CAN / RS485

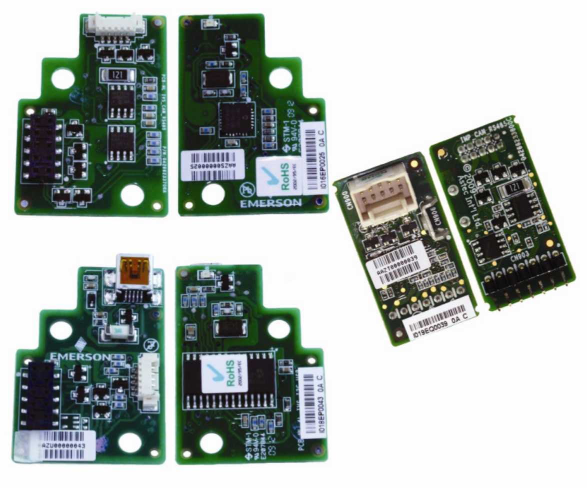
Interface Adapter

For iMP or iVS Series

Total Power:	< 1 Watts
Input Voltage:	5 Vdc Internal
Outputs:	CAN, RS485, USB, I²C

Special Features

- Input Protocols:
  1) RS485 using Modbus
  2) CAN using modified Modbus
- Output Protocol:
  I²C with SMBus Support



Product Description

The new iMP Extension adapter enables both USB and Controller Area Network (CAN) or RS485 Bus connectivity, providing a complete interface between the iMP device and the I²C bus with a simple command set which enables the highest levels of configuration flexibility.

The CAN/RS485 to I²C interface adapter modules connect with CAN Bus architectures through CaseRx/CaseTx interfaces found on the iMP case. The modules communicate with on-board I²C Bus via Modbus Protocol for RS485 Bus and modified Modbus for CANbus.

The new adapters now enable the iMP to be used in a host of new ruggedized applications including automotive networks, industrial networks, medical equipment and building automation systems.

The RS485/CAN-to-I²C uses 2 Input Protocols and 1 Output Protocol. The Input Protocols used are: RS485 using Modbus (Command Index: 0x01), and CAN using modified Modus (Command Index: 0x02). The Output Protocol use is: I²C with SMBus support (Command Index: 0x80).

ARTESYN™
EMBEDDED TECHNOLOGIES

Table of Contents

Table of Contents

1. Model Number

Table 1. Model Numbers

Model No.	Model Supported	Interface Bus	Description
73-546-001	iVS Series	USB	USB to I²C Module for iVS Case
73-544-001	iVS Series	RS485	RS485 to I²C Module for iVS Case
73-544-002	iMP Series	RS485	RS485 to I²C Module for iMP Case
73-544-003	iMP Series	CAN	CAN to I²C Module for iMP Case
73-544-004	iVS Series	CAN	CAN to I²C Module for iMP Case

Options

None

2. General Description

73-546-001

This module is for iVS USB to I²C adapter module that connects to a standard USB port (see Figure 4) found on most compatible for any IBM PCs and provides bi-directional communication with I²C devices using the I²C protocol. The adapter is powered directly from the iVS case internal power supply 5VSB and PC's USB port. The on-board LED illuminates after the USB host has successfully enumerated it.

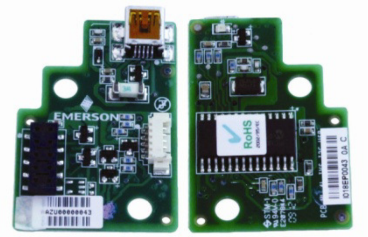


Figure 1. iVS USB to I²C Adapter

73-544-001 / 73-544-004

These modules are for iVS RS485/CAN to I²C adapter module that connects to CAN bus through CaseRx/CaseTx found on iVS cases output connector (see Figure 5) and communicates to I²C Bus using Modbus Protocol for RS485 Bus and Modified Modbus for CANbus. The adapter is powered directly from the iVS case internal power supply 5VSB. The on-board LED illuminates after the CAN/RS485 host has successfully enumerated it.

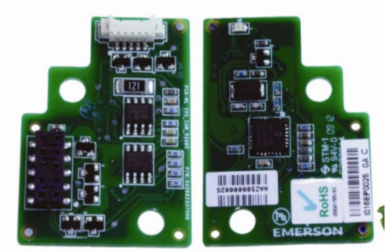


Figure 2. iVS CAN/RS485 to I²C Adapter

73-544-002 / 73-544-003

These modules are for iMP RS485/CAN to I²C adapter that connects to CAN bus through CaseRx/CaseTx found on iMP cases output connector (see Figure 6) and communicates to I²C Bus using Modbus Protocol for RS485 Bus and Modified Modbus for CAN bus. The adapter is powered directly from the iMP case internal power supply 5VSB.

Figure 3. iMP CAN/RS485 to I²C Adapter

3. Electrical Specifications

Absolute Maximum Ratings

Stress in excess of those listed in the “Absolute Maximum Ratings” may cause permanent damage to the power supply. These are stress ratings only and functional operation of the unit is not implied at these or any other conditions above those given in the operational sections of this TRN. Exposure to any absolute maximum rated condition for extended periods may adversely affect the power supply’s reliability.

Table 2. Absolute Maximum Ratings

Parameter	Model	Symbol	Min	Typ	Max	Unit
Input Voltage Internal Supply +5Vsb Standby	All models	V_{CC}	4.75	-	5.25	Vdc
Maximum Output Power	All models	$P_{O,max}$	-	-	2.5	W
Ambient Operating Temperature	All models	T_A	0	-	+40	ºC
Storage Temperature	All models	T_{STG}	-20	-	+50	ºC
Humidity (non-condensing)						
Operating	All models		20	-	90	%
Non-operating	All models		10	-	95	%
Altitude						
Operating	All models		-	-	10,000	feet
Non-operating	All models		-	-	30,000	feet

Hardware Signal Definition

Name of Signal:	A0, A1, A2
Description:	Pins that determine the address of the adapter.
Source of Input or Destination of output:	By default the adapter address is 111, but if connected to the PSU, the address could be changed through the header.
Physical Implementation:	A0, A1, A2 is connected to RA0, RA1, RA2 of the microcontroller and is connected to Vcc and to the header.
Valid Range, Accuracy, and/or Tolerance:	High or Low
Timing:	N/A

Name of Signal:	SCL, SDA
Description:	These pins are the clock and data pins of the I²C.
Source of Input or Destination of output:	PSU I²C (when connected to the header).
Physical Implementation:	Connected to the header.
Valid Range, Accuracy, and/or Tolerance:	High or Low
Timing:	I²C Frequency: 10 to 100kHz

Name of Signal:	UART_TX, UART_RX
Description:	transmit and receive pins of UART
Source of Input or Destination of output:	Other systems connected to the header.
Physical Implementation:	RC6 and RC7 connected to DI, and RO pins of the RS485 transceiver.
Valid Range, Accuracy, and/or Tolerance:	High or Low
Timing:	1.2kbps to 115.2 kbps

Name of Signal:	CAN_TX, CAN_RX
Description:	Transmit and receive pins of CAN
Source of Input or Destination of output:	Other systems connected to the header.
Physical Implementation:	RB2 and RB3 connected to RXD and TXD pins of the CAN transceiver.
Valid Range, Accuracy, and/or Tolerance:	High or Low
Timing:	125kbps to 500kbps

Name of Signal:	LED
Description:	pin that controls the LED output
Source of Input or Destination of output:	output controlled by microcontroller.
Physical Implementation:	RB5 connected to transistor.
Valid Range, Accuracy, and/or Tolerance:	High or Low
Timing:	N/A

Function Summary

Function Description	Sub-Function Description	Referenced Signals	Remarks
CAN communication	CAN transmit	CAN_TX	Refer to software requirements
	CAN receive	CAN_RX	Refer to software requirements
UART communication	UART transmit	UART_TX	Refer to software requirements
	UART receive	UART_RX	Refer to software requirements
I²C communication	I²C send	SDA, SCL	Refer to software requirements
	I²C receive	SDA, SCL	Refer to software requirements
Control LED (for 73-546-001 only)	N/A	LED	Should be ON when the unit is powered on. Blinking fast when an error is received.

4. Mechanical Specification

Connector Definition

73-546-001

Standard Mini-B USB to I^2C Interface.

Description	Pin Name	Pin No.
Bus Supply Voltage	VBUS	1
USB Data Low	DATA -	2
USB Data High	DATA +	3
USB ID	ID	4
Bus Supply Return	GND	5

Connector Type	Part No.	Vendor
Output	54819-0572	Molex
Mate	88732-8600	Molex

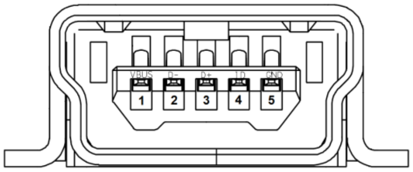


Figure 4. Connector Mini-USB

73-544-001 / 73-544-004

iVS CAN/RS485 to I^2C Interface.

Description	Pin Name	Pin No.
No Connection	N/C	1
CAN Bus L / RS485 A	CASE RX	2
CAN Bus H / RS485 B	CASE TX	3
Serial Clock Signal	SCL	4
Serial Data Signal	SDA	5
Address Bit 0	A0	6
Address Bit 1	A1	7
Address Bit 2	A2	8
Bus Supply Return	COMMON	9
5V System Bus	5VCC	10

Connector Type	Part No.	Vendor
Output	CI4405M1HR0	Molex
Mate	CI4405S0000	Molex

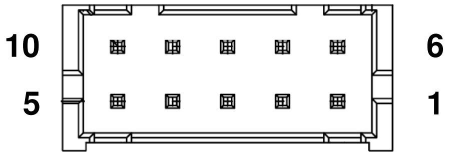


Figure 5. Connector J2

Connector Definition

73-544-002 / 73-544-003

iMP CAN/RS485 to I²C Interface.

Description	Pin Name	Pin No.
No Connection	N/C	1
CAN Bus L / RS485 A	CASE RX	3
CAN Bus H / RS485 B	CASE TX	5
Serial Clock Signal	SCL	7
Serial Data Signal	SDA	9
Address Bit 0	A0	2
Address Bit 1	A1	4
Address Bit 2	A2	6
Bus Supply Return	COMMON	8
5V System Bus	5VCC	10

Connector Type	Part No.	Vendor
Output	5016451020	Molex
Mate	5016461000	Molex
Terminal Pin	5016471000	Molex

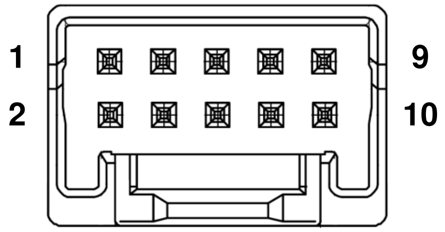


Figure 6. Connector J2

Weight

The 73-544-001 / 73-544-004 weight is 4.2 grams maximum.

The 73-544-002 / 73-544-003 weight is 3.8 grams maximum.

The 73-546-001 weight is 5.1 grams maximum.

5. Hardware Interfaces

See below system overview for 73-544-XXX.

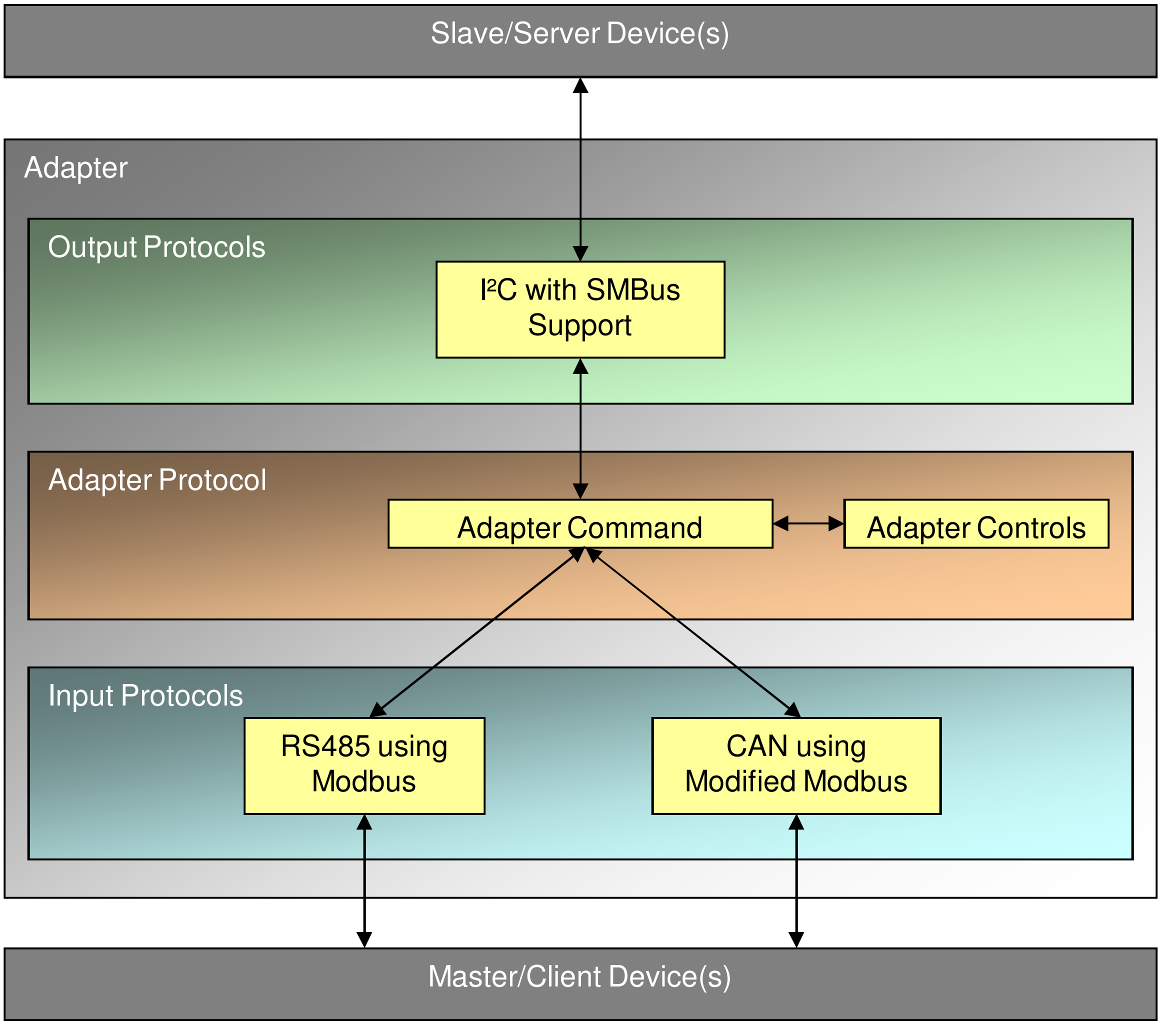


Figure 7. RS485/CAN-to-I^{2}C Overview

6. Software Interfaces

This specification defines the software design specification for the extension board adapter – for interfacing the iVS and iMP Series to CAN Bus and RS485 Bus. The extension board has microcontroller to convert I²C data lines to CAN Bus and RS485 Bus data lines.

6.1 Adapter Protocