THE XPERTS IN POWER

HPA1K5
Communication, Control and Status Specification

Revision 3.1

07/29/2020

XP Power
Arnaud Hugron

Contents

1 Introduction 5
1.1 Supported Physical Interface and Communication Protocol 5
2 References 6
3 I²C/PMBUS Protocol 7
3.1 Device address and Group Command 7
3.2 I²C Clock Speed 7
3.3 I²C Clock Stretching 7
3.4 PMBus Packet Error Checking 7
3.5 Pull-up Resistors 7
3.6 I²C Bus Lock Detection 8
3.7 PMBus Read Packet Structure 8
3.8 PMBus Write Packet Structure 8
3.9 PMBus Block Read Packet Structure 8
3.10 PMBus Block Write Packet Structure 8
3.11 Data Format for Output Voltage 9
3.12 Data Format for Other Parameters 9
3.13 PMBus Status Registers 9
3.14 Restart after latch 9
4 PMBUS Command Set 11
4.1 CLEAR_FAULTS (0x03) 13
4.2 WRITE_PROTECT (0x10) 13
4.3 STORE_DEFAULT_ALL (0x11) 13
4.4 RESTORE_DEFAULT_ALL (0x12) 13
4.5 STORE_USER_ALL (0x15) 13
4.6 RESTORE_USER_ALL (0x16) 14
4.7 VOUT_COMMAND (0x21) 14
4.8 FAN_COMMAND_1_2 (0x3B) 14
4.9 IOUT_OC_FAULT_LIMIT (0x46) 14
4.10 STATUS_BYTE (0x78) 14
4.11 STATUS_WORD (0x79) 14
4.12 STATUS_VOUT (0x7A) 15
4.13 STATUS_IOUT (0x7B) 15
4.14 STATUS_INPUT (0x7C) 15
4.15 STATUS_TEMPERATURE (0x7D) 15
4.16 STATUS_CML (0x7E) 16
4.17 STATUS_MFR_SPECIFIC (0x80) 16
4.18 STATUS_FAN_1_2 (0x81) 16
4.19 STATUS_FAN_3_4 (0x82) 16
4.20 SLAVE_ID (0xD3) 17
4.21 SLAVE_BASE_ADR (0xD4) 17

4.22 CANBUS_BIT_RATE (0xD5) 17
4.23 USER_CONFIGURATION (0xD6) 17
4.24 SERIAL_COMM_CONFIG (0xD7) 18
4.25 HARDWARE_CONFIG (0xDE) 19
4.26 READ_DATA_PFC1 (0xE0) 19
4.27 READ_INFO_PFC1 (0xE3) 19
4.28 READ_CONDITION (0xE6) 20
4.29 READ_OUTPUT (0xE7) 20
4.30 SHUTDOWN_EVENT (0XE8), SHUTDOWN_EVENT_LAST (0xE9) 20
4.31 STATUS_INTERNAL (0xEB) 21
4.32 STATE_INTERNAL (0xEC) 22
4.33 STATUS_PRIMARY (0xED) 22
4.34 XXX_FAULT_RESPONSE 22
4.34.1 XXX_FAULT_RESPONSE to Voltage/Temperature 22
4.34.2 IOUT_FAULT_RESPONSE 23
5 Modbus Protocol 25
5.1 Modbus Physical Communication 25
5.2 Modbus Message RTU Framing 25
5.3 Modbus Function Code 26
5.3.1 Modbus Function Code Read Holding Registers (0x03) 26
5.3.2 Modbus Function Code Read Input Registers (0x04) 27
5.3.3 Modbus Function Code Write Single Registers (0x06) 28
5.3.4 Modbus Function Code Write Multiple Registers (0x10) 29
5.4 Modbus Exception Responses 30
6 Physical Serial Communication 31
6.1 Serial UART, RS232 Connection 31
6.2 Serial RS485 Half Duplex Connection 31
6.3 Serial RS485 Full Duplex Connection 32
7 CANopen protocol 33
7.1 CANBus Physical Communication 33
7.2 Frame Format 33
7.3 Identifier Setup 33
7.4 Object Dictionary 34
7.5 Service Data Object (SDO) 34
7.5.1 Service SDO download initiate (HPx write) 34
7.5.2 Service SDO download segment (HPx write) 35
7.5.3 Service SDO upload initiate (HPx read) 35
7.5.4 Service SDO upload segment (HPx read) 36
7.5.5 Service SDO abort transfer 36
8 SCPI Protocol 38
8.1 SCPI Physical Communication 38

8.2 SCPI Programming Commands 38
9 LED and Signals 41

1 Introduction

This document describes Status/Control signals and supported protocol commands of HPx single phase power supply family.

HPx supports two means of control and monitoring:

- Digital communication:
  - PMBus over I²C/SMBUS (standard)
  - CANOpen over CANBus (standard)
  - Modbus (RTU) or SCPI protocol over UART, RS232, RS485 full/half duplex. The standard HPx is configured with Modbus over RS485 Full Duplex. SCPI protocol and RS485 half-duplex can be set using HARDWARE_CONFIG (0xDE) command.
  - Physical protocols like UART and RS232 are optional and require a specific control board.
- Analog programming can be selected using 'PMBUS_EN" signal pin at the output signal connector or by a digital user configuration (see USER_CONFIGURATION register for more details)

Digital communication allows:

- Voltage, current and temperature monitoring
- Voltage and current control
- Fan speed control and monitoring
- Fault and Status
- Fault Limit Setting (OVP, OCP, OTP)
- Fault Response
- ON/OFF control
- Specific configuration or feature

Analog programming allows:

- Voltage programming (VPROG signal)
- Current limit programming (IPROG signal), Maximum programming is set by digital current limit specified in IOUT_OC_FAULT_LIMIT register.
- Fault and Status (AC OK, DC OK, FAN OK, TEMPERATURE OK)

Both REMOTE ON/OFF signal and digital ON/OFF control have to be in ON state to turn ON the power supply.

1.1 Supported Physical Interface and Communication Protocol

Supported communication protocol for HPx:

Physical Interface	Communication Protocol	Comments
I²C/SMBUS (Standard Feature)	PMBus	Speed 100KHz or more depending on bus line capacitance
CAN bus (Standard Feature)	CANOpen (2)	Default Bit rate: 125Kbits/s
RS485 Full-duplex(1)(3) (Standard Feature)	Modbus (RTU) (2) (Standard Setting)	Default baudrate: 19200 baud, even parity, 1 stop bit
	SCPI(4)	Default baudrate: 19200 baud, even parity, 1 stop bit
RS485 Half-duplex(1)(3) (Optional Setting)	Modbus (RTU) (2)	Default baudrate: 19200 baud, even parity, 1 stop bit
	SCPI(4)	Default baudrate: 19200 baud, even parity, 1 stop bit
UART(1) (TTL level) (Optional Feature)	Modbus (RTU)(2)	Default baudrate: 19200 baud, even parity, 1 stop bit
	SCPI(4)	Default baudrate: 19200 baud, even parity, 1 stop bit
RS232(1) (Optional Feature)	Modbus (RTU) (2)	Default baudrate: 19200 baud, even parity, 1 stop bit
	SCPI(4)	Default baudrate: 19200 baud, even parity, 1 stop bit

(1) Only one serial physical interface can be implemented at a time. Customer can also specify one of the communication protocols (SCPI or Modbus) for the serial interface.
(2) Modbus/CANOpen protocol implement the same command set as PMBus protocol.
(3) RS485 Full Duplex or Half Duplex can be selected using PMBus command (0xDE, HARDWARE_CONFIG)
(4) SCPI or Modbus protocol can be selected using PMBus command (0xDE, HARDWARE_CONFIG)

2 References

1. PMBus Power System Management Protocol Specification Revision 1.2
2. SMBUS Specification Version 3.0
3. Modbus over serial line specification and implementation guide V1.0 (http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf)
4. Modbus application protocol specification V1.1b3
5. CiA 301 v4.2, CANopen application layer and communication profile
6. Standard Commands for Programmable Instruments (SCPI) Version 1999.0

This document makes frequent mention of PMBus specification. The specification is published by the Power Management Bus Implementers Forum and is available from http://pmbus.org.

3 I²C/PMBUS Protocol

HPx power supply behaves like a slave device across the I²C/SMBUS bus. It cannot send any request or generate a command. It will only respond to command initiated by customer host controller or system.

The power supply is compliant to I²C/SMBUS hardware communication protocol and to PMBus interface protocol.

3.1 Device address and Group Command

Signals A2, A1, A0 in the output signal connector are used to differentiate up to 8 power supplies on the same I²C bus.

Device	Address	Address Bits (MSB to LSB)							
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
HPx	0xBx	1	0	1	1	A2	A1	A0	R/W
Global Broadcast	0x00	0	0	0	0	0	0	0	W

Global Broadcast address is used to send a single write command to all HPx connected to the same I²C bus. All HPx will execute the command almost at the same time. The user will need to check the status of each power supply to check that the broadcast command was effectively received and executed.

Global Broadcast should only be used for 'write' command only not for 'read' instruction. If a Global Broadcast address is used for a 'read' instruction, invalid data and/or invalid Packet Error Checking (PEC) may occur.

HPx supports Group Command Protocol. It is used to send commands to more than one PMBUS device. Commands are sent in one continuous transmission. The STOP condition will trigger execution of the command that they received. (See PMBus Specification Part I)

For CANOpen, the define device address is divided by two to form the CANOpen Node ID.
i.e.: if Device Address = 0xBE (A2=1, A1=1, A0=1), CANOpen Node ID = 0x5F.

3.2 I²C Clock Speed

HPx supports I²C clock speed at 100 KHz. Faster speed (up to 400 KHz) can be used if signal rise time for SDA and SCL meet I²C specification.

3.3 I²C Clock Stretching

Each PMBus command received by the power supply requires a small amount of time in order to be interpreted and executed. In other word, a response of a read command may take few 100us or more before the data can be send back to the host. To avoid communication issue or error, the power supply I²C controller supports clock stretching. In this case, the power supply will hold the clock low until it is ready to send data back to the I²C master. The I²C host controller recognizes the clock stretching and will wait until the clock is released. A delay is needed beyond the clock stretch section if the I²C master controller does not support clock stretching feature.

3.4 PMBus Packet Error Checking

Packet Error Checking (PEC) is implemented but optional. It is using the CRC-8 format as defined in the SMBUS specification. It will be up to the master host to decide if it wants to use it or not. We highly recommend using PEC to prevent any corruption and significantly improve robustness of the PMBUS communication protocol. I²C busses are very sensitive to noise.

3.5 Pull-up Resistors

SDA and SCL communication lines have 10KΩ internal pull-up resistor to 3.3v. It is the user system responsibility to ensure that signal rise time meet I²C specification to avoid communication errors.

3.6 I²C Bus Lock Detection

The power supply I²C controller aborts communication and resets the internal I²C state if it detects that the I²C bus is held low for more than 40ms. This will prevent the I²C bus to be held low forever and prevent any further transaction.

3.7 PMBus Read Packet Structure

Example of an I²C packet transfer for a 2 bytes data read. The PEC byte (CRC) is optional but recommended for data validation.

S	Slave Adr	Wr	A	Cmd Code	A	Sr	Slave Adr	Rd	A	LSB	A	MSB	A	PEC	NA	P
1	8		1	8	1	1	8		1	8	1	8	1	8	1	1

Annotations: S – Start bit , Wr – Write bit, Sr – restart bit, Rd – Read bit,
A – Acknowledge bit, NA – no acknowledge bit, P – Stop bit

Master to Slave **Slave to Master**

3.8 PMBus Write Packet Structure

Example of an I²C packet transfer for 2 bytes data write transfer. The PEC byte (CRC) is optional but recommended for data validation.

S	Slave Adr	Wr	A	Cmd Code	A	LSB	A	MSB	A	PEC	A	P
1	8		1	8	1	8		8		8	1	1

Annotations: S – Start bit , Wr – Write bit, Sr – restart bit, Rd – Read bit,
A – Acknowledge bit, NA – no acknowledge bit, P – Stop bit

Master to Slave Slave to Master

3.9 PMBus Block Read Packet Structure

For commands returning more than two bytes of data, BLOCK READ format is used:

S	Slave Adr	Wr	A	Cmd Code	A	Sr	Slave Adr	Rd	A	Byte Count = N	A	Data1	A	Data2	A
1	8		1	8	1	1	8		1	8	1	8	1	8	1

…….	A	DataN	A	PEC	NA	P
8	1	8	1	8	1	1

Annotations: S – Start bit , Wr – Write bit, Sr – restart bit, Rd – Read bit,
A – Acknowledge bit, NA – no acknowledge bit, P – Stop bit

Master to Slave Slave to Master

The PEC byte (CRC) is optional but recommended for data validation.

3.10 PMBus Block Write Packet Structure

For commands requesting more than 2 bytes of data, BLOCK WRITE format is used:

S	Slave Adr	Wr	A	Cmd Code	A	Byte Count = N	A	Data1	A	Data2	A
1	8		1	8	1	8		8		8	1

……	A	DataN	A	PEC	A	P
8	1	8		8	1	1

Annotations: S – Start bit , Wr – Write bit, Sr – restart bit, Rd – Read bit,
A – Acknowledge bit, NA – no acknowledge bit, P – Stop bit

Master to Slave Slave to Master

The PEC byte (CRC) is optional but recommended for data validation.

3.11 Data Format for Output Voltage

For parameters related to output voltage, HPx supports linear format defined in section “VOUT_MODE Command” of the PMBus specification. The linear format uses a 16 bits unsigned mantissa for each parameter, along with an exponent that is shared by all the voltage related parameters. The exponent is reported in the bottom 5 bits of the VOUT_MODE parameter.

VOUT_MODE (0x20)

7	6	5	4	3	2	1	0
Mode			Exponent N				

VOUT_COMMAND (Linear Mode)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mantissa V															

Mode bits = 000b (linear format)
The Voltage, in volts, is calculated from the equation:

Voltage = V * 2^N

Where:

Voltage: parameter of interest in volts
V: 16 bit unsigned binary integer (command: VOUT_COMMAND, VOUT_XXX_LIMIT, READ_VOUT…)
N: 5 bit two’s complement binary integer

Example:
VOUT_COMMAND = 0x3700, VOUT_MODE = 0x16, N=-10 (1/1024v) -> 0x3700 / 1024 = 13.75v (VOUT Setting = 13.75v).
READ_VOUT = 0x1234, VOUT_MODE = 0x16, N=-10 (1/1024) -> 0x1234 / 1024 = 4.55v (VOUT Reading = 4.55v).

Note: PMBus standard assumes that all output voltages are expressed as positive numbers. VOUT_MODE register may be different for different model in order to increase resolution of the reading.

3.12 Data Format for Other Parameters

For parameters not directly related to output voltage (Vin, Iout, Pout, Pin, Temperature, Fan …), HPx supports the linear data format defined in “Linear Data Format” of the PMBus specification. This linear format is a 16 bits value that contains 11 bits two’s complement mantissa and 5 bits two’s complement exponent.

15	14	13	12	11
Exponent N				

10	9	8	7	6	5	4	3	2	1	0
Mantissa Y										

The relation between *Y*, *N* and the “real world” value is:

X = Y * 2^N

Where:

X: real world value
Y: 11 bits, signed two’s complement binary integer mantissa
N: signed 5 bits two’s complement binary integer exponent

3.13 PMBus Status Registers

Almost all of the warning or fault bits set in the PMBus status registers (0x78 to 0x82) remain set, even if the fault or warning condition is removed or corrected. See CLEAR_FAULTS command section for more details about how to clear them.

3.14 Restart after latch

Some PMBus fault response registers can be configured to latch the power supply when the fault occurred. The output can be restarted using one of the following:

- Cycle remote on/off pin
- Use command OPERATION (0x01): Write 0x00 (OFF) then 0x80 (ON)
- Recycle input power. Wait until all standby are gone after removing input power

4 PMBUS Command Set

Most of the standard PMBUS values are returned in linear data format according to PMBUS specification.

Access: RO = Read Only; R/W = Read/Write; -E = Registers can be stored into EEPROM using STORE_USER_ALL command (this memory will be used as default at power up)
WRITE_PROTECT (0x10) must be set to 0x00 to enable write to any writable register.
Byte order is LSB first, Default are specified for 24v model and may change.

Cmd Code	Command Name	Type	# bytes	Default (100v)	Comments
0x01	OPERATION	R/W-E	1	0x80	0x80: Turn ON, 0x00: Turn OFF
0x03	CLEAR_FAULTS	W	0		Clear PMBus status registers
0x10	WRITE_PROTECT	R/W	1	0x80	Write protected at power up by default 0x80: Disable all writes except WRITE_PROTECT 0x00: Enable writes to all commands **Needs to be 0x00 to enable write commands**
0x11	STORE_DEFAULT_ALL	W	0		Mfg mode only, set default factory setting
0x12	RESTORE_DEFAULT_ALL	W	0		Restore Default Manufacturing Setting (all E type registers)
0x15	STORE_USER_ALL	W	0		Store all (E type) registers into EEPROM, will also be used after power cycle
0x16	RESTORE_USER_ALL	W	0		Restore all default user setting (E type registers) (Set by STORE_USER_ALL)
0x20	VOUT_MODE	RO	1	0x16	Use linear format mode, may vary from different model, N=-10 (1/1024v), used by all cmds related to output voltage
0x21	VOUT_COMMAND	R/W-E	2	0x6000	linear format, 0x6000 for 24v model, based on VOUT_MODE exponent
0x31	POUT_MAX	RO	2	0x0AEE	linear format, available output power High Line:0x0AEE (1500w) Low Line: 0x0ABC (1400w)
0x3A	FAN_CONFIG_1_2	RO	1	0x99	Fan controlled in duty cycle, 2 fans
0x3B	FAN_COMMAND_1	R/W-E	2	0x0000	linear format, duty cycle 0 -> 100%
0x3D	FAN_CONFIG_3_4	RO	1	0x00	Fan controlled in duty cycle, depends on model if more than 2 fans
0x40	VOUT_OV_FAULT_LIMIT	R/W-E	2	0x6C00	linear format, 27v
0x41	VOUT_OV_FAULT_RESPONSE	RO	1	0x80	Shutdown, no retry, always latch on output OV
0x42	VOUT_OV_WARN_LIMIT	R/W-E	2	0x6800	linear format, 26v
0x43	VOUT_UV_WARN_LIMIT	R/W-E	2	0x5C00	linear format, 23v, 96% of nominal
0x44	VOUT_UV_FAULT_LIMIT	R/W-E	2	0x5B33	linear format, 22.8v, 95% of nominal
0x45	VOUT_UV_FAULT_RESPONSE	R/W-E	1	0x00	Continue, no Shutdown
0x46	IOUT_OC_FAULT_LIMIT	R/W-E	2	0x0044	linear format, 68 A, 108% of nominal by default max. Also set max setting for IPROG in analog programming
0x47	IOUT_OC_FAULT_RESPONSE	R/W-E	1	0x00	Continue, no Shutdown, constant current limit
0x48	IOUT_OC_LV_FAULT_LIMIT	R/W-E	2	0x0000	linear format, 0v
0x4A	IOUT_OC_WARN_LIMIT	R/W-E	2	0x0042	linear format, 42A
0x4D	OT_PRI_WARN_LIMIT	R/W-E	2	0x0056	linear format, 86°C
0x4E	OT_PRI_FAULT_LIMIT	R/W-E	2	0x005A	linear format, 90°C
0x4F	OT_SEC_FAULT_LIMIT	R/W-E	2	0x006E	linear format, 110°C
0x50	OT_FAULT_RESPONSE	R/W-E	1	0xC0	SD delay, retry when fault is gone
0x51	OT_SEC_WARN_LIMIT	R/W-E	2	0x006A	linear format, 106 °C
0x55	VIN_OV_FAULT_LIMIT	RO	2	0x010E	linear format, 270v
0x56	VIN_OV_FAULT_RESPONSE	R/W-E	1	0xC0	SD delay, retry when fault is gone
0x57	VIN_OV_WARN_LIMIT	RO	2	0x010C	linear format, 268v
0x58	VIN_UV_WARN_LIMIT	RO	2	0x57	linear format, 87v
0x59	VIN_UV_FAULT_LIMIT	RO	2	0x55	linear format, 85v
0x5A	VIN_UV_FAULT_RESPONSE	R/W-E	1	0x70	SD Delay, retry 6 times

0x78	STATUS_BYTE	RO	1	0x00	Summary of most critical faults
0x79	STATUS_WORD	RO	2	0x0000	Summary of unit's fault condition
0x7A	STATUS_VOUT	RO	1	0x00	
0x7B	STATUS_IOUT	RO	1	0x00	
0x7C	STATUS_INPUT	RO	1	0x00	
0x7D	STATUS_TEMPERATURE	RO	1	0x00	
0x7E	STATUS_CML	RO	1	0x00	
0x7F	STATUS_OTHER	RO	1	0x00	Not Used
0x80	STATUS_MFR_SPECIFIC	RO	1	0x00	Manufacturer Specific status
0x81	STATUS_FAN_1_2	RO	1	0x00	
0x82	STATUS_FAN_3_4	RO	1	0x00	
0x88	READ_VIN	RO	2		linear format, VIN RMS voltage
0x8B	READ_VOUT	RO	2		linear format
0x8C	READ_IOUT	RO	2		linear format
0x8D	READ_TEMPERATURE_1	RO	2		linear format, °C (Secondary Sensor)
0x8E	READ_TEMPERATURE_2	RO	2		linear format, °C (Primary Sensor)
0x90	READ_FAN_SPEED_1	RO	2		linear format, RPM
0x91	READ_FAN_SPEED_2	RO	2		linear format, RPM (if applicable)
0x92	READ_FAN_SPEED_3	RO	2		linear format, RPM (if applicable)
0x93	READ_FAN_SPEED_4	RO	2		linear format, RPM (if applicable)
0x96	READ_POUT	RO	2		linear format
0x99	MFR_ID	RO	16		Write protected (Block Read/Write)
0x9A	MFR_MODEL	RO	32		Write protected (Block Read/Write)
0x9B	MFR_REVISION	RO	4		Write protected (Block Read/Write)
0x9C	MFR_LOCATION	RO	16		Write protected (Block Read/Write)
0x9D	MFR_DATE	RO	6		Write protected (Block Read/Write), YYMMDD
0x9E	MFR_SERIAL	RO	16		Write protected (Block Read/Write)
0xA0	MFR_VIN_MIN	RO	2	0x005A	linear format, 90v
0xA1	MFR_VIN_MAX	RO	2	0x0108	linear format, 264v
0xA2	MFR_IIN_MAX	RO	2	0x000A	linear format, 10A
0xA3	MFR_PIN_MAX	RO	2	0x0B52	linear format, 1700w
0xA4	MFR_VOUT_MIN	RO	2	0x0000	linear format, 0v
0xA5	MFR_VOUT_MAX	RO	2	0x64CD	linear format, 25.2v
0xA6	MFR_IOUT_MAX	RO	2	0x003F	linear format, 63A
0xA7	MFR_POUT_MAX	RO	2	0x0AEE	linear format, 1500w
0xA8	MFR_TAMBIENT_MAX	RO	2	0x0032	linear format, 50°C
0xA9	MFR_TAMBIENT_MIN	RO	2	0x07EC	linear format, -20°C
0xAD	MFR_PRODUCT_CODE	RO	2		Product Code: - 0x0102: HPA1K5-24 (RS485, CANBUS, PMBUS) - 0x0105: HPA1K5-48 (RS485, CANBUS, PMBUS)
0xB0	USER_DATA_00	R/W-E	16		16 bytes user block, (Block Read/Write format)
0xB1	USER_DATA_01	R/W-E	16		16 bytes user block, (Block Read/Write format)
0xD0	FIRMWARE_REVISION	RO	1		
0xD1	RUN_TIME	RO	3		Number of operating Hours, (Block format)
0xD2	VOUT_RAMP_UP	R/W-E	2	0x0028	When using voltage soft start (no SOC), defines ramp up time (0 to 100% volt) in ms (40ms <-> 1000ms). linear format, default: 40ms
0xD3	SLAVE_ID	R/W-E	1	0x00	If 0x00, slave ID is based on register 0xD4 (SLAVE_BASE_ADR) below. If different than 0x00 (0x01->0xFF), this register will be used as the effective slave ID. Register 0xD4 will be ignored as well as A2-A0 address lines.
0xD4	SLAVE_BASE_ADR	R/W-E	1	0xB0	Defines the default base address of the slave ID (high nibble). It is used in conjunction with address line A2-A0 at the signal connector (low nibble).
0xD5	CANBUS_BIT_RATE	R/W-E	4	125000	CAN Bus data rate in bits/s (Block Format) Default: 125 Kbits/s
0xD6	USER_CONFIGURATION	R/W-E	2	0x0300	User option/feature (enable/disable)

0xD7	SERIAL_COMM_CONFIG	R/W-E	8		Serial communication settings (Block format) Default: 19200 baud, even parity, 1 stop bit
0xDE	HARDWARE_CONFIG	R/W-E	1	0x00	Some hardware configuration settings, it is recommended to use I2C to configure this register
0xDF	VOUT_RAMP_DOWN	R/W-E	2	0x0028	Defines ramp down time (100% to 0% volt) in ms (40ms <-> 1000ms). At light load, smart preload may extend ramp down time if timing is too long. linear format, default: 40ms
0xE0	READ_DATA_PFC1	RO	9		Block format, pfc data structure
0xE3	READ_INFO_PFC1	RO	18		Block format, pfc info structure
0xE6	READ_CONDITION	RO	8		Block format, return hottest secondary and primary temperature, min fan speed and Vin RMS
0xE7	READ_OUPUT	RO	8		Block format, return VOUT, IOUT, POUT and STATUS_WORD register
0xE8	SHUTDOWN_EVENT	RO	4		Block format, indicate shutdown reason, 32 bits
0xE9	SHUTDOWN_EVENT_LAST	RO	4		Block format, previous shutdown reason, 32 bits
0xEA	Reserved				
0xEB	STATUS_INTERNAL	RO	4		Block format, internal status condition, 32 bits
0xEC	STATE_INTERNAL	RO	2		Operating State
0xED	STATUS_PRIMARY	RO	2		Primary Status
0xEE	FAN_DUTY_CYCLE	RO	2	0x0000	Linear format, internal fan control in duty cycle (%)

4.1 CLEAR_FAULTS (0x03)

Almost all of the warning or fault bits set in the PMBus status registers (0x78 to 0x82) remain set even if the fault or warning condition is removed or corrected, until one of the following occur:

- The device receives a CLEAR_FAULTS command,
- The output is commanded through REMOTE_OFF signal pin, the OPERATION command, or the combined action of the REMOTE_OFF pin and OPERATION command, to turn off and then to turn back on,
- Bias power is removed.

4.2 WRITE_PROTECT (0x10)

The WRITE_PROTECT command is used to control writing to the PMBus device. The intent of this command is to provide protection against accidental changes or memory corruption. By default, or after a power up cycle, the register is always set to 0x80 (Disable all writes except to WRITE_PROTECT command).

Register value	Description
0x80	Disable all writes except to the WRITE_PROTECT command
0x40	Disable all writes except to the WRITE_PROTECT, OPERATION and PAGE commands
0x20	Disable all writes except to the WRITE_PROTECT, OPERATION, PAGE, ON_OFF_CONFIG and VOUT_COMMAND commands
0x00	Enable writes to all commands.

Note: WRITE_PROTECT register must be set to 0x00 in order to enable write to any writable register. It is recommended to set it back to 0x80 (Disable all writes) when all updates are done to avoid corruption.

4.3 STORE_DEFAULT_ALL (0x11)

This command can only be used in factory mode.

4.4 RESTORE_DEFAULT_ALL (0x12)

This command restores all (-E) registers to factory default parameters. The factory default cannot be changed by the user.

4.5 STORE_USER_ALL (0x15)

This command saves all (-E) registers in the non-volatile user memory. This memory will be used as default after power up.

4.6 RESTORE_USER_ALL (0x16)

This command restores all (-E) registers to user default parameters.

4.7 VOUT_COMMAND (0x21)

This command is used to change the output voltage. It can also be used to change the default output voltage at power up by updating this register and saving it by using STORE_USER_ALL command. The format of the data is specified in section “Data Format for Output Voltage”. WRITE_PROTECT register must be set correctly before writing to this register or for any write command.

4.8 FAN_COMMAND_1_2 (0x3B)

This command can control the fan speed by duty cycle (0 to 100). The register value is only used if it is higher than the internal fan control. Writing 0 will leave entire fan control to the power supply.

4.9 IOUT_OC_FAULT_LIMIT (0x46)

This command is used to change the current limit. It can also be used to change the default output current limit at power up by updating this register and saving it by using STORE_USER_ALL command. By default, the power supply will work as a constant current limit if the output current exceeds the limit. The response at current limit can be changed using IOUT_OC_FAULT_RESPONSE register.

4.10 STATUS_BYTE (0x78)

This command returns an abbreviated status for fast reads. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	BUSY	A fault was declared because the device was busy and unable to respond.
6	OFF	This bit is asserted if the unit is not providing power to the output, regardless of the reason, including simply not being enabled.
5	VOUT_OV_FAULT	An output overvoltage fault has occurred
4	IOUT_OC_FAULT	An output overcurrent fault has occurred
3	VIN_UV_FAULT	An input under voltage fault has occurred
2	TEMPERATURE	A temperature fault or warning has occurred
1	CML	A communications, memory or logic fault has occurred
0	NONE_OF_THE_ABOVE	A fault or warning not listed in bits [7:1] has occurred

4.11 STATUS_WORD (0x79)

Command returns the general status information used to indicate subsequent status to be read for more detail. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
15	VOUT	An output voltage fault or warning has occurred
14	IOUT/POUT	An output current or output power fault or warning has occurred
13	INPUT	An input voltage, input current, or input power fault or warning has occurred
12	MFR_SPECIFIC	A manufacturer specific fault or warning has occurred
11	POWER_GOOD#	The POWER_GOOD signal, if present, is negated[1]
10	FANS	A fan or airflow fault or warning has occurred
9	OTHER	A bit in STATUS_OTHER is set
8	UNKNOWN	A fault type not given in bits [15:1] of the STATUS_WORD has been detected
7	BUSY	A fault was declared because the device was busy and unable to respond.
6	OFF	This bit is asserted if the unit is not providing power to the output, regardless of the reason, including simply not being enabled.
5	VOUT_OV_FAULT	An output overvoltage fault has occurred
4	IOUT_OC_FAULT	An output overcurrent fault has occurred
3	VIN_UV_FAULT	An input under voltage fault has occurred
2	TEMPERATURE	A temperature fault or warning has occurred
1	CML	A communications, memory or logic fault has occurred
0	NONE_OF_THE_ABOVE	A fault or warning not listed in bits [7:1] has occurred

4.12 STATUS_VOUT (0x7A)

Command returns the output voltage related status. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	VOUT_OV_FAULT	Output Overvoltage Fault
6	VOUT_OV_WARNING	Output Overvoltage Warning
5	VOUT_UV_WARNING	Output Undervoltage Warning
4	VOUT_UV_FAULT	Output Undervoltage Fault
3	VOUT_MAX_WARNING	Not Used
2	TON_MAX_FAULT	Not Used
1	TOFF_MAX_WARNING	Not Used
0	VOUT Tracking Error	Not Used

4.13 STATUS_IOUT (0x7B)

Command returns the output current related status. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	OUT_OC_FAULT	Output Overcurrent Fault
6	IOUT_OC_LV_FAULT	Output Overcurrent and Low Voltage Fault
5	IOUT_OC_WARNING	Output Overcurrent Warning
4	IOUT_UC_FAULT	Not Used, Output Undercurrent Fault
3	Current Share Fault	Not Used
2	IN_POWER_LIMIT	In Power Limit or in constant current
1	POUT_OP_FAULT	Not Used, Output Overpower Fault
0	POUT_OP_WARNING	Not Used, Output Overpower Warning

4.14 STATUS_INPUT (0x7C)

Command returns status specific to the input. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	VIN_OV_FAULT	Input Overvoltage Fault
6	VIN_OV_WARNING	Input Overvoltage Warning
5	VIN_UV_WARNING	Input Undervoltage Warning
4	VIN_UV_FAULT	Input Undervoltage Fault
3	Unit Off	Not Used, Unit Off for Insufficient Input Voltage
2	IIN_OC_FAULT	Not Used, Input Overcurrent Fault
1	IIN_OC_WARNING	Not Used, Input Overcurrent Warning
0	PIN_OP_WARNING	Not Used, Input Overpower Warning

4.15 STATUS_TEMPERATURE (0x7D)

Command returns the temperature specific status. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	OT_FAULT	Over temperature Fault
6	OT_WARNING	Over temperature Warning
5	UT_WARNING	Not Used, Under temperature Warning
4	UT_FAULT	Not Used, Under temperature Fault
3	Reserved	
2	Reserved	
1	Reserved	
0	OT_PRELOAD	Over temperature Smart Preload (preload disable for some period of time)

4.16 STATUS_CML (0x7E)

Command returns the Communication, Logic and Memory specific status. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	INVALID_COMMAND	Invalid Or Unsupported Command Received
6	INVALID_DATA	Invalid Or Unsupported Data Received
5	PEC_FAILED	Packet Error Check Failed
4	MEMORY_FAULT	Memory Fault Detected
3	MCU_FAULT	Processor Fault Detected
2	Reserved	
1	OTHER_CML_FAULT	A communication fault other than the ones listed in this table has occurred
0	MEM_LOGIC_FAULT	Other Memory Or Logic Fault has occurred

4.17 STATUS_MFR_SPECIFIC (0x80)

Command returns the manufacturer specific status. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	Reserved	
6	SMART_PRELOAD_OV_PWR	Smart preload over power as occurred
5	SW_SHORT2_DETECTED	Short type 2 as occurred
4	SW_SHORT1_DETECTED	Short type 1 as occurred
3	SW_OVP_DETECTED	Software overvoltage as occurred
2	SW_OCP_DETECTED	Software overcurrent as occurred
1	HW_OVP_DETECTED	Hardware overvoltage as occurred
0	HW_OCP_DETECTED	Hardware overcurrent as occurred

4.18 STATUS_FAN_1_2 (0x81)

Command returns fan status. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	FAN_1_FAULT	Fan 1 Fault
6	FAN_2_FAULT	Fan 2 Fault
5	FAN_1_WARNING	Fan 1 Warning
4	FAN_2_WARNING	Fan 2 Warning
3	FAN_1_OVERRIDDEN	Fan 1 Speed Override
2	FAN_2_OVERRIDDEN	Fan 2 Speed Override
1	AIRFLOW_FAULT	Not Used, Airflow Fault
0	AIRFLOW_WARNING	Not Used, Airflow Warning

4.19 STATUS_FAN_3_4 (0x82)

Command returns fan status. See “CLEAR_FAULTS” to clear bits.

Bit #	Status Bit Name	Description
7	FAN_3_FAULT	Fan 3 Fault
6	FAN_4_FAULT	Fan 4 Fault
5	FAN_3_WARNING	Fan 3 Warning
4	FAN_4_WARNING	Fan 4 Warning
3	FAN_3_OVERRIDDEN	Fan 3 Speed Override
2	FAN_4_OVERRIDDEN	Fan 4 Speed Override
1	Reserved	
0	Reserved	

4.20 SLAVE_ID (0xD3)

If SLAVE_ID register is 0x00, HPx slave ID is based on register 0xD4 below. If different than 0x00, this register will be used as the effective slave ID. Register 0xD4 (SLAVE_BASE_ADR) will be ignored as well as A2-A0 address lines on the output connector.

Bit 0 of the SLAVE ID is ignored and set to 0 (if an odd number is entered) as bit 0 is used for R/W bit when using PMBus protocol. With this limitation, only 127 slave ID are available.

When changing this register, the new Slave ID is effective immediately. In this case, the next command needs to use the new ID. User needs to make sure that the slave ID is unique for the same communication bus.

For CANOpen, Node ID is equivalent to SLAVE_ID / 2.
i.e.: if SLAVE_ID = 0xBE (or 0xBF), CANOpen Node ID = 0x5F as CANOpen has a maximum of 127 address.

4.21 SLAVE_BASE_ADR (0xD4)

Define the default base address of the device (only bit7..bit4 is used, bit3..bit0 is ignored). It is used in conjunction with address line A2-A0 at the signal connector (low nibble). This device base address is only used if register 0xD3 (SLAVE_ID) is set to 0x00 else the value defined in 0xD3 (SLAVE_ID) will be used instead.
i.e.:
- (SLAVE_BASE_ADR = 0x40 and A2=0, A1=0, A0=1 and SLAVE_ID = 0), device address = 0x42.
- (SLAVE_BASE_ADR = 0x60 and A2=1, A1=0, A0=0 and SLAVE_ID = 0), device address = 0x68.

See chapter 3.1 (Device Address and Group Command) for more details.
User needs to make sure that the slave ID is unique for the same communication bus.

For CANOpen, the device address is divided by two to form the CANOpen Node ID.
i.e.: if device address = 0xBE (SLAVE_BASE_ADR = 0xB0 and A2=1, A1=1, A0=1 and SLAVE_ID = 0), CANOpen Node ID = 0x5F.

4.22 CANBUS_BIT_RATE (0xD5)

Command is used to configure the CANBus bit rate.

Byte#	Description
0..3	32 bits value indicating CANBus bit rate (LSB) Default is 125000 bits/s

Note: Use block format, can be saved into EEPROM using STORE_USER_ALL (0x15) command. If STORE_USER_ALL is not used, the register will be restored to the previous setting at the next power cycle.

4.23 USER_CONFIGURATION (0xD6)

Command is used to enable or disable some specific feature that can be set by the user.

Bit #	Status Bit Name	Description
15	CFG_NO_PRELOAD_IN_SD	**0**: Preload is enabled during any output SD. Will help discharge output capacitors at light or no load 1: Preload is disabled during any output SD (still enable during voltage ramp down) Note: this bit is valid only if bit 2 (CFG_PRELOAD_DISABLE) is 0. Smart preload feature is active.
14	CFG_DISABLE_VPROG	**0**: In analog mode, VPROG must be set to control output voltage 1: In analog mode, VPROG is ignored (use digital voltage setting), only IPROG signal is used to control current limit if enable
13	CFG_DISABLE_IPROG	**0**: In analog mode, IPROG and VPROG must be set to control output 1: In analog mode, IPROG is ignored (use digital current limit), only VPROG is used to control output
12	CFG_ANALOG_PROG	**0**: PMBUS_EN signal will dictated if analog or digital programming

		1: Use Analog programming regardless of PMBUS_EN signal Note: If IPROG is used, the maximum programmable current limit will be set by the digital setting in IOUT_OC_FAULT_LIMIT register. Default is 110% of nominal.
11	CFG_POTENTIOMETER_FULL_ADJ	**0**: Potentiometer can adjust output +- 10% of nominal 1: Potentiometer can adjust output from 0% to 105% of nominal Note: CFG_POTENTIOMETER_DISABLE must be clear in other to use the potentiometer else potentiometer adjustment will be ignored.
10	CFG_POTENTIOMETER_DISABLE	**0**: Enable potentiometer output adjustment 1: Disable potentiometer output adjustment
9	CFG_REMOTE_INHIBIT_LOGIC	Set the active logic for remote inhibit signal: 0: (signal = 0 -> output OFF ; signal = 1 -> output ON) **1**: (signal = 0 -> output ON ; signal = 1 -> output OFF) By default, this bit is set to 1 (Need 0v or no connection to turn ON output)
8	CFG_SYNC_PWR_ON	This signal is used to sync units during AC turn ON in parallel mode. 0: AC Turn ON Sync disable (no sync, some units may go in current limit) **1**: AC Turn ON enable. All units with their Sync signal connected will turn ON their output at the same time during AC power ON.
7	CFG_FAN_TEMP_OK_SIG_LOGIC	Set the logic for the FAN_TEMP_OK signal: **0**: (signal = 0 -> good ; 1 -> bad) 1: (signal = 1 -> good ; 0 -> bad)
6	Reserved	Reserved
5	CFG_DCOK_SIG_LOGIC	Set the logic for the DCOK signal: **0**: (signal = 0 -> good ; 1 -> bad) 1: (signal = 1 -> good ; 0 -> bad)
4	CFG_ACOK_SIG_LOGIC	Set the logic for the ACOK signal: **0**: (signal = 0 -> good ; 1 -> bad) 1: (signal = 1 -> good ; 0 -> bad)
3	CFG_FAN_OFF	**0**: Fan(s) still running during remote OFF, set to minimum 1: Turn OFF fan(s) during remote OFF if not too hot
2	CFG_PRELOAD_DISABLE	**0**: Enable smart preload 1: Disable smart preload
1	CFG_FAST_SOFTSTART_DISABLE	**0**: Enable fast safe soft start when applicable 1: Disable fast safe soft start, use ramp control voltage soft start Note: In ramp control voltage soft start, ramp up control time is defined by VOUT_RAMP_UP register.
0	CFG_CURRENT_SOFTSTART_ENABLE	**0**: Use fast safe soft start or voltage soft start (see CFG_FAST_SOFTSTART_DISABLE bit) 1: Enable Current Soft Start, use current soft start only Note: In current soft start, the current ramp up will be controlled at a rate of around 200ms from 0% to 100% load.

Note: Value in **bold** defined the default setting. Register can be saved into EEPROM using STORE_USER_ALL (0x15) command. If STORE_USER_ALL is not used, the register will be restored to the previous setting at the next power cycle.

4.24 SERIAL_COMM_CONFIG (0xD7)

Command is used to configure the serial port when using UART, RS232 or RS485 as physical communication protocol.

<table>
<tr><th>Byte#</th><th>Description</th></tr>
<tr><td>0..3</td><td>32 bits value indicating baudrate (LSB)
Default is 19200 baud (maximum baudrate setting : 921600 baud)</td></tr>
<tr><td>4</td><td>Stop Bits, Default is 1 stop bit (0)
<table>
<tr><th>Value</th><th>Stop Bit</th></tr>
<tr><td>0</td><td>1 bit</td></tr>
<tr><td>1</td><td>2 bits</td></tr>
</table></td></tr>
<tr><td>5</td><td>Parity, Default is Even parity (2)
<table>
<tr><th>Value</th><th>Parity</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>2</td><td>Even</td></tr>
<tr><td>3</td><td>Odd</td></tr>
</table></td></tr>
</table>

6	Data Bits Count, Default is 8 bits (0). 9 bits count is not used in this application Value / Bits Count / Byte 0 / 8 bits 1 / 9 bits

Note: Use block format, can be saved into EEPROM using STORE_USER_ALL (0x15) command. If STORE_USER_ALL is not used, the register will be restored to the previous setting at the next power cycle.

4.25 HARDWARE_CONFIG (0xDE)

Command is used to enable/disable or change some specific hardware related feature that can be configured by the user.

Bit #	Status Bit Name	Description
1	CFG_HW_RS485_HALF_DUPLEX	**0**: RS485 transceiver in Full Duplex mode 1: RS485 transceiver in Half Duplex mode
0	CFG_HW_SCPI_PROTOCOL_ENABLE	**0**: For serial communication, protocol is Modbus 1: For serial communication, protocol is SCPI

Note: Value in **bold** defined the default setting. Register can be saved into EEPROM using STORE_USER_ALL (0x15) command. If STORE_USER_ALL is not used, the register will be restored to the previous setting at the next power cycle. It is recommended to use I2C to configure this register to avoid communication issue.

4.26 READ_DATA_PFC1 (0xE0)

Command returns status and input reading about primary PFC.

Byte#	Description
0	Vin Peak LSB (linear format, v)
1	Vin Peak MSB (linear format, v)
2	Pfc DC Bus Voltage LSB (linear format, v)
3	Pfc DC Bus Voltage MSB (linear format, v)
4	Pfc Temperature LSB (linear format, °C)
5	Pfc Temperature MSB (linear format, °C)
6	DC Bus reference adjustment
7	Pfc Shutdown Fault

Bit#	Fault Bit Name	Description
0	AC_UV	input under voltage fault has occurred
1	AC_OV	input overvoltage fault has occurred
2	PFC_LATCH	Pfc is latched
3	PFC_UV	Pfc under voltage fault has occurred
4	PFC_OV	Pfc overvoltage voltage fault has occurred
5	PFC_TOO_LOW	Pfc voltage is too low
6	PFC_OTP	Pfc temperature fault has occurred
7	PFC_REM_OFF	Pfc remote off

8 Pfc Status

Bit#	Fault Bit Name	Description
0	PFC_READY	Pfc is boosting and ready
1	RELAY_CLOSE	Pfc relay is closed
2	AC_QUICK_FLT	AC failure detected, use for early warning
3	PFC_STATE	Pfc ON
4	HIGH_LINE	High Line
5	OT_WARNING	Over temperature warning has occurred
6	PFC_TIMEOUT	Pfc startup failed
7	AC_LOSS	AC loss detection

4.27 READ_INFO_PFC1 (0xE3)

Command returns PFC information.

Byte#	Description
0	Primary Firmware version
1	Pfc State: 0x00: PFC_START_DELAY_STATE 0x01: PFC_BUS_CHARGE_STATE 0x02: PFC_CLOSE_RELAY_STATE 0x03: PFC_ON_WAIT_STATE 0x04: PFC_FAULT_DELAY_STATE

<table>
<tr><td></td><td>0x05: PFC_ON_STATE
0x06: PFC_SHUTDOWN_STATE
0x07: PFC_HICCUP_DELAY_STATE</td></tr>
<tr><td>2</td><td>Communication Frame Error Count LSB</td></tr>
<tr><td>3</td><td>Communication Frame Error Count MSB</td></tr>
<tr><td>4</td><td>Communication Overrun Count LSB</td></tr>
<tr><td>5</td><td>Communication Overrun Count MSB</td></tr>
<tr><td>6</td><td>Communication Packet Error Count LSB</td></tr>
<tr><td>7</td><td>Communication Packet Error Count MSB</td></tr>
<tr><td>8</td><td>Communication Packet CRC Error Count LSB</td></tr>
<tr><td>9</td><td>Communication Packet CRC Error Count MSB</td></tr>
<tr><td>10</td><td>Communication Packet TimeOut Count LSB</td></tr>
<tr><td>11</td><td>Communication Packet TimeOut Count MSB</td></tr>
<tr><td>12</td><td>Relay Closed Count LSB, Saved into EEPROM every time the relay closed</td></tr>
<tr><td>13</td><td>Relay Closed Count MSB, Saved into EEPROM every time the relay closed</td></tr>
<tr><td>14</td><td>Previous PFC Shutdown Fault
<table>
<tr><th>Bit#</th><th>Fault Bit Name</th><th>Description</th></tr>
<tr><td>0</td><td>AC_UV</td><td>input under voltage fault has occurred</td></tr>
<tr><td>1</td><td>AC_OV</td><td>input overvoltage fault has occurred</td></tr>
<tr><td>2</td><td>PFC_LATCH</td><td>Pfc is latched</td></tr>
<tr><td>3</td><td>PFC_UV</td><td>PFC under voltage fault has occurred</td></tr>
<tr><td>4</td><td>PFC_OV</td><td>PFC overvoltage voltage fault has occurred</td></tr>
<tr><td>5</td><td>PFC_TOO_LOW</td><td>PFC voltage is too low</td></tr>
<tr><td>6</td><td>PFC_OTP</td><td>PFC temperature fault has occurred</td></tr>
<tr><td>7</td><td>PFC_REM_OFF</td><td>PFC remote off</td></tr>
</table></td></tr>
<tr><td>15</td><td>Reserved</td></tr>
<tr><td>16</td><td>Pfc UV limit LSB (linear Format)</td></tr>
<tr><td>17</td><td>Pfc UV limit MSB (linear Format)</td></tr>
</table>

4.28 READ_CONDITION (0xE6)

Command returns multiple registers about operating condition in a single command.

Byte#	Description
0	Hottest Secondary Temperature LSB (linear format, °C)
1	Hottest Secondary Temperature MSB (linear format, °C)
2	Hottest Primary Temperature LSB (linear format, °C)
3	Hottest Primary Temperature MSB (linear format, °C)
4	Lowest Fan Speed LSB (linear format, RPM)
5	Lowest Fan Speed MSB (linear format, RPM)
6	Vin RMS LSB (linear format, V)
7	Vin RMS MSB (linear format, V)

4.29 READ_OUTPUT (0xE7)

Command returns multiple registers about output in a single command.

Byte#	Description
0	Vout LSB (linear format, use VOUT_MODE, V)
1	Vout MSB (linear format, use VOUT_MODE, V)
2	Iout LSB (linear format, A)
3	Iout MSB (linear format, A)
4	Pout LSB (linear format, W)
5	Pout MSB (linear format, W)
6	STATUS_WORD LSB (see STATUS_WORD register)
7	STATUS_WORD MSB (see STATUS_WORD register)

4.30 SHUTDOWN_EVENT (0XE8), SHUTDOWN_EVENT_LAST (0xE9)

Command returns the shutdown reason (0xE8) and the previous shutdown reason (0xE9) of the power supply.

Bit #	Status Bit Name	Description
31	PFC3_COMM_FAULT	Primary to Secondary communication #3 failure
30	PFC2_COMM_FAULT	Primary to Secondary communication #2 failure
29	PFC1_COMM_FAULT	Primary to Secondary communication #1 failure
28	IOUTX_OCP_FAULT	Overcurrent fault has occurred in one of the converter channels
27	PMBUS_CMD_OFF	PMBus Command OFF
26	REMOTE_OFF	Remote OFF
25	PFC_OTP_FAULT	Pfc temperature fault has occurred
24	FAN_FAULT	Fan fault has occurred
23	SEC3_OTP_FAULT	Secondary temperature #3 fault has occurred
22	SEC2_OTP_FAULT	Secondary temperature #2 fault has occurred
21	SEC1_OTP_FAULT	Secondary temperature #1 fault has occurred
20	PFC3_OTP_FAULT	Pfc temperature #3 fault has occurred
19	PFC2_OTP_FAULT	Pfc temperature #2 fault has occurred
18	PFC1_OTP_FAULT	Pfc temperature #1 fault has occurred
17	LATCH_FAULT	A fault has latched the power supply
16	OUTPUT_UV_FAULT	Output under voltage fault has occurred
15	HOLD_FAULT	Hold fault until the restart timer has expired
14	SW_OCP_FAULT	Software overcurrent fault has occurred, (firmware test)
13	HW_OCP_FAULT	Hardware overcurrent fault has occurred, (hardware comparator)
12	IOUTX_IMBALANCE	Output Current imbalance fault has occurred
11	Reserved	
10	SW_OVP_FAULT	Software output overvoltage fault has occurred, (firmware test)
9	USER_OVP_FAULT	User setting output overvoltage fault has occurred, (internal comparator)
8	HW_OVP_FAULT	Hardware output overvoltage fault has occurred, (hardware comparator)
7	INPUT_OVP_FAULT	Input overvoltage fault has occurred
6	PFC3_BAD_FAULT	Pfc #3 fault has occurred
5	PFC2_BAD_FAULT	Pfc #2 fault has occurred
4	PFC1_BAD_FAULT	Pfc #1 fault has occurred
3	INPUT_IMBALANCE_FAULT	Input imbalance fault has occurred
2	INPUT3_BAD_FAULT	Input #3 fault has occurred
1	INPUT2_BAD_FAULT	Input #2 fault has occurred
0	INPUT1_BAD_FAULT	Input #1 fault has occurred

4.31 STATUS_INTERNAL (0xEB)

Command returns internal state or status of the secondary controller.

Bit #	Status Bit Name	Description
31	SEC3_OTP_FAULT	Secondary temperature #3 fault has occurred
30	SEC2_OTP_FAULT	Secondary temperature #2 fault has occurred
29	SEC1_OTP_FAULT	Secondary temperature #1 fault has occurred
28	PFC3_OTP_FAULT	Pfc temperature #3 fault has occurred
27	PFC2_OTP_FAULT	Pfc temperature #2 fault has occurred
26	PFC1_OTP_FAULT	Pfc temperature #1 fault has occurred
25	PFC_OTP_FAULT	Pfc temperature fault has occurred
24	Reserved	
23	DC_OK_FAST	Use for fast output under voltage detection
22	Reserved	
21	ISHARE_ENABLE	Ishare pin is enabled
20	Reserved	
19	PRELOAD_ENABLE	Preload is enabled
18	PRELOAD_OVERLOAD	Preload is in overloaded condition
17	SOFT_START	Power supply in soft start condition
16	Reserved	

15	PRIMARY_CALIBRATION	Primary calibration mode
14	SECONDARY_CALIBRATION	Secondary calibration mode
13	HW_OVP_TEST	Hardware OVP test mode
12	CURRENT_SOFTSTART	Soft start in current mode
11	VSTANDBY_ENABLE	5v output standby enable
10	ANALOG_PROG	Analog programming enabled (VPROG and IPROG signal)
9	OUTPUT_UNDER_VOLTAGE	Output under voltage condition
8	TEMPERATURE_COLD	Temperature is cold
7	FAN_FAULT	Fan fault has occurred
6	TEMPERATURE_TOO_COLD	Temperature is very cold
5	OTP_FAULT	Temperature fault has occurred
4	OTP_WARNING	Temperature warning has occurred
3	PWM_ENABLE	Converter PWM is running
2	INPUT_HIGH_LINE	Input High Line
1	IN_POWER_LIMIT	In power limit
0	IN_CURRENT_LIMIT	In current limit

4.32 STATE_INTERNAL (0xEC)

Indicates the state of the main output controller:

0x0000: Initialization state
0x0001: OFF state
0x0002: Current soft start state
0x0003: Voltage soft start state
0x0004: Reserved
0x0005: ON state

4.33 STATUS_PRIMARY (0xED)

Command returns the general status information about all PFCs.

Bit #	Status Bit Name	Description
15	VIN_IMBALANCE	Input imbalance has occurred
14	VIN_LOSS	Input loss has occurred
13	VIN_LOW	Input is low
12	Reserved	
11	PFC_OT_WARNING	Pfc over temperature warning
10	PFC_OT_FAULT	Pfc over temperature fault
9	PFC_LATCH	Pfc is in latched state
8	PFC_STATE_OFF	Pfc is off
7	BOOST_NOT_READY	Pfc boost is not ready
6	PFC_TOO_LOW	Pfc voltage is too low
5	PFC_OV_FAULT	Pfc overvoltage fault
4	PFC_UV_FAULT	Pfc under voltage fault
3	RELAY_OPEN	Relay is open
2	INPUT_OV_FAULT	Input overvoltage fault
1	INPUT_UV_FAULT	Input under voltage fault
0	INPUT_UV_EARLY_WARNING	Input under voltage early warning

4.34 XXX_FAULT_RESPONSE

These commands specify how the power supply behaves during a fault.

4.34.1 XXX_FAULT_RESPONSE to Voltage/Temperature

The data byte specifying the response to a voltage or temperature fault is detailed below:

Bit #	Description	Value	Meaning
7:6	Response	00	The PMBus device continues operation without interruption.
		01	The PMBus device continues operation for the delay time specified by bits [2:0] and the delay time unit specified for that particular fault. If the fault condition is still present at the end of the delay time, the unit responds as programmed in the Retry Setting (bits [5:3]).
		10	The device shuts down (disables the output) and responds according to the retry setting in bits [5:3].
		11	The device's output is disabled while the fault is present. Operation resumes and the output is enabled when the fault condition no longer exists.
5:3	Retry Setting	000	A zero value for the Retry Setting means that the unit does not attempt to restart. The output remains disabled until the fault is cleared.
		001 - 110	The PMBus device attempts to restart the number of times set by these bits. The minimum number is 1 and the maximum number is 6. If the device fails to restart (the fault condition is no longer present and the device is delivering power to the output and operating as programmed) in the allowed number of retries, it disables the output and remains off until the fault is cleared. The time between the start of each attempt to restart is set by the value in bits [2:0] along with the delay time unit specified for that particular fault.
		111	The PMBus device attempts to restart continuously, without limitation, until it is commanded OFF (by the CONTROL pin or OPERATION command or both), bias power is removed, or another fault condition causes the unit to shut down.
2:0	Delay Time	xxx	The number of delay time units, which vary depending on the type of fault. This delay time is used for either the amount of time a unit is to continue operating after a fault is detected or for the amount of time between attempts to restart.

Supported fault response:

Cmd Code	Command	Response Supported	Shutdown Delay	Retry Setting	Delay Time
0x41	VOUT_OV_FAULT_RESPONSE (default: 0x80, SD, no retry, latch output), register is read only.	10	0ms SD immediately	000 No retry	000 No restart Delay
0x45	VOUT_UV_FAULT_RESPONSE (default: 0x00, continue, do nothing)	00, 01, 10, 11	20ms * Delay Time (0ms -> 140ms)	000 – 111 111 = Max 20 retries	10ms * Delay Time (0 -> 70ms)
0x50	OT_FAULT_RESPONSE (default: 0xC0, SD and resume when fault is gone) (apply for FAN_FAULT also)	01, 10, 11	10s	000 – 111 111 = Max 20 retries	10s
0x56	VIN_OV_FAULT_RESPONSE (default: 0xC0, SD and resume when fault is gone)	01, 10, 11	6s	000 – 111 111= Max 20 retries	6s
0x5A	VIN_UV_FAULT_RESPONSE (default: 0x77, SD delay and retry up to 6 times)	01, 10, 11	0.6s	000 – 111 111 = Max 20 retries	6s

4.34.2 IOUT_FAULT_RESPONSE

The data byte specifying the response to an output current fault is detailed below according to PMBus specification:

Bit #	Description	Value	Meaning
7:6	Response	00	The PMBus device continues to operate indefinitely while maintaining the output current at the value set by IOUT_OC_FAULT_LIMIT without regard to the output voltage (known as constant-current).
		01	The PMBus device continues to operate indefinitely while maintaining the output current at the value set by IOUT_OC_FAULT_LIMIT as long as the output voltage remains above the minimum value specified by IOUT_OC_LV_FAULT_LIMIT. If the output voltage is pulled down to less than that value, then the PMBus device shuts down and responds according to the Retry setting in bits [5:3].
		10	The PMBus device continues to operate, maintaining the output current at the value set by IOUT_OC_FAULT_LIMIT without regard to the output voltage, for

			the delay time set by bits [2:0] and the delay time units for specified in the IOUT_OC_FAULT_RESPONSE. If the device is still operating in current limiting at the end of the delay time, the device responds as programmed by the Retry Setting in bits [5:3].
		11	The PMBus device shuts down and responds as programmed by the Retry Setting in bits [5:3].
5:3	Retry Setting	000	A zero value for the Retry Setting means that the unit does not attempt to restart. The output remains disabled until the fault is cleared.
		001 - 110	The PMBus device attempts to restart the number of times set by these bits. The minimum number is 1 and the maximum number is 6. If the device fails to restart (the fault condition is no longer present and the device is delivering power to the output and operating as programmed) in the allowed number of retries, it disables the output and remains off until the fault is cleared. The time between the start of each attempt to restart is set by the value in bits [2:0] along with the delay time unit specified for that particular fault.
		111	The PMBus device attempts to restart continuously, without limitation, until it is commanded OFF (by the CONTROL pin or OPERATION command or both), bias power is removed, or another fault condition causes the unit to shut down.
2:0	Delay Time	xxx	The number of delay time units, which vary depending on the type of fault. This delay time is used for either the amount of time a unit is to continue operating after a fault is detected or for the amount of time between attempts to restart.

Supported fault response for IOUT_OC_FAULT_RESPONSE:

Cmd Code	Command	Response Supported	Shutdown Delay	Retry Setting	Delay Time
0x47	IOUT_OC_FAULT_RESPONSE (default: 0x00)	00, 01, 10,11	50ms * Delay Time (0ms -> 350ms)	000 – 111 111 = Max 20 retries	10ms * Delay Time (0ms -> 70ms)

Default register value is 0x00 for IOUT_OC_FAULT_RESPONSE: Continues to operate indefinitely while maintaining the output current at the value set by IOUT_OC_FAULT_LIMIT without regard to the output voltage (known as constant-current).

5 Modbus Protocol

Communication on a Modbus network is initiated by a master with a query to a slave. After receiving and processing the query, the slave returns a reply to the master. The master can address individual slaves or uses a special broadcast address (0x00) to initiate a broadcast message to all slaves. No response is returned to broadcast requests.

HPx only supports Modbus RTU. ASCII mode is not supported. Only one command (See chapter 4 table) can be read or written in a frame. A Modbus register is always 2 bytes of data (MSB first) even if the command table in chapter 4 specifies only 1 byte (# bytes column). In this case, MSB byte will always be 0.

See chapter (3.1 Device address and Group Command) about how the HPx address is set.

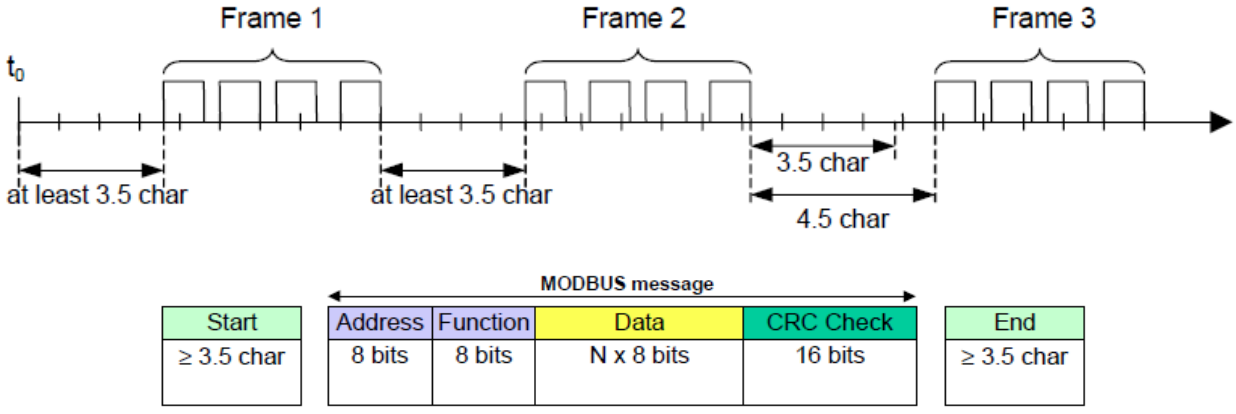
5.1 Modbus Physical Communication

HPx supports Modbus RTU over different serial port depending on control board configuration:

Physical Interface	Comments
RS485 Full-duplex (Standard Feature)	Default baudrate: 19200 baud, even parity, 1 stop bit
RS485 Half-duplex (Standard with different Setting)	Default baudrate: 19200 baud, even parity, 1 stop bit
UART (TTL level, Optional)	Default baudrate: 19200 baud, even parity, 1 stop bit
RS232 (Optional)	Default baudrate: 19200 baud, even parity, 1 stop bit

5.2 Modbus Message RTU Framing

In RTU mode, messages (request or response) frames are separated by a silent interval of at least 3.5 character times. Each character is 11 bits (1 start, 8 bits data, 1 bit parity and 1 stop). 2 stops bits is required if no parity is used.



Modbus RTU Message Frame

The entire message frame must be transmitted as a continuous stream of characters. If a silent interval of more than 1.5 characters times occurs between 2 characters, the frame is declared incomplete and should be discarded by the receiver.
Character timing will depend on baudrate selected up to 19200 baud. For baudrate greater than 19200 baud, it is recommended to use a value of at least 1.75ms for inter frame interval.
For example, at 19200 baud, 3.5 char is about 2ms for frame interval. At 115200 baud, a frame interval of 1.75ms is used.
The CRC field is appended as the last field of the message. The LSB is appended first followed by the MSB. The CRC calculation is started by first pre-loading 16 bits register to all 1s.

5.3 Modbus Function Code

HPx supports the following Modbus function code:

Function Description	Function Code
Read Holding Registers	0x03
Read Input Register	0x04
Write Single Register	0x06
Write Multiple Registers	0x10

HPx only supports access to one PMBus command at a time. In other word, reading the value of 2 successive commands/registers is not supported.

5.3.1 Modbus Function Code Read Holding Registers (0x03)

This function code is used to read the contents of a contiguous block of holding registers. The Request specifies the starting register address and the number of registers to read. A register is 2 bytes of data (MSB first) even if the command table in chapter 4 specifies only 1 byte (# bytes column). This function can be used to read all readable commands defined in chapter 4.

Request Frame:

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address)
Function Code	1 Byte	0x03
Starting Address MSB	1 Byte	0x00
Starting Address LSB	1 Byte	0x00 to 0xFF (see PMBUS Cmd Code, chap 4.0)
Quantity of Register MSB	1 Byte	0x00
Quantity of Register LSB	1 Byte	0x01 to 0x7B (number of words to read)
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Response Frame:

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address, even number)
Function Code	1 Byte	0x03
Byte count	1 Byte	2 * N (N = Quantity of Register in Request)
Register Value	N * 2 Bytes	Data
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Error Frame (in case of error):

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address)
Error Code	1 Byte	0x83
Exception Code	1 Byte	0x01, 0x02, 0x03 or 0x04
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Here is an example of a request to read output voltage. PMBus command (0x8B, READ_VOUT).

Request Frame:

HPx Address	Function Code	Starting Address MSB	Starting Address LSB	Qty of Register MSB	Qty of Register LSB	CRC LSB	CRC MSB
0xBE	0x03	0x00	0x8B	0x00	0x01	0xEE	0xEF

Response Frame:

HPx Address	Function Code	Byte Count	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x03	0x02	0x00	0x00	0xAD	0x9F

Value = 0x0000, if VOUT_MODE = 0x16, N=-10 (1/1024) -> 0x0000 / 1024 = 0v (READ_VOUT = 0v).

5.3.2 Modbus Function Code Read Input Registers (0x04)

This function code is used to read the content of a contiguous input registers. The Request specifies the starting register address and the number of registers to read. A register is 2 bytes of data (MSB first). This function can be used to read all readable commands defined in chapter 4.

Request Frame:

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address)
Function Code	1 Byte	0x04
Starting Address MSB	1 Byte	0x00
Starting Address LSB	1 Byte	0x00 to 0xFF (see PMBUS Cmd Code, chap 4.0)
Quantity of Register MSB	1 Byte	0x00
Quantity of Register LSB	1 Byte	0x01 to 0x7B (number of words to read)
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Response Frame:

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address, even number)
Function Code	1 Byte	0x04
Byte count	1 Byte	2 * N (N = Quantity of Register in Request)
Register Value	N * 2 Bytes	Data
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Error Frame (in case of error):

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address)
Error Code	1 Byte	0x84
Exception Code	1 Byte	0x01, 0x02, 0x03 or 0x04
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Here is an example of a request to read output voltage setting. PMBus command (0x21, COMMAND_VOUT).

Request Frame:

HPx Address	Function Code	Starting Address MSB	Starting Address LSB	Qty of Register MSB	Qty of Register LSB	CRC LSB	CRC MSB
0xBE	0x04	0x00	0x21	0x00	0x01	0x7B	0x0F

Response Frame:

HPx Address	Function Code	Byte Count	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x04	0x02	0x37	0x00	0xBA	0xDB

Value = 0x3700, VOUT_MODE = 0x16, N=-10 (1/1024) -> 0x3700 / 1024 = 13.75v (COMMAND_VOUT = 13.75v).

Here is another example of a request to read the manufacturer revision. PMBus command (0x9B, MFR_REVISION, size = 4 bytes).

Request Frame:

HPx Address	Function Code	Starting Address MSB	Starting Address LSB	Qty of Register MSB	Qty of Register LSB	CRC LSB	CRC MSB
0xBE	0x04	0x00	0x9B	0x00	0x02	0x1A	0xEB

Response Frame:

HPx Address	Function Code	Byte Count	Data	Data	Data	Data	CRC LSB	CRC MSB
0xBE	0x04	0x04	0x30	0x30	0x30	0x32	0x2F	0x95

Value = 0x30303032, in ascii -> 0002

5.3.3 Modbus Function Code Write Single Registers (0x06)

This function code is used to write a single register. It is used for all command defined in chapter 4 with a size of 2 or less. The Request specifies the address of the register to be written. A register is 2 bytes of data (MSB first). A normal response is the echo of the request.

Note: The write command is only executed if WRITE_PROTECT (0x10) register is set to 0x00 (Enable writes).

Request Frame:

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address)
Function Code	1 Byte	0x06
Register Address MSB	1 Byte	0x00
Register Address LSB	1 Byte	0x00 to 0xFF (see PMBUS Cmd Code, chap 4.0)
Register Value MSB	1 Byte	Value MSB
Register Value LSB	1 Byte	Value LSB
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Response Frame:

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address, even number)
Function Code	1 Byte	0x06
Register Address MSB	1 Byte	0x00
Register Address LSB	1 Byte	0x00 to 0xFF (see PMBUS Cmd Code, chap 4.0)
Register Value MSB	1 Byte	Written Value MSB
Register Value LSB	1 Byte	Written Value LSB
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Error Frame (in case of error):

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address)
Error Code	1 Byte	0x86
Exception Code	1 Byte	0x01, 0x02, 0x03 or 0x04
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Here is an example of a request to enable any write commands. PMBus command (0x10, WRITE_PROTECT).

Request Frame:

HPx Address	Function Code	Register Address MSB	Register Address LSB	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x06	0x00	0x10	0x00	0x00	0x92	0xC0

Set WRITE_PROTECT register to 0x00.

Response Frame:

HPx Address	Function Code	Register Address MSB	Register Address LSB	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x06	0x00	0x10	0x00	0x00	0x92	0xC0

Echo back the request if write successful.

Here is an example of a request to set output voltage. PMBus command (0x21, VOUT_COMMAND).

Request Frame:

HPx Address	Function Code	Register Address MSB	Register Address LSB	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x06	0x00	0x21	0x37	0x00	0xD5	0x3F

Set 13.75v to output with VOUT_MODE=0x16 (N=-10, 1/1024), (13.75 * 1024 = 0x3700)

Response Frame:

HPx Address	Function Code	Register Address MSB	Register Address LSB	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x06	0x00	0x21	0x37	0x00	0xD5	0x3F

Echo back the request if write successful.

Here is an example of a request to clear fault. PMBus command (0x03, CLEAR_FAULTS). This command has no data.

Request Frame:

HPx Address	Function Code	Register Address MSB	Register Address LSB	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x06	0x00	0x03	0x00	0x00	0x63	0x05

Response Frame:

HPx Address	Function Code	Register Address MSB	Register Address LSB	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x06	0x00	0x03	0x00	0x00	0x63	0x05

Echo back the request if write successful.

Here is an example of a request to turn ON the unit. PMBus command (0x01, OPERATION). This command has 1 byte of data.

Request Frame:

HPx Address	Function Code	Register Address MSB	Register Address LSB	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x06	0x00	0x01	0x00	0x80	0xC3	0x65

Response Frame:

HPx Address	Function Code	Register Address MSB	Register Address LSB	Register Value MSB	Register Value LSB	CRC LSB	CRC MSB
0xBE	0x06	0x00	0x01	0x00	0x80	0xC3	0x65

Echo back the request if write successful.

5.3.4 Modbus Function Code Write Multiple Registers (0x10)

This function code is used to write a block of contiguous registers. The Request specifies the starting register address and the number of registers to write. A register is 2 bytes of data (MSB first). This function is used for command (see chapter 4 table) with more than 2 bytes of data like USER_DATA_00, SERIAL_COMM_CONFIG …

Note: The write command is only executed if WRITE_PROTECT (0x10) register is set to 0x00 (Enable writes).

Request Frame:

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address)
Function Code	1 Byte	0x10
Starting Address MSB	1 Byte	0x00
Starting Address LSB	1 Byte	0x00 to 0xFF (see PMBUS Cmd Code, chap 4.0)
Quantity of Register MSB	1 Byte	0x00
Quantity of Register LSB	1 Byte	0x01 to 0x7B (number of words to write)
Byte Count	1 Byte	2 * N (N = Quantity of Register or Qty in byte size)
Registers Value	N * 2 Bytes	Value
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Response Frame:

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address, even number)
Function Code	1 Byte	0x10
Starting Address MSB	1 Byte	0x00
Starting Address LSB	1 Byte	0x00 to 0xFF (see PMBUS Cmd Code, chap 4.0)
Quantity of Register MSB	1 Byte	0x00
Quantity of Register LSB	1 Byte	0x01 to 0x7B (number of registers written)
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Error Frame (in case of error):

Slave Address	1 Byte	0xB0 to 0xBE (HPx Address)
Error Code	1 Byte	0x90
Exception Code	1 Byte	0x01, 0x02, 0x03 or 0x04
CRC LSB	1 Byte	
CRC MSB	1 Byte	

Here is an example of a request to write/set the serial configuration settings. PMBus command (0xD7, SERIAL_COMM_CONFIG, size = 8).

Request Frame:

HPx Address	Function Code	Starting Address MSB	Starting Address LSB	Qty of Register MSB	Qty of Register LSB	Byte Count	Data	Data	Data	Data	Data
0xBE	0x10	0x00	0xD7	0x00	0x04	0x08	0x80	0x25	0x00	0x00	0x00

Data	Data	CRC LSB	CRC MSB
0x02	0x00	0xA3	0x1D

Set 9600 baud (LSB 32 bits: 80250000), 1 stop bit (0x00), Even Parity (0x02), 8 Bits data (0x00)

Response Frame:

HPx Address	Function Code	Starting Address MSB	Starting Address LSB	Qty of Register MSB	Qty of Register LSB	CRC LSB	CRC MSB
0xBE	0x10	0x00	0xD7	0x00	0x04	0x6B	0x3D

5.4 Modbus Exception Responses

When a master sends a request to a power supply it expects a normal response. One of four possible events can occur from the master's query:

- If the slave device receives the request without a communication error, and can handle the query normally, it returns a normal response.
- If the slave does not receive the request due to a communication error, no response is returned. The master will eventually process a timeout condition for the request.
- If the slave receives the request, but detects a communication error (parity, CRC, ...), no response is returned. The master will eventually process a timeout condition for the request.
- If the slave receives the request without a communication error, but cannot handle it (for example, if the request is to read a non–existent output or register), the slave will return an exception response informing the client of the nature of the error.

The exception response message has two fields that differentiate it from a normal response:

- **Function Code Field:** In a normal response, the slave echoes the function code of the original request in the function code field of the response. All function codes have a most–significant bit (MSB) of 0 (their values are all below 80 hexadecimal). In an exception response, the slave sets the MSB (bits 7) of the function code to 1. This makes the function code value in an exception response exactly 80 hexadecimal higher than the value would be for a normal response. With the function code's MSB set, the master can recognize the exception response and can examine the data field for the exception code.
- **Data Field:** In a normal response, the slave may return data or statistics in the data field (any information that was requested in the request). In an exception response, the slave returns an exception code in the data field. This defines the slave condition that caused the exception.

MODBUS Exception Codes		
Code	**Name**	**Meaning**
0x01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the slave. This may be because the function code is only applicable to newer devices and was not implemented in the unit selected. It could also indicate that the slave is in the wrong state to process a request of this type, for

		example because it is unconfigured and is being asked to return register values.
0x02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the slave. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, the PDU addresses the first register as 0, and the last one as 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 4, then this request will successfully operate (address-wise at least) on registers 96, 97, 98, 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 5, then this request will fail with Exception Code 0x02 “Illegal Data Address” since it attempts to operate on registers 96, 97, 98, 99 and 100, and there is no register with address 100.
0x03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for slave. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
0x04	SERVER DEVICE FAILURE	An unrecoverable error occurred while the slave was attempting to perform the requested action.

6 Physical Serial Communication

At the physical level, HPx may use different serial physical interface (UART, RS232, RS485 half/full duplex). HPx will always behave as a slave device. Slave nodes will not transmit data without a request from the master node and do not communicate with other slaves. UART (TTL) and RS232 serial interface may be used as an interface when only a short distance communication is required.
Each HPx device may be connected either directly on the trunk cable forming a daisy-chain or on a passive Tap with a derivation cable.
The default serial configuration is: 19200 baud, 1 Stop bit, Even Parity and 8 bits data

6.1 Serial UART, RS232 Connection

HPx does not support flow control (RTS/CTS). It is using only 3 wires for RS232 or UART connection (RX, TX and SGND).

In this serial configuration:

- All HPx RX signal (pin 13) need to be connected together. This signal is than connected to TX of the master node.
- All HPx TX signal (pin 15) need to be connected together. This signal is than connected to RX of the master node.

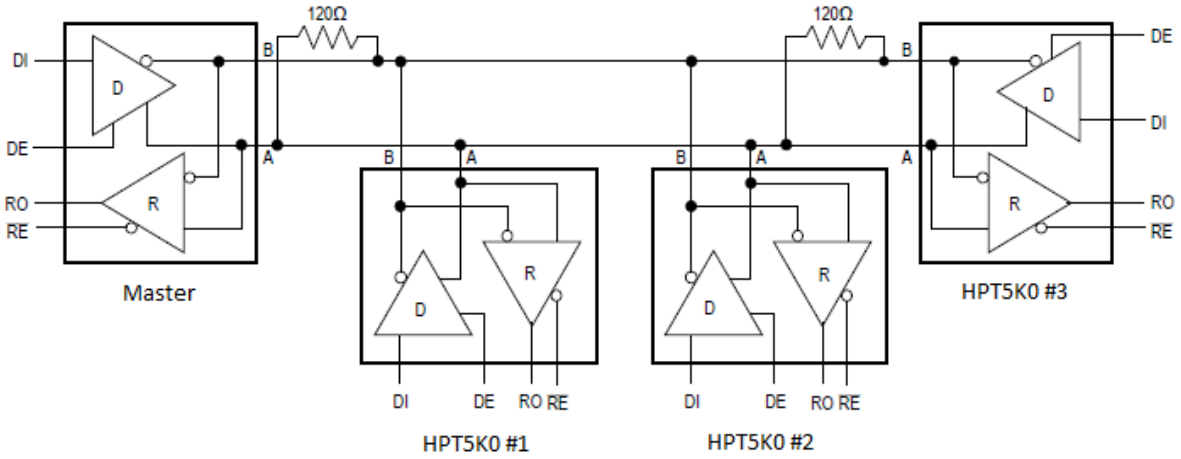
6.2 Serial RS485 Half Duplex Connection

HPx supports 2 wires (twisted pair) RS485 connections (half duplex) with ground signal.

In half duplex mode:

- All A(+, Non-Inverting input/output) are connected together
- All B(- or Inverting input/output) are connected together
- All Ground are connected together

Ideally, two ends of the cable will have a termination resistor connected across the two twisted wires. HPx does not have any termination resistor.

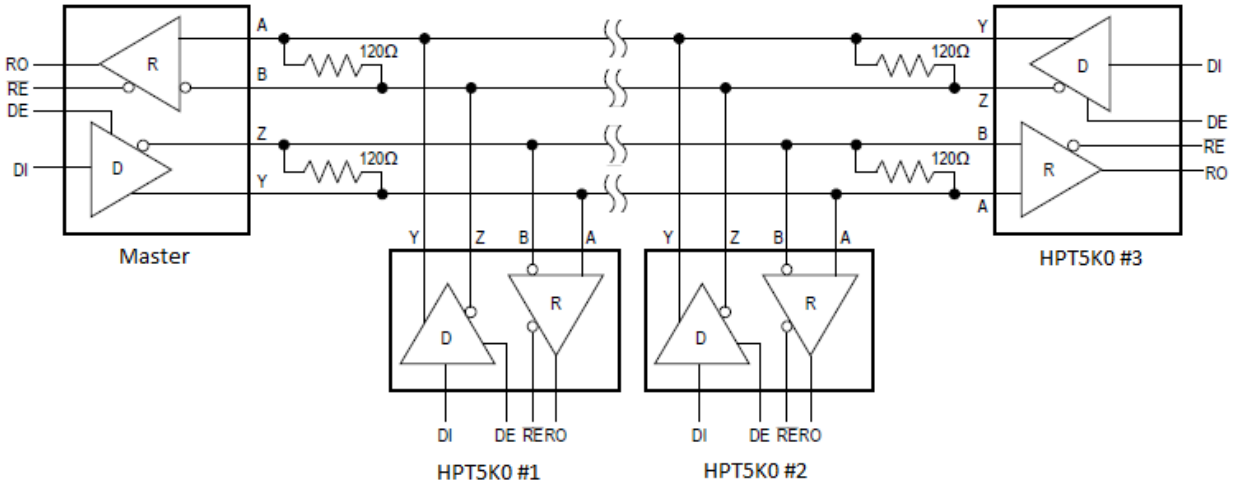


HPx signal connector (Half Duplex)	
A	RS485_Y (pin 9), Noninverting Driver Output and Noninverting Receiver Input
B	RS485_Z (pin 11), Inverting Driver Output and Inverting Receiver Input
GND	SGND (pin 12)

6.3 Serial RS485 Full Duplex Connection

HPx supports 4 wires (2 twisted pair) RS485 connections (full duplex) with ground signal. The data sent by the master are only received by the slaves. Data sent by a slave are only received by the master.

Ideally, both ends of both twisted cables will have a termination resistor connected across the twisted pair wires. HPx does not have any termination resistor.



HPx signal connector (Full Duplex)	
A	RS485_A (pin 13), Noninverting Receiver Input
B	RS485_B (pin 15), Inverting Receiver Input
Y	RS485_Y (pin 9), Noninverting Driver Output
Z	RS485_Z (pin 11), Inverting Driver Output
GND	SGND (pin 12)

7 CANopen protocol

A CAN master (client) is a controller that makes requests to nodes (server or HPx) to respond to its commands. A CAN slave (server or HPx) responds to the commands issued by the CAN master (client). The CAN protocol permits both single-master and multiple-master networks.
The data-byte units transported through a CAN network are called communication objects (COBs). HPx supports only Service Data Object (SDO). Objects are based on PMBus command specified in chapter 4.

Please, refer to "CiA 301 CANopen application layer and communication profile" for more protocol details.

7.1 CANBus Physical Communication

The network topology is a twisted two wire bus line (CANH and CANL) with common return (SGND) being terminated at both ends by resistors (120 Ω) representing the characteristic impedance of the bus line. All devices in a CANBus network shall use the very same bit rate.

HPx support the following bit timing:

Bit Rate	Estimated Bus Length
1 Mbits/s	25 m
800 Kbits/s	50 m
500 Kbits/s	100 m
250 Kbits/s	250 m
125 Kbits/s	500 m
50 Kbits/s	1000 m
20 Kbits/s	2500 m
10 Kbits/s	5000 m

HPx signal connector (CAN Bus)	
Pin 8	CANH bus line, CAN High (dominant high)
Pin 10	CANL bus line, CAN Low (dominant low)
GND	SGND (pin 12)

There is no termination resistor inside the power supply. User shall set termination resistors accordingly depending on the bus layout.

7.2 Frame Format

HPx is only using standard SDO data frame format with 11 bits identifier.

Identifier	Control	8 Data Bytes							
11 bits ID	DLC = 8	Cmd	Index LSB	Index MSB	SubIndex	Object Data			
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

7.3 Identifier Setup

HPx uses SDO transmit COB type (0x0580-0x05FF) and SDO receive COB Type (0x0600-0x067F). The Node ID (address) of a CAN device must be within 1 to 127. In this case, up to 127 servers (HPx) can be addressed. The Communication Object Identifier (COB-ID) is directly dependent on the selected node ID.

- 0x0600 + Node ID for receiving (COB-ID that the server receives on)
- 0x0580 + Node ID for transmitting (COB-ID that the server responds with)

The Node ID is based on the following table:

Device	Address	Address Bits (MSB to LSB)							
HPx	0x5x	0	1	0	1	1	A2	A1	A0
Global Broadcast	0x00	0	0	0	0	0	0	0	0

Note: Signals A2, A1, A0 in the output signal connector are used to differentiate up to 8 power supplies on the same bus.

The default Node ID address (with A2, A1, A0 not connected, all '1') is 0x5F. In this case, COB-ID = 0x065F (0x0600 + 0x005F) for receiving and COB-ID = 0x5DF (0x0580 + 0x005F) for transmitting with respect to the slave device or server (HPx).

7.4 Object Dictionary

An object dictionary (OD) is a naming system that gives a unique identifier to each data item or object that is communicated over the CAN bus. An object is identified by an index and, if it is a complex object, also by a sub-index.
Each object of the HPx is addressed using a 16 bit index. HPx only supports the Manufacturer Specific Profile area from 0x2000 to 0x20FF. The LSB of this index represents the PMBus command code defined in chapter 4.

7.5 Service Data Object (SDO)

Basically, an SDO is transferred as a sequence of segments. Prior to transferring the segments there is an initialization phase where client and server prepare themselves for transferring the segments. For SDOs, it is also possible to transfer a data set of up to four bytes during the initialization phase. This mechanism is called SDO expedited transfer.
Always the client (master) initiates an SDO transfer for any type of transfer. The owner of the accessed object dictionary is the server (HPx) of the SDO. Either the client or the server may take the initiative to abort the transfer of an SDO.

HPx support the following services:

- SDO download (transfer data from the client (master) to the server (HPx)), which is subdivided into:
  - SDO download initiate
  - SDO download segment
- SDO upload (transfer data from the server (HPx) to the client (master)), which is subdivided into:
  - SDO upload initiate
  - SDO upload segment
- SDO abort transfer

SDO expedited transfer is used for transmission of up to 4 data bytes. It consists of one SDO request and one response. SDO segmented transfer is used for data objects larger than 4 Bytes.

HPx does not support SDO block transfer.

7.5.1 Service SDO download initiate (HPx write)

The client requests the server to prepare downloading of data by using the SDO download initiate service. WRITE_PROTECT (0x10) must be set to 0x00 to enable write to any writable register.

Client request

<table>
<tr><th>Byte #</th><th>Description</th></tr>
<tr><td>0</td><td>SDO command specifier. 8 bits: “0010nnes” (nn: only valid if e=s=1, number of bytes in d that do not contain data; e=1 for expedited transfer; s=1 if data size is indicated)</td></tr>
<tr><td>1..2</td><td>Object index (LSB)</td></tr>
<tr><td>3</td><td>Object subIndex</td></tr>
<tr><td>4..7
d</td><td>d : Expedited data or data size if segmented transfer:
<table>
<tr><th>e</th><th>s</th><th>Description for d (4..7)</th></tr>
<tr><td>0</td><td>0</td><td>d is reserved for further use.</td></tr>
<tr><td>0</td><td>1</td><td>d contains the number of bytes to be downloaded. Byte 4 contains the LSB and byte 7 contains the MSB</td></tr>
<tr><td>1</td><td>1</td><td>d contains the data of length 4-n to be downloaded, the encoding depends on the type of the data referenced by index and subIndex</td></tr>
<tr><td>1</td><td>0</td><td>d contains unspecified number of bytes to be downloaded</td></tr>
</table></td></tr>
</table>

Server response

Byte #	Description

0	SDO command specifier. 8 bits: “01100000” (0x60)
1..2	Object index (LSB)
3	Object subIndex
4..7	Reserved

Example: an SDO client requests the HPx at address 0xBE to write object identified by index 0x2010 (0x2000 + 0x10 for WRITE_PROTECT register) with 0x00 to enable writes.

Master (or Client) command packet:

Identifier	Control	Cmd	Index		SubIndex	Data			
0x65F	0x08	0x2F	0x10	0x20	0x00	0x00	0x00	0x00	0x00

Expedited transfer, data of length=1, data to write=0x00 -> Enable Write.
Cmd=0x2F: nn=1 (size=4-nn=1 bytes of data), es=11, expedited transfer, d contains the data.

HPx (or Server) response packet:

0x5DF	0x08	0x60	0x10	0x20	0x00	0x00	0x00	0x00	0x00

Example: an SDO client requests the HPx at address 0xBE to write object identified by index 0x2021 (0x2000 + 0x21 for VOUT_COMMAND register) with 0x3200 (set Vout=12.5v, if VOUT_MODE=0x16, 1/1024v, 12.5v*1024=12800=0x3200)

Master (or Client) command packet:

Identifier	Control	Cmd	Index		SubIndex	Data			
0x65F	0x08	0x2B	0x21	0x20	0x00	0x00	0x32	0x00	0x00

Expedited transfer, data of length=2, data to write=0x3200 -> if 1/1024v (see VOUT_MODE), VOUT_COMMAND=12.5v.
Cmd=0x2B: nn=2 (size=4-nn=2 bytes of data), es=11, expedited transfer, d contains the data.

HPx (or Server) response packet:

0x5DF	0x08	0x60	0x21	0x20	0x00	0x00	0x00	0x00	0x00

7.5.2 Service SDO download segment (HPx write)

The client transfers the segmented data to the server by using the SDO download service. The continue parameter indicates the server whether there are still more segments to be downloaded or that this was the last segment to be downloaded.

Client request

Byte #	Description
0	SDO command specifier. 8 bits: “000tnnnc” (t: toggle bit set to 0 in first segment; nnn: number of bytes in d that do not contain data; c=1 if this is the last segment; c=0 more segments to download)
1..7	d : Data segment

Server response

Byte #	Description
0	SDO command specifier. 8 bits: “001t0000” (t: toggle bit set to 0 in first segment)
1..7	Reserved

7.5.3 Service SDO upload initiate (HPx read)

The client is using the service SDO upload for transferring the data from the server to the client. The client requests the server to prepare the data for uploading by using the SDO upload initiate service.

Client request

Byte #	Description
0	SDO command specifier. 8 bits: “01000000” (0x40)
1..2	Object index (LSB)
3	Object subIndex
4..7	Reserved

Server response

Byte #	Description
0	SDO command specifier. 8 bits: “0100nnes” (nn: only valid if e=s=1, number of bytes in d that do not contain data; e=1 for expedited transfer; s=1 if data size is indicated)
1..2	Object index (LSB)
3	Object subIndex
4..7	d : Expedited data or data size if segmented transfer

e	s	Description for d (4..7)
0	0	d is reserved for further use.
0	1	d contains the number of bytes to be uploaded. Byte 4 contains the LSB and byte 7 contains the MSB
1	1	d contains the data of length 4-n to be uploaded, the encoding depends on the type of the data referenced by index and subIndex
1	0	d contains unspecified number of bytes to be uploaded

Example: an SDO client requests the HPx at address 0xBE to read object identified by index 0x2021 (0x2000 + 0x21 for VOUT_COMMAND register).

Master (or Client) command packet:

Identifier	Control	Cmd	Index		SubIndex	Data			
0x65F	0x08	0x40	0x21	0x20	0x00	0x00	0x00	0x00	0x00

HPx (or Server) response packet:

0x5DF	0x08	0x4B	0x21	0x20	0x00	0x00	0x32	0x00	0x00

Cmd=0x4B: nn=2 (size=4-nn=2 bytes of data returned), es=11, expedited transfer, d contains the data.
data=0x3200 -> if 1/1024v (see VOUT_MODE), VOUT_COMMAND=12.5v.

7.5.4 Service SDO upload segment (HPx read)

The client requests the server to supply the data of the next segment by using the SDO upload segment service. The continue parameter indicates the client whether there are still more segments to be uploaded or that this was the last segment to be uploaded.

Client request

Byte #	Description
0	SDO command specifier. 8 bits: “011t0000” (t: toggle bit set to 0 in first segment)
1..7	Reserved

Server response

Byte #	Description
0	SDO command specifier. 8 bits: “000tnnnc” (t: toggle bit set to 0 in first segment; nnn: number of bytes in d that do not contain data; c=1 if this is the last segment; c=0 more segments to upload)
1..7	d: Data segment

7.5.5 Service SDO abort transfer

The SDO abort transfer service aborts the SDO upload service or SDO download service of an SDO referenced by its number. The reason is indicated.

Abort transfer (from client or server)

Byte #	Description
0	SDO command specifier. 8 bits: “10000000” (0x80)
1..2	Object index (LSB)
3	Object subIndex
4..7	SDO abort code (LSB)

Description	Abort Code
CO_SDO_AB_NONE	0x00000000UL
CO_SDO_AB_TOGGLE_BIT	0x05030000UL
CO_SDO_AB_TIMEOUT	0x05040000UL
CO_SDO_AB_CMD	0x05040001UL

CO_SDO_AB_BLOCK_SIZE	0x05040002UL
CO_SDO_AB_SEQ_NUM	0x05040003UL
CO_SDO_AB_CRC	0x05040004UL
CO_SDO_AB_OUT_OF_MEM	0x05040005UL
CO_SDO_AB_UNSUPPORTED_ACCESS	0x06010000UL
CO_SDO_AB_WRITEONLY	0x06010001UL
CO_SDO_AB_READONLY	0x06010002UL
CO_SDO_AB_NOT_EXIST	0x06020000UL
CO_SDO_AB_NO_MAP	0x06040041UL
CO_SDO_AB_MAP_LEN	0x06040042UL
CO_SDO_AB_PRAM_INCOMPAT	0x06040043UL
CO_SDO_AB_DEVICE_INCOMPAT	0x06040047UL
CO_SDO_AB_HW	0x06060000UL
CO_SDO_AB_TYPE_MISMATCH	0x06070010UL
CO_SDO_AB_DATA_LONG	0x06070012UL
CO_SDO_AB_DATA_SHORT	0x06070013UL
CO_SDO_AB_SUB_UNKNOWN	0x06090011UL
CO_SDO_AB_INVALID_VALUE	0x06090030UL
CO_SDO_AB_VALUE_HIGH	0x06090031UL
CO_SDO_AB_VALUE_LOW	0x06090032UL
CO_SDO_AB_MAX_LESS_MIN	0x06090036UL
CO_SDO_AB_NO_RESOURCE	0x060A0023UL
CO_SDO_AB_GENERAL	0x08000000UL
CO_SDO_AB_DATA_TRANSF	0x08000020UL
CO_SDO_AB_DATA_LOC_CTRL	0x08000021UL
CO_SDO_AB_DATA_DEV_STATE	0x08000022UL
CO_SDO_AB_DATA_OD	0x08000023UL
CO_SDO_AB_NO_DATA	0x08000024UL

8 SCPI Protocol

HPx supports SCPI (Standard Commands for Programmable Instruments) over serial bus. SCPI protocol can be selected instead of Modbus using PMBus command (0xDE, HARDWARE_CONFIG)

8.1 SCPI Physical Communication

HPx supports SCPI over different serial port depending on control board configuration:

Physical Interface	Comments
RS485 Full-duplex (Standard Feature)	Default baudrate: 19200 baud, even parity, 1 stop bit
RS485 Half-duplex (Standard with different Setting)	Default baudrate: 19200 baud, even parity, 1 stop bit
UART (TTL level)	Default baudrate: 19200 baud, even parity, 1 stop bit
RS232	Default baudrate: 19200 baud, even parity, 1 stop bit

8.2 SCPI Programming Commands

The CR LF is defined as the mandatory message terminator for serial interfaces.

The maximum string size is limited to 128 characters total including message terminators. Up to 10 commands (separated by ';') can be sent into a single character string.

If multiple HPx are connected to the same physical bus, see ":INSTrument:NSELect" or ":INSTrument:SELect <value>" below.

Note:

Expression enclosed in [ ] are optional.
Expression enclosed in < > are programming values. They can be expressed in hexadecimal (#H1234), decimal (100) or floating point (10.5).
A vertical stroke '|' in parameter definitions indicates alternative possibilities in the sense of OR

Standard Commands:

Command	Description
*CLS	(ESR) Clears the Standard Event Status Register and Clears the Events Queue
*ESE <NR1>	(ESE) Sets the Standard Event Status Enable Register
*ESE?	(ESE) Queries the Standard Event Status Enable Register
*ESR?	(ESR) Queries and Clears the Standard Event Status Register
*IDN?	Return Device ID as "<Manufacturer>, <ModelNumber>, <SerialNumber>, <FirmwareRev>"
*OPC	Start Operation Complete Active Mode
*OPC?	Has Operation completed? <0/1>
*RCL	Restore to the default user setting (E type registers saved by STORE_USER_ALL or *SAV)
*RST	Reset the Device
*SAV	Store all (E type) user setting registers into EEPROM, saved settings will be used after power cycle
*SRE <NR1>	(SRER) Sets the Service Request Enable Register
*SRE?	(SRER) Queries the Service Request Enable Register
*STB?	(SBR) Queries the Status Byte Register. Bit 6 is reset to 0, the other bits kept unchanged.
*TST?	Test the Device and Return the Result <0/1>
*WAI	Wait Until the Last Operation is Completed

Power Supply supported Commands:

Command	Description
:CURRent {<value>|MAX|MIN|DEF} :CURRent:AMPLitude {<value>|MAX|MIN|DEF} :CURRent:PROTection {<value>|MAX|MIN|DEF}	Set output current limit :Current 14.5 (set current limit to 14.5A) :Current MAX (set to maximum current limit) :Current def (set to default current limit)
:CURRent:AMPLitude?	Read output current limit setting

<table>
<tr><td>:CURRent:PROTection?
:CURRent?</td><td>:CURRent:AMPLitude?
14.5 (response in A)</td></tr>
<tr><td>:INSTrument:NSELect <nn></td><td>Select a specific power supply when multiple power supplies are connected to the same bus. Set <nn> to 0 for broadcast (default). <nn> is from 0 to 127. <nn> is the Slave Address divided by 2. At power up, no specific power supply is selected (broadcast mode, <nn> = 0). They will all execute or respond to command at the same time. Response to a query can be corrupted if multiple devices are connected and try to respond at the same time. (see chapter 3.1 for slave address)
:INSTrument:NSELect 2 (select power supply address 0x04)</td></tr>
<tr><td>:INSTrument:NSELect?</td><td>Return the current selected power supply. Return a value between 0 and 127.
See <nn> in :INSTrument:NSELect <nn></td></tr>
<tr><td>:INSTrument:SELect <value></td><td>Select a specific power supply when multiple power supplies are connected to the same bus. Set <value> to 0 for broadcast (default). <value> depends on A2-A0 on output signal connector and defines the power supply slave address (see chapter 3.1). At power up, no specific power supply is selected (broadcast mode, <value> = 0). They will all execute or respond to command at the same time. Response to a query can be corrupted if multiple devices are connected and try to respond at the same time.
If A2-A0 = 000 -> <value> = #hB0
If A2-A0 = 001 -> <value> = #hB2
- -
If A2-A0 = 010 -> <value> = #hB4
If A2-A0 = 111 -> <value> = #hBE
:INSTrument:SELect #hBE (select power supply address 0xBE)</td></tr>
<tr><td>:INSTrument:SELect?</td><td>Return the current selected power supply address. Between #hB0 and #hBE
:INSTrument:SELect?
190 (response, 0xBE)</td></tr>
<tr><td>:MEASure:CURRent?</td><td>Measure main output current
:MEASure:CURRent?
5.5 (Response in A)</td></tr>
<tr><td>:MEASure:POWer?</td><td>Measure main output power
:MEASure:POWer?
1050 (Response in Watts)</td></tr>
<tr><td>:MEASure:TEMPerature?</td><td>Measure hottest temperature
:MEASure: TEMPerature?
88 (Response in Celsius)</td></tr>
<tr><td>:MEASure:VOLTage?</td><td>Measure main output voltage
:MEASure: VOLTage?
100.4 (Response in Volt)</td></tr>
<tr><td>:OUTPut:STATe {ON|OFF|1|0}</td><td>Turn ON or OFF main output
:OUTPut:STATe ON (turn ON output)
:OUTP:STAT 1 (turn ON output)
:OUTPut:STAT OFF (turn OFF output)
:OUTP:STAT 0 (turn OFF output)</td></tr>
<tr><td>:OUTPut:STATe?</td><td>Return power supply output enable state
:OUTPut:STATe?
0 (output if OFF state)</td></tr>
<tr><td>:PMBUs <opcode>[,<size>][,<value>]</td><td>Specific command to write PMBus related command in SCPI format.
<opcode> is PMBus Cmd Code defined in Chapter 4
<size> if there, <value> represente a HEX string of byte (LSB if size=2)
<value> is the data formatted according to Chapter 4 (linear, string,…) depending on the command.
:PMBUs 33, 2560 (cmd 0x21, set vout 10v = 2560, if VOUT_MODE=0x18, 1/256v, this format is only used for data size of 2 bytes or less)
:pmbus 33,2,#h8034 (<value> = 0x3480, set vout = 52.5v, LSB)
:pmbus 3 (send a clear fault command)</td></tr>
<tr><td>:PMBUs? <opcode></td><td>Specific command to read PMBus related command in SCPI format.
<opcode> is PMBus Cmd Code defined in Chapter 4.
Response is always a Hexadecimal string (#Hxxxxxx). For 2 or less byte, LSB is returned first. Response data is formatted according to Chapter 4 table.
:pmbus? #h21 (read VOUT_COMMAND register)
#H8034 (response: 0x3480, if 1/256v, 0x3480 / 256 = 52.5v)</td></tr>
<tr><td>:STATus:OPERation:CONDition?</td><td>Read the Operational Condition register (16 bits)
<table>
<tr><th>Bit#</th><th>Bit Name</th><th>Description</th></tr>
<tr><td>0</td><td>in_current_limit</td><td>In current limit</td></tr>
<tr><td>1</td><td>in_power_limit</td><td>In power limit</td></tr>
<tr><td>2</td><td>ac_high_line</td><td>Input High Line</td></tr>
<tr><td>3</td><td>pwm_enable</td><td>Converter PWM is running</td></tr>
<tr><td>4</td><td>ot_warning</td><td>Temperature warning</td></tr>
<tr><td>5</td><td>ot_fault</td><td>Temperature fault</td></tr>
<tr><td>6</td><td>temp_too_cold</td><td>Temperature is very cold</td></tr>
<tr><td>7</td><td>fan_fault</td><td>Fan fault</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>8</td><td>temp_cold</td><td>Temperature is cold</td></tr>
<tr><td>9</td><td>under_voltage</td><td>Output under voltage condition</td></tr>
<tr><td>10</td><td>analog_prog</td><td>Analog programming enable</td></tr>
<tr><td>11</td><td>vstby_enable</td><td>5v output standby enable</td></tr>
<tr><td>12</td><td>current_walkin</td><td>Soft start in current mode</td></tr>
<tr><td>13</td><td>hw_ovp_test_mode</td><td>Hardware OVP test mode</td></tr>
<tr><td>14</td><td>sec_calibration_mode</td><td>Secondary calibration mode</td></tr>
<tr><td>15</td><td>pri_calibration_mode</td><td>Primary calibration mode</td></tr>
</table>
</td></tr>
<tr><td>:STATus:OPERation:ENABle <value></td><td>Sets the Operational Condition Enable register</td></tr>
<tr><td>:STATus:OPERation:ENABle?</td><td>Read the Operational Condition Enable register</td></tr>
<tr><td>:STATus:OPERation[:EVENt]?</td><td>Read the Operational Condition Event register</td></tr>
<tr><td>:STATus:PRESet</td><td>Preset all Operation and Questionable Enable registers</td></tr>
<tr><td>:STATus:QUEStionable:CONDition?</td><td>Read the Questionable Condition register (16 bits). Indicate the shutdown condition of the unit.
<table>
<tr><th>Bit#</th><th>Bit Name</th><th>Description</th></tr>
<tr><td>0</td><td>AC_BAD</td><td>Input AC Fault</td></tr>
<tr><td>1</td><td>PFC_BAD</td><td>PFC Fault</td></tr>
<tr><td>2</td><td>AC_IMB</td><td>AC Input Imbalance</td></tr>
<tr><td>3</td><td>PRI_OV</td><td>Primary OVP</td></tr>
<tr><td>4</td><td>HW_OV</td><td>Hardware OVP</td></tr>
<tr><td>5</td><td>SW_OV</td><td>Software OVP</td></tr>
<tr><td>6</td><td>OCP</td><td>Over Current</td></tr>
<tr><td>7</td><td>UV</td><td>Output Under Voltage</td></tr>
<tr><td>8</td><td>PFC_OTP</td><td>Primary Over Temperature</td></tr>
<tr><td>9</td><td>SEC_OTP</td><td>Secondary Over Temperature</td></tr>
<tr><td>10</td><td>FAN_FAULT</td><td>Fan Failure</td></tr>
<tr><td>11</td><td>PRI_COMM_ERR</td><td>Primary Communication Timeout</td></tr>
<tr><td>12</td><td>OUTPUT_FAULT</td><td>Output is turn Off due to fault</td></tr>
<tr><td>13</td><td>SW_OFF</td><td>Output Off state (Pmbus command)</td></tr>
<tr><td>14</td><td>REMOTE_OFF</td><td>Remote Off</td></tr>
<tr><td>15</td><td>RFU</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>:STATus:QUEStionable:ENABle <value></td><td>Set the Questionable Condition Enable register</td></tr>
<tr><td>:STATus:QUEStionable:ENABle?</td><td>Read the Questionable Condition Enable register</td></tr>
<tr><td>:STATus:QUEStionable[:EVENt]?</td><td>Read the Questionable Condition Event register</td></tr>
<tr><td>:SYSTem:CAPability?</td><td>Return device class capability. (DCPSUPPLY)
:SYSTem:CAPability?
DCPSUPPLY (Response)</td></tr>
<tr><td>:SYSTem:ERRor?</td><td>The oldest error message is removed from the Error Queue and returned. If the Error Queue is empty, a 0 is returned.
:SYSTem:ERRor?
-222, “Data out of range” (Response if error)</td></tr>
<tr><td>:SYSTem:VERSion?</td><td>Read the SCPI Compliance year that this interface adheres to.
:SYSTem:VERSion?
1999.0 (Response)</td></tr>
<tr><td>:VOLTage {<value>|MAX|MIN|DEF}
:VOLTage:AMPLitude {<value>|MAX|MIN|DEF}</td><td>Set main output voltage
:VOLTage 64 (set output voltage to 64v)</td></tr>
<tr><td>:VOLTage?
:VOLTage:AMPLitude?</td><td>Read output voltage setting
: VOLTage:AMPLitude?
64 (response in V)</td></tr>
<tr><td>:VOLTage:LIMit:LOW {<value>|MAX|MIN|DEF}</td><td>Set Under Voltage Limit
:VOLTage:LIMit:LOW 25 (set under voltage detection to 25v)</td></tr>
<tr><td>:VOLTage:LIMit:LOW?</td><td>Read the Under Voltage Limit setting
:VOLTage:LIMit:LOW?
25 (Response in V)</td></tr>
<tr><td>:VOLTage:PROTection:LEVel {<value>|MAX|MIN|DEF}</td><td>Set Over Voltage Protection level
:VOLTage:PROTection:LEVel 105 (set OVP to 105v)</td></tr>
<tr><td>:VOLTage:PROTection:LEVel?</td><td>Read the Over Voltage Protection setting
:VOLTage:PROTection:LEVel?
105 (Response in V)</td></tr>
</table>

9 LED and Signals

Alarm signal and LED State at different condition:

	LED State			Signals			
Condition	AC OK	DC OK		ACOK	DCOK	FAN_FAIL \ TEMP	Remote Inhibit
AC input OK	ON	ON(4)		LOW	LOW(4)	LOW	LOW
AC not present or too low	OFF	OFF		HIGH	HIGH	LOW	X(3)
AC Present but out of range or PFC failure or no Primary to secondary communication	Blink (0.2s ON, 0.2s OFF)	OFF		HIGH	HIGH	LOW	X(3)
Output Over Voltage	ON	OFF		LOW	HIGH	LOW	LOW
Output Under Voltage	ON	OFF		LOW	HIGH(4)	LOW	LOW
Current Limit (Constant current response)	ON	Blink (0.2s ON, 0.2s OFF)		LOW	LOW or HIGH(4)	LOW	LOW
Fan Failure / Thermal Shutdown	ON	OFF		LOW	HIGH	HIGH(1)	LOW
Remote OFF	ON	Blink (1s ON, 1s OFF)		LOW	HIGH	LOW	HIGH
PMBus Operation OFF	ON	Blink (1s ON, 1s OFF)		LOW	HIGH	LOW	LOW

(1) In case of fan failure, FAN_FAIL signal will be set 10s before output shutdown.
(2) In case of over temperature, TEMP signal will be set 10s before output shutdown.
(3) Don't care.
(4) DC_OK LED is ON if Output Voltage >= VOUT_UV_FAULT_LIMIT, if Output Voltage < VOUT_UV_FAULT_LIMIT, the DC_OK LED will be OFF