current, power |
| 04h | +12V auxiliary voltage |
| 05h | -12V auxiliary voltage |
| 06h – FFh | Not used |

Note 1: +12V current and power readings represent the combined current and power of +12V, +12Vaux and -12V aux outputs.

**Table D: PMBus Page Description**

## 4.5 Supported PMBus Command List

The following table provides a list of supported PMBus commands by the SynQor VPX 6U module.

| Command Code | Command Name | Type | Number of Data Bytes | Description |
|---|---|---|---|---|
| 00h | PAGE | R/W Byte | 1 | Allows user to set or read the page for any commands that require page selection. |
| 30h | COEFFICIENTS | Block Write-Block Read | See PMBus | Retrieve coefficients for data conversion. See section 4.7.5 for example. |
| 88h | READ_VIN | Read Word | 2 | Read input voltage. |
| 89h | READ_IIN | Read Word | 2 | Read input current. |
| 8Bh | READ_VOUT | Read Word | 2 | Read output voltage for the selected page. Page number between 01h – 05h must be selected. |
| 8Ch | READ_IOUT | Read Word | 2 | Read output current for the selected page. Page number between 01h – 03h must be selected. |
| 8Dh | READ_TEMPERATURE_1 | Read Word | 2 | Output side card edge temperature. |
| 8Eh | READ_TEMPERATURE_2 | Read Word | 2 | Input side card edge temperature. |
| 96h | READ_POUT | Read Word | 2 | Read output power for the selected page. Page number between 01h – 03h must be selected. |

| | | | | |
|---|---|---|---|---|
| 97h | READ_PIN | Read Word | 2 | Read input power. |
| D0 | READ_TIMER | Block Read | 6 | OEM command – read module operating time. |
| D1 | READ_FIRMWARE | Read Word | 2 | Read firmware information. |
| D2 | READ_SERIAL_NUMBER | Block Read | Module Dependent | Read VPX serial number. |
| D3 | READ_PART_NUMBER | Block Read | Module Dependent | Read VPX part number. |
| FEh | COMMAND_EXT | Extended Command | 0 or 1 | OEM extended write byte command. |

**Table E: PMBus Supported Commands**

## 4.6 SynQor Specific PMBus Commands

### 4.6.1 READ_TIMER

READ_TIMER is a SynQor defined Block Read PMBus command. It allows the user to retrieve a timer value that shows how long the module has been operating. This is a volatile time that increment when input voltage is applied, regardless of whether outputs are enabled or not. It will reset upon removing of the input voltage.

The command code for READ_TIMER is D0h. Upon receiving this command, the VPX module will transmit six data bytes followed by PEC. The data description and an example of 1 Day, 6 Hours, 32 Minutes, 22 Seconds in operation are shown below.

| Data Byte | Example | Description |
|---|---|---|
| 1 | 05h | The first data byte will always be 05h, indicating there are five data bytes following |
| 2 | 16h | **Seconds**, a value between 0 and 59 will be shown here |
| 3 | 20h | **Minutes**, a value between 0 and 59 will be shown here |
| 4 | 06h | **Hours**, a value between 0 and 23 will be shown here |
| 5 | 01h | Low data byte of **Days** |
| 6 | 00h | High data byte of **Days** |

**Table F: READ_TIMER Command**

### 4.6.2 READ_FIRMWARE

READ_FIRMWARE command allows the user to retrieve the VPX module firmware information. The command code for READ_FIRMWARE is D1h. Upon receiving this command, the VPX module will transmit two data bytes followed by PEC. The first data byte indicates the product family; the second data byte indicates the firmware revision on the module.

| Data Byte | Example | Description |
|---|---|---|
| 1 | 02h | The first data byte indicates the product family of the 6U VPX power supply:<br>01h: VPX-6U-DC28P-001<br>02h: VPX-6U-DC270P-001<br>03h: VPX-6U-ACUNV-1-C-001<br>04h: VPX-6U-DC28T-001 |
| 2 | 01h | Firmware revision on the VPX module, 01h indicates revision 01. |

**Table G: READ_FIRMWARE Command**

### 4.6.3 READ_SERIAL_NUMBER

READ_SERIAL_NUMBER command (D2h) allows the user to retrieve the VPX module serial number. The command uses the Block Read syntax. The first byte of the response from the VPX indicates how many data bytes will be transmitted. The VPX module will then transmit a string of data in hexadecimal, which can be then converted to ASCII to reconstruct its serial number. For example, if the VPX serial number is "S12345678", the corresponding data bytes will be "53h 31h 32h 33h 34h 35h 36h 37h 38h". They will then be followed by PEC to end the transmission.

| Data Byte | Example | Description |
|---|---|---|
| 1 | 09h | 09h indicates there are nine data bytes following |
| 2 - 10 | 53h 31h 32h 33h 34h 35h 36h 37h 38h | Hexadecimal for S12345678 |

**Table H: READ_SERIAL_NUMBER Command**

### 4.6.4 READ_PART_NUMBER

READ_PART_NUMBER command (D3h) allows the user to retrieve the VPX module part number. An example part number can be "VPX-6U-DC270P-001-SN2". Similar to the READ_SERIAL_NUMBER command, the READ_PART_NUMBER command also uses Block Read syntax. The data transmitted is the hexadecimal conversion of the part number in ASCII code, including the "-"symbol. For example, the above part number in hexadecimal will be "56h 50h 58h 2Dh 36h 55h 2Dh 44h 43h 32h 37h 30h 50h 2Dh 30h 30h 31h 2Dh 53h 4Eh 32h".

| Data Byte | Example | Description |
|---|---|---|
| 1 | 15h | 21 in decimal, it indicates there are twenty-one data bytes following |
| 2 - 22 | 56h 50h 58h 2Dh 36h 55h 2Dh 44h 43h 32h 37h 30h 50h 2Dh 30h 30h 31h 2Dh 53h 4Eh 32h | Hexadecimal for VPX-6U-DC270P-001-SN2 |

**Table I: READ_PART_NUMBER Command**

### 4.6.5 COMMAND_EXT

The COMMAND_EXT (FEh) defines SynQor OEM write byte commands. The command code following the COMMAND_EXT (FEh) is the command to be executed by the VPX module, such as 01h. The supported OEM commands are listed below.

| Command | Action | Description |
| --- | --- | --- |
| 01h | Latch Off | The user can issue a latch-off command to the VPX module. Upon receiving this command, the VPX module will permanently disable all the output voltages until either input voltage is removed and re-applied; or a latch-off clear command is received by the VPX module through $I^2C$ interface. |

Table J: SynQor Extended Command

#### 4.6.5.1 LATCH_OFF

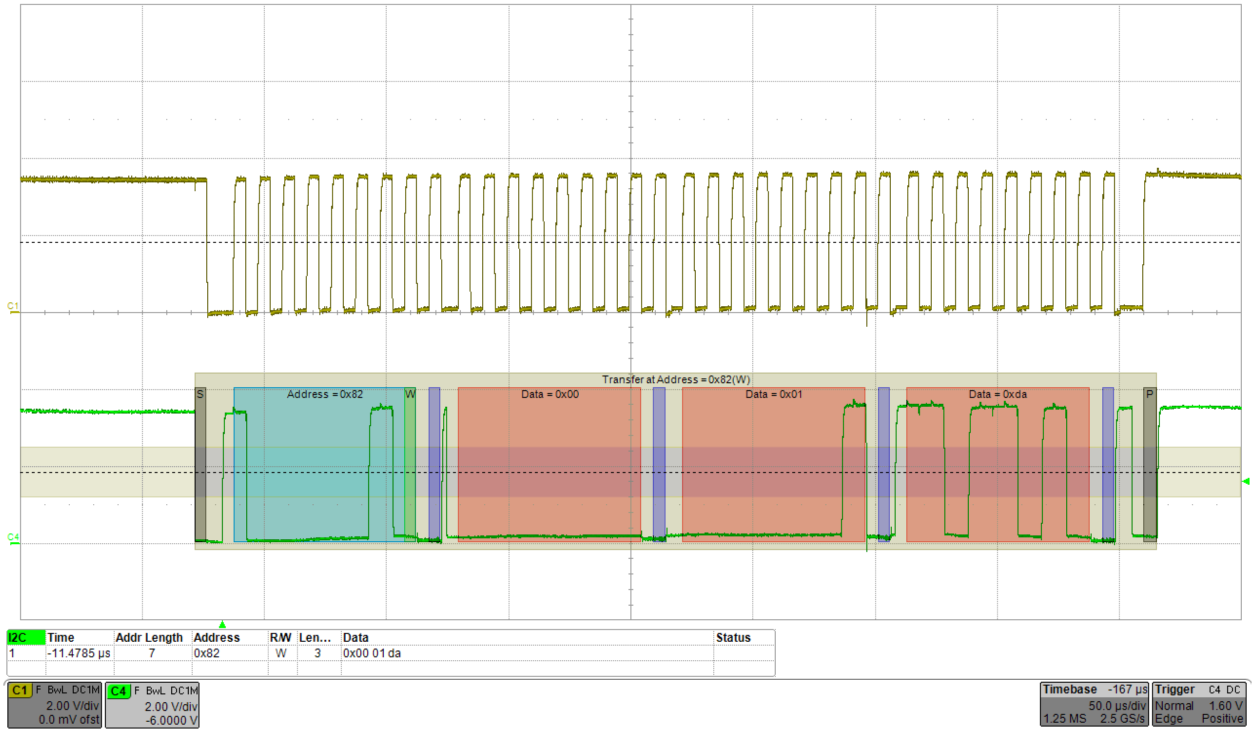
The data byte following the LATCH_OFF command is optional. If no data byte is sent, the VPX module will latch off its outputs. However, if a data byte is sent by the user, it must be either 01h: set the VPX module to the latch-off state; or 00h – clear the latch-off state, VPX will resume normal operation. Data received other than 01h or 00h will be ignored. PEC is also optional.

If PEC is sent by the user, the VPX module will verify its validity before processing the command. If no PEC is received, the VPX will process the command based on the data it received. Examples of LATCH_OFF commands are shown in the next section.

## 4.7 PMBus Examples

In the following PMBus examples, the 7-bit I$^2$C address is configured to 41h.

### 4.7.1 WRITE_PAGE



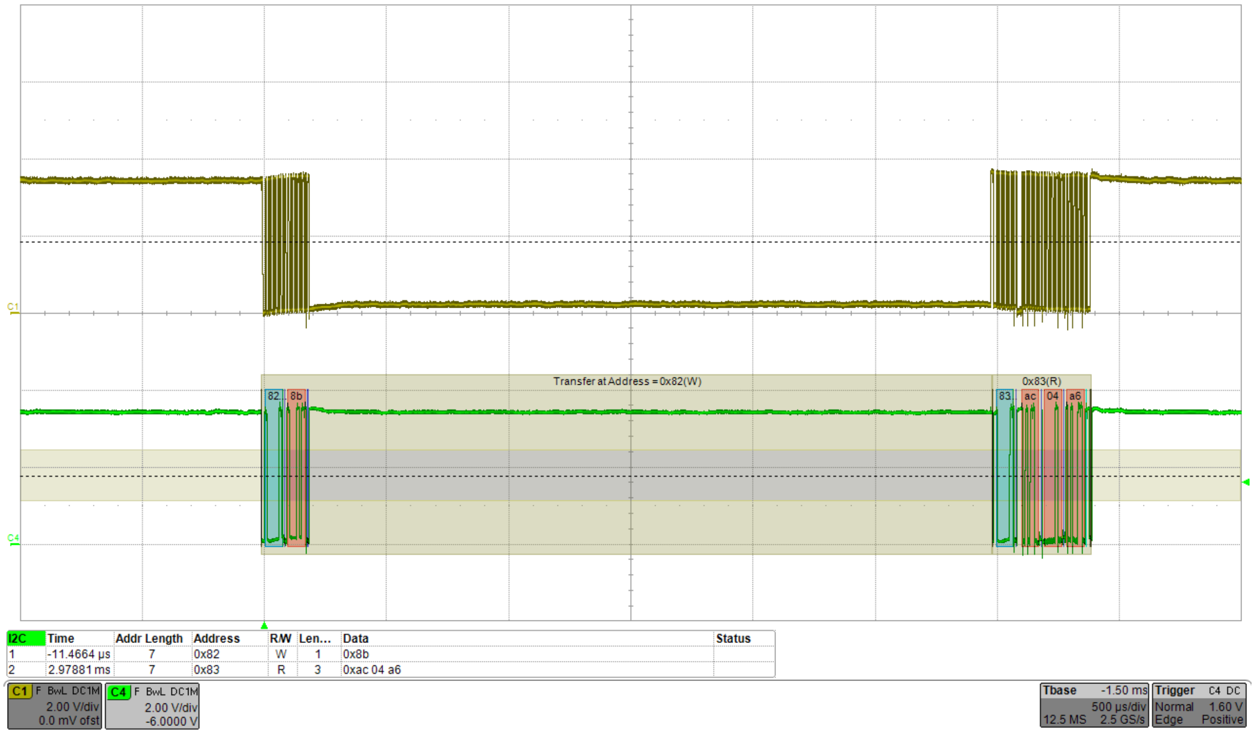

YELLOW – I²C CLOCK GREEN – I²C DATA

FIGURE 4.1: PMBUS WRITE_PAGE COMMAND – WRITE BYTE

| Byte | Value | Description |
|---|---|---|
| 1 | 82h | I$^2$C address followed by Write bit (0) |
| 2 | 00h | WRITE_PAGE command code 00h |
| 3 | 01h | Page 01h: +12V output |
| 4 | dah | Packet Error Code |

The above example configures the VPX page to 01h: +12V output. If a READ_VOUT command is received afterwards, the VPX will respond the voltage reading on the +12V output.

### 4.7.2 READ_VOUT



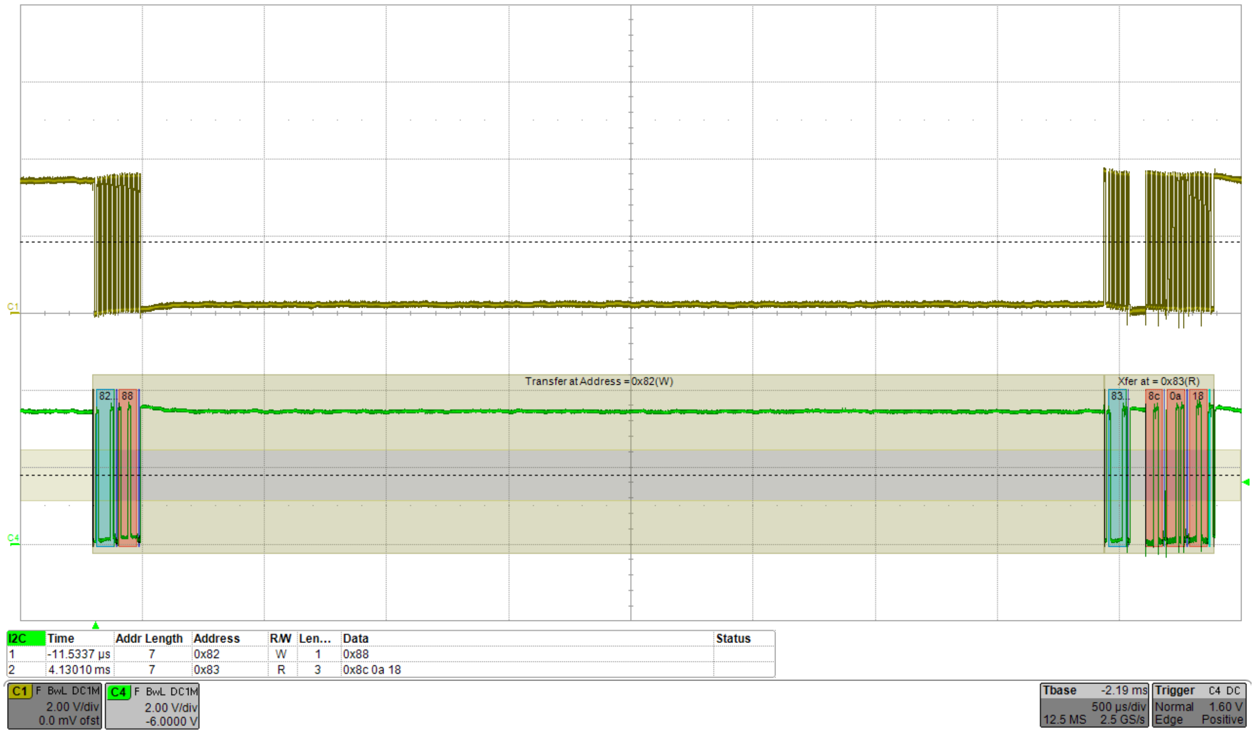

YELLOW – I²C CLOCK GREEN – I²C DATA

FIGURE 4.2: PMBUS READ_VOUT COMMAND – READ WORD

| Byte | Value | Description |
|---|---|---|
| 1 | 82h | I²C address followed by Write bit (0) |
| 2 | 8bh | READ_VOUT command code 8bh |
| 3 | 83h | I²C address followed by Read bit (1) |
| 4 | ach | Low-byte of the VPX response data |
| 5 | 04h | High-byte of the VPX response data |
| 6 | a6h | Packet Error Code |

Page 01 (+12V output) is configured on the VPX module in the above example. The response data to the READ_VOUT command is 04ach, i.e. 1196 in decimal. The converted real world value is 11.96V.

### 4.7.3 READ_VIN



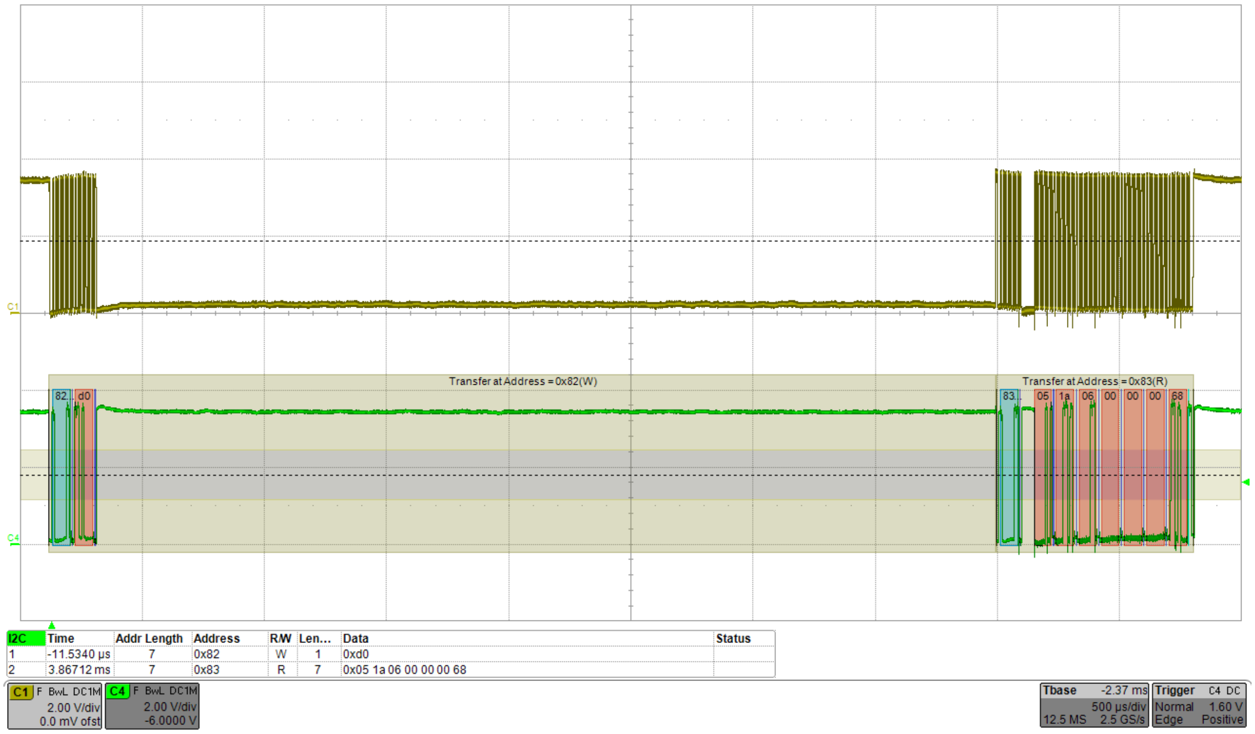

YELLOW – I²C CLOCK GREEN – I²C DATA

FIGURE 4.3: PMBUS READ_VIN COMMAND – READ WORD

| Byte | Value | Description |
|---|---|---|
| 1 | 82h | I²C address followed by Write bit (0) |
| 2 | 88h | READ_VIN command code 88h |
| 3 | 83h | I²C address followed by Read bit (1) |
| 4 | 8ch | Low-byte of the VPX response data |
| 5 | 0ah | High-byte of the VPX response data |
| 6 | 18h | Packet Error Code |

The response data to the READ_VOUT command in the above example is 0a8ch, i.e. 2700 in decimal. The converted real world value is 270.0V.

### 4.7.4 READ_TIMER




**YELLOW – I²C CLOCK GREEN – I²C DATA**

**FIGURE 4.4: PMBUS READ_TIMER COMMAND – BLOCK READ**

| Byte | Value | Description |
|---|---|---|
| 1 | 82h | I²C address followed by Write bit (0) |
| 2 | d0h | READ_TIMER command code d0h |
| 3 | 83h | I²C address followed by Read bit (1) |
| 4 | 05h | First byte of the response data, 5 bytes of data following |
| 5 | 1ah | **Seconds** |
| 6 | 06h | **Minutes** |
| 7 | 00h | **Hours** |
| 8 | 00h | Low data byte of **Days** |
| 9 | 00h | High data byte of **Days** |
| 10 | 68h | Packet Error Code |

The response to the READ_TIMER command in the above example is 6 minute and 26 seconds.

### 4.7.5 COEFFICIENTS

The VPX module supports COEFFICIENTS command to allow user to retrieve information listed on section 4.2. The following is an example to retrieve the input current reading coefficient.

| Byte | Value | Description |
|---|---|---|
| 1 | 82h | $I^2C$ address followed by Write bit (0) |
| 2 | 30h | COEFFICIENTS command code 30h |
| 3 | 02h | Byte count is 2 |
| 4 | 89h | Command code for READ_IIN |
| 5 | 01h | Indicates the coefficient is used to decode the value read from VPX |
| 6 | 05h | First byte of the response data, 5 bytes of data following |
| 7 | e8h | m: Low Byte |
| 8 | 03h | m: High Byte, m = 03e8h = 1000 in decimal |
| 9 | 00h | b: Low Byte |
| 10 | 00h | b: High Byte, b = 0 |
| 11 | 00h | R: One Byte, R = 0 |
| 12 | d8h | Packet Error Code |

### 4.7.6 LATCH_OFF

Refer to section 4.6.5.1, LATCH_OFF command can be sent with data byte and PEC being optional. The following are examples to set and clear latch-off on the VPX module.

Set latch-off:

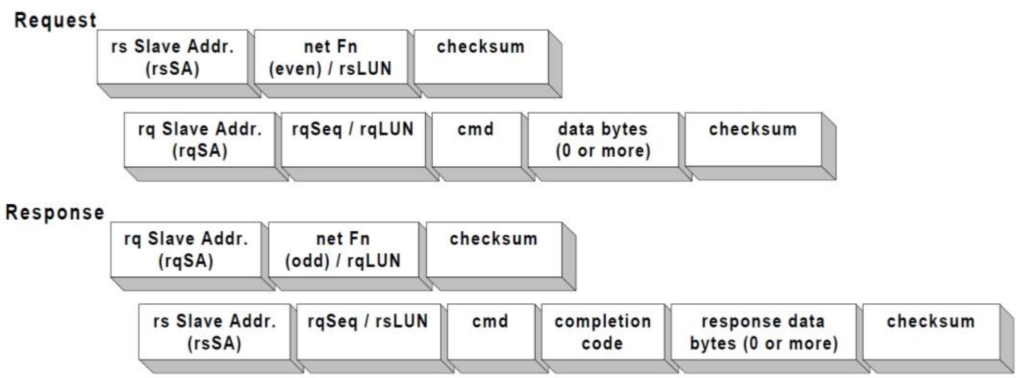
| Byte | Value | Description |
|---|---|---|
| 1 | 82h | $I^2C$ address followed by Write bit (0) |
| 2 | feh | COMMAND_EXT code |
| 3 | 01h | Command code for LATCH_OFF |
| 4 | 18h | Packet Error Code (no data byte in this example) |

Clear latch-off:

| Byte | Value | Description |
|---|---|---|
| 1 | 82h | $I^2C$ address followed by Write bit (0) |
| 2 | feh | COMMAND_EXT code |
| 3 | 01h | Command code for LATCH_OFF |
| 4 | 00h | Data byte for clearing the latch off (no PEC in this example) |

# 5 IPMI Interface

The SynQor VPX module supports IPMI interface and VITA 46.11. It allows the user to retrieve power supply status information using standard IPMB Communications Protocol shown below.




The responder Slave Address (rsSA) is the 8-bit IPMB address described in Section 2. The responder LUN (rsLUN) is 00h. The VPX module supports Sensor/Event messages (netFn 04, 05).

Checksum is the 2's complement of preceding bytes in the connection header or between the previous checksum. The 8-bit checksum can be computed using the following algorithm: Initialize checksum to 0. For each byte, checksum = (checksum + byte) module 256, then checksum = - checksum. When the checksum and the bytes are added together, modulo 256, the result should be 0.

## 5.1 IPMI Sensor Overview

The VPX module allows users to use *Get Sensor Reading* command to retrieve power supply status including voltage, current, power and temperature measurements. The *Get Sensor Reading* command for a threshold-based sensor contains the present analog reading from the sensor.

The following table lists all IPMI sensors available on the VPX 6U modules.

| Sensor Number | Sensor Name | Sensor Type Code | Event/Reading Type Code | Description | Unit |
|---|---|---|---|---|---|
| 2 | FRU Health | F2h - VITA 46.11 OEM | 04h | Predictive Failure | Discrete |
| 3 | FRU Voltage | 02h - Voltage | 05h | Limit Not Exceeded /Limit Exceeded | Discrete |
| 4 | FRU Temperature | F3h - VITA 46.11 OEM | 6Fh | Sensor Specific | Discrete |
| 7 | Input Voltage | 02h - Voltage | 01h | Analog | Volts |
| 8 | +12V Voltage | 02h - Voltage | 01h | Analog | Volts |
| 9 | +5V Voltage | 02h - Voltage | 01h | Analog | Volts |

| Sensor Number | Sensor Name | Sensor Type | Event/Reading Type Code | Description | Unit |
|---|---|---|---|---|---|
| 10 | +3.3Vaux Voltage | 02h - Voltage | 01h | Analog | Volts |
| 11 | +12Vaux Voltage | 02h - Voltage | 01h | Analog | Volts |
| 12 | -12Vaux Voltage | 02h - Voltage | 01h | Analog | Volts |
| 13 | Input Current | 03h - Current | 01h | Analog | Amperes |
| 14 | +12V Current (Note 2) | 03h – Current | 01h | Analog | Amperes |
| 15 | +3.3Vaux Current | 03h - Current | 01h | Analog | Amperes |
| 16 | +5V Current | 03h - Current | 01h | Analog | Amperes |
| 17 | Output Side Card Edge Temperature | 01h - Temperature | 01h | Analog | Kelvin |
| 18 | Input Side Card Edge Temperature | 01h - Temperature | 01h | Analog | Kelvin |
| 19 | Input Power | 0Bh - Other Units-based Sensor | 01h | Analog | Watts |
| 20 | +12V Power (Note 3) | 0Bh - Other Units-based Sensor | 01h | Analog | Watts |
| 21 | +3.3Vaux Power | 0Bh - Other Units-based Sensor | 01h | Analog | Watts |
| 22 | +5V Power | 0Bh - Other Units-based Sensor | 01h | Analog | Watts |

**Table K: IPMI Supported Sensors**

Note 2: +12V current reading represents the combined current of +12V, +12Vaux and -12V aux outputs.
Note 3: +12V power reading represents the combined power of +12V, +12Vaux and -12V aux outputs.

## 5.2 IPMI Sensor Reading Conversion

The following formula defined in IPMI specification should be used to convert the one byte “raw” sensor readings to real values in the desired units (e.g. Volts, Amps, Watts, degrees Celcius).

$$y = (Mx + B * 10^{K_1}) * 10^{K_2}$$

x, is one byte unsigned integer, received from the VPX module;

y, is the calculated, “real world” value in the appropriate units (A, V, W, °C);

M, is signed integer constant multiplier;

B, is signed additive offset;

$K_1$, is signed “B” exponent;

$K_2$, is signed *Result* “R” exponent;

The table below provides the coefficients used for IPMI analog sensor conversion.

| Sensor | | Conversion Coefficients | | | | Valid y Range (calculated “real world” value) | Typical Examples | |
|---|---|---|---|---|---|---|---|---|
| | | M | B | B exp. ($K_1$) | R exp. ($K_2$) | | x | y |
| 7: Input Voltage | 28V Model | 20 | 90 | 1 | -2 | 9 V – 60 V | 95 | 28 V |
| | 270V and AC Model | 20 | 0 | 0 | -1 | 0 – 510 V | 135 | 270 V |
| 8: +12V Voltage | | 20 | 90 | 2 | -3 | 9 V – 14.1 V | 150 | 12 V |
| 9: +5V Voltage | | 10 | 35 | 2 | -3 | 3.5 V – 6.05 V | 150 | 5 V |
| 10: +3.3Vaux Voltage | | 10 | 20 | 2 | -3 | 2 V – 4.55 V | 130 | 3.3 V |
| 11: +12Vaux Voltage | | 20 | 90 | 2 | -3 | 9 V – 14.1 V | 150 | 12 V |
| 12: -12Vaux Voltage | | -20 | -90 | 2 | -3 | -9 V – -14.1 V | 150 | -12 V |
| 13: Input Current | 28V Model | 30 | 0 | 0 | -2 | 0 – 76.5 A | 150 | 45 A |
| | 270V and AC Model | 40 | 0 | 0 | -3 | 0 – 10.2 A | 100 | 4 A |
| 14: +12V Current | 28V Model | 40 | 0 | 0 | -2 | 0 – 102 A | 150 | 60 A |
| | 270V and AC Model | 30 | 0 | 0 | -2 | 0 – 76.5 A | 150 | 45 A |
| 15: +3.3Vaux Current | | 30 | 0 | 0 | -2 | 0 – 76.5 A | 100 | 30 A |
| 16: +5V Current | | 30 | 0 | 0 | -2 | 0 – 76.5 A | 200 | 60 A |
| 17: Output Side Card Edge Temperature | | 1 | 20 | 1 | 0 | 200 K – 455 K | 150 | 350 K |
| 18: Input Side Card Edge Temperature | | 1 | 20 | 1 | 0 | 200 K – 455 K | 150 | 350 K |
| 19: Input Power | 28V Model | 50 | 0 | 0 | -1 | 0 – 1275 W | 200 | 1000 W |
| | 270V and AC Model | 40 | 0 | 0 | -1 | 0 – 1020 W | 200 | 800 W |
| 20: +12V Power | 28V Model | 50 | 0 | 0 | -1 | 0 – 1275 W | 160 | 800 W |
| | 270V and AC Model | 30 | 0 | 0 | -1 | 0 – 765 W | 160 | 480 W |
| 21: +3.3Vaux Power | | 10 | 0 | 0 | -1 | 0 – 255 W | 100 | 100 W |
| 22: +5V Power | | 20 | 0 | 0 | -1 | 0 – 510 W | 150 | 300 W |

**Table L: IPMI Coefficients Table**

For the Analog sensors, FFh is reserved to indicate the reading is above the valid range shown in the table; 00h indicates the reading is below the valid range if the minimum allowed value is not 0.

## 5.3 IPMI Commands

### 5.3.1 Sensor Device Commands

The VPX supports IPMI *Get Sensor Reading* command as defined in VITA 46.11 and IPMI specifications.

| **Command Name** | **NetFn** | **CMD** |
|---|---|---|
| Get Sensor Reading | Sensor/Event | 2Dh |

**Get Sensor Reading** description:

| | | |
|---|---|---|
| Request Data | 1 | sensor number (FFh = reserved) |
| Respond Data | 1 | Completion Code. |
| | 2 | Sensor reading<br>Byte 1: byte of reading.<br>Write as 00h if sensor does not return a numeric (analog) reading, ignore on read. |
| | 3 | Provides information on the sensor. |
| | 4 | For "Analog" sensor: Indicates where the reading stands against the threshold values. Applicable to sensor number 7 to 24.<br><br>For VITA 46.11 "Discrete" sensor: Indicates the state of the sensor. Applicable to sensor number 2, 3, 4. |

### 5.3.2 Event Commands

The VPX supports Event Messages as defined in VITA 46.11 for FRU Health Sensor, FRU Voltage Sensor and FRU Temperature Sensor.

| **Command Name** | **NetFn** | **CMD** |
|---|---|---|
| Set Event Receiver | Sensor/Event | 00h |
| Get Event Receiver | Sensor/Event | 01h |
| Platform Event (a.k.a. "Event Message") | Sensor/Event | 02h |

In order for the VPX to transmit an event message, *Set Event Receiver* command must be received by the VPX to acquire the IPMB ($I^2C$) Slave Address and LUN.

**FRU Health Sensor Event Message** description:

Request Data

| | | |
|---|---|---|
| Request Data | 1 | Event Message Rev=04h (IPMI 1.5) |
| | 2 | Sensor Type = F2h (VITA-defined FRU Health) |
| | 3 | Sensor Number |
| | 4 | [7] – Event Direction: 0b = Assertion, 1b = Deassertion<br>[6:0] – Event Type: 04h (Predictive Failure) |
| | 5 | Event Data 1<br>[7:4] – 0000b<br>[3:0] – 0h = (change in) Predictive Failure Deasserted<br>1h = (change in) Predictive Failure Asserted |

**FRU Voltage Sensor Event Message** description:

| | | |
|---|---|---|
| Request Data | 1 | Event Message Rev=04h (IPMI 1.5) |
| | 2 | Sensor Type = 02h (Voltage) |
| | 3 | Sensor Number |
| | 4 | [7] – Event Direction: 0b = Assertion, 1b = Deassertion<br>[6:0] – Event Type: 05h (digital discrete Limit Not Exceeded/Limit Exceeded) |
| | 5 | Event Data 1<br>[7:4] – 0000b<br>[3:0] – 0h = (change in) “Limit Not Exceeded” status bit<br>1h = (change in) “Limit Exceeded” status bit |

**FRU Temperature Sensor Event Message** description:

| | | |
|---|---|---|
| Request Data | 1 | Event Message Rev=04h (IPMI 1.5) |
| | 2 | Sensor Type = F3h (VITA-defined OEM for FRU Temperature) |
| | 3 | Sensor Number |
| | 4 | [7] – Event Direction: 0b = Assertion, 1b = Deassertion<br>[6:0] – Event Type: 6Fh (Sensor-specific discrete) |
| | 5 | Event Data 1<br>[7:4] – 0000b<br>[3:0] – 0h = Change in bit 0 (temp at or below lower non-critical) state<br>1h = Change in bit 1 (temp at or below lower critical) state<br>2h = Change in bit 2 (temp at or below lower non-recoverable) state<br>3h = Change in bit 3 (temp at or above upper non-critical) state<br>4h = Change in bit 4 (temp at or above upper critical) state<br>5h = Change in bit 5 (temp at or above upper non-recoverable) state |

## 5.4 IPMI Message Handling

### 5.4.1 Message Queuing

The VPX can queue up to 16 incoming messages from the requester. If the requester sends more than 16 messages in a row while the VPX response is still pending, subsequent messages after message 16 will be ignored.

### 5.4.2 Corrupted Request

If a request is received with a bad checksum, the VPX will ignore the request entirely and let the requester retry the message.

### 5.4.3 Unexpected Request

If an IPMI request message is not supported by the VPX module, the request will be ignored and no response will be generated.

### 5.4.4 Message Time-out

Once a valid request is received and a response is generated by the VPX module, it is expected that the requester will ACK the response message. However, if the response is not transmitted successfully for any reason, e.g. due to bus corruption, the VPX will retry the response three times. If the message is still not transmitted after three retries, the response will be disregarded.

## 5.5 IPMI Examples

In the following IPMI examples, the 7-bit I$^2$C address is configured to 41h. The Responder's Slave Address (rsSA) is 82h and Responder's LUN (rsLUN) is 0.

The Requester's Slave Address (rqSA) is set to 80h and Requester's LUN (rqLUN) set to 0. These two fields are arbitrarily chosen.

### 5.5.1 Get Sensor Reading – Input Voltage


YELLOW – I$^2$C CLOCK GREEN – I$^2$C DATA

FIGURE 5.1: IPMI GET SENSOR READING – INPUT VOLTAGE

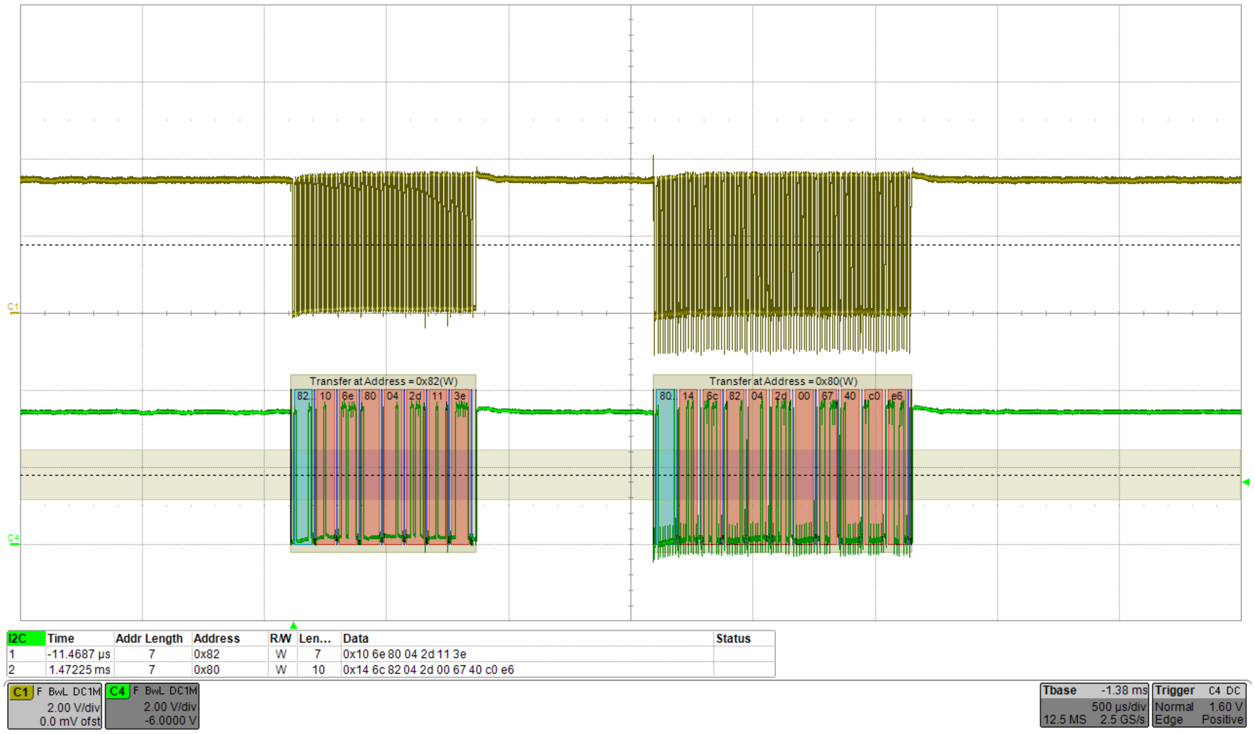
The first block of data is the Request message sent to the VPX module:

| Byte | Value | Function | Description |
|---|---|---|---|
| 1 | 82h | rs Slave Addr. (rsSA) | VPX IPMB address, LS is always 0 |
| 2 | 10h | Net Fn (even) / rsLUN | Net Fn is 04 (Sensor / Event), rsLUN is 0 |
| 3 | 6eh | checksum | Checksum for the connection header |
| 4 | 80h | rq Slave Addr. (rqSA) | Requester's slave address, LS always 0 |
| 5 | 04h | rqSeq / rqLUN | Sequence is 1 and requester's LUN is 0 |
| 6 | 2dh | command | Command 2dh – Get Sensor Reading |
| 7 | 07h | sensor number | Sensor 07 – Input Voltage |
| 8 | 48h | checksum | Checksum for preceding bytes between the previous checksum |

The second block of data is the Response message transmitted from the VPX module:

| Byte | Data | Function | Description |
|---|---|---|---|
| 1 | 80h | rq Slave Addr. (rqSA) | Requester's slave address |
| 2 | 14h | Net Fn (odd) / rqLUN | Net Fn is 05 (Sensor / Event), rqLUN is 0 |
| 3 | 6ch | checksum | Checksum for the connection header |
| 4 | 82h | rs Slave Addr. (rsSA) | Responder's slave address |
| 5 | 04h | rqSeq / rsLUN | Sequence is 1 and responder's LUN is 0 |
| 6 | 2dh | command | Command 2dh – Get Sensor Reading |
| 7 | 00h | completion code | Completion code |
| 8 | 87h | sensor reading | Sensor reading is 135 in decimal, converted real world value is 270V |
| 9 | 40h | sensor information | Event message is disabled for this sensor |
| 10 | c0h | threshold comparison | Sensor reading is within normal range |
| 11 | c6h | checksum | Checksum for preceding bytes between the previous checksum |

### 5.5.2 Get Sensor Reading – Temperature

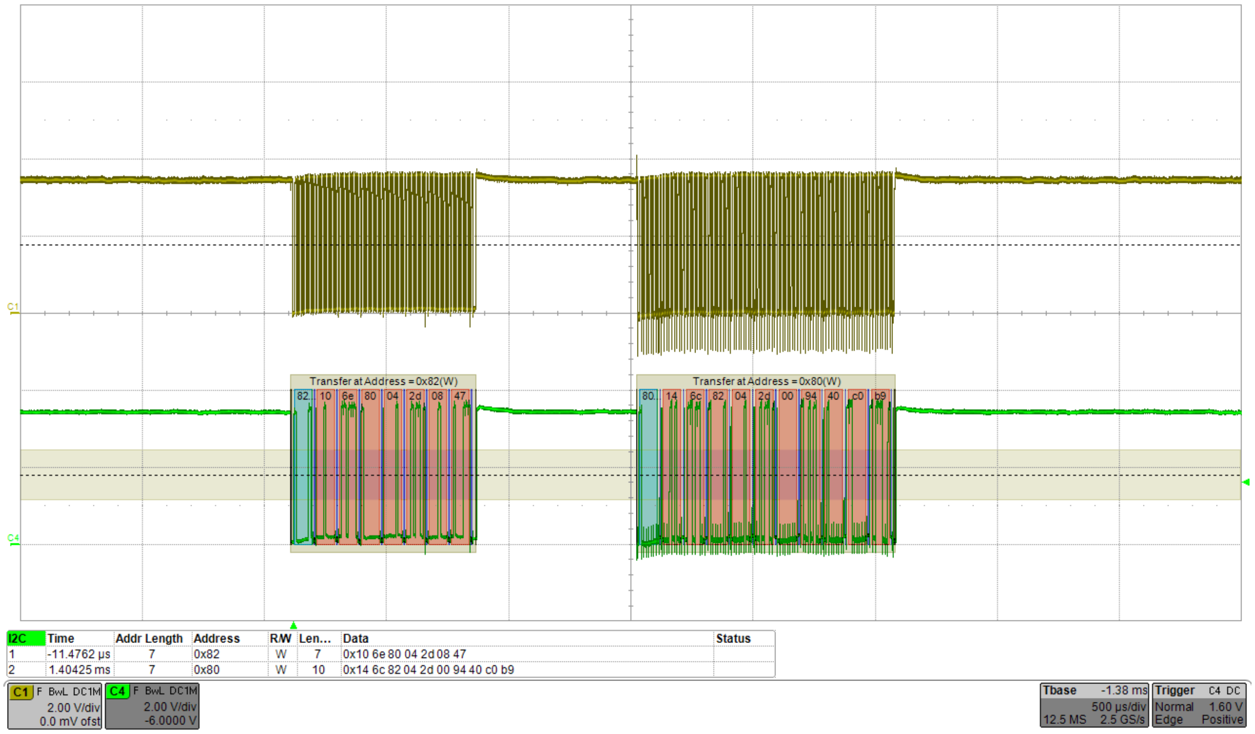



YELLOW – I²C CLOCK GREEN – I²C DATA

FIGURE 5.2: IPMI GET SENSOR READING – TEMPERATURE

Similar to the previous example, the sensor number in this request message is 11h, i.e. sensor number 17: output side card edge temperature. Response data is 67h, 103 in decimal. Converted real world value is 303 Kelvin or 30 degrees Celcius.

### 5.5.3 Get Sensor Reading – 12V voltage




YELLOW – I²C CLOCK GREEN – I²C DATA

FIGURE 5.3: IPMI GET SENSOR READING – TEMPERATURE

The sensor number in the above request message is 08h, i.e. sensor number 8: +12V output voltage. Response data is 94h, 148 in decimal. Converted real world value is 11.96V.

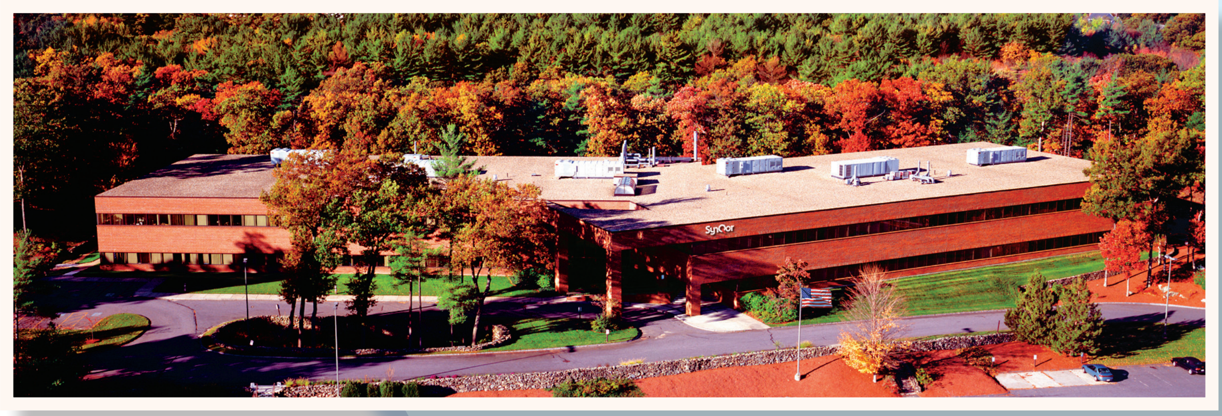
This page intentionally left blank

# VPX

# Operator's Guide

## VPX 6U I2C

## Military Grade VITA 62 Compliant Power Supply



SynQor®

Advancing The Power Curve®

Made in USA

006-0007037

VPX 6U I2C

Rev B

*1.888.567.9596 (USA only)* | **SynQor Headquarters** *155 Swanson Road, Boxborough, MA 01719-1316 USA* | **www.synqor.com**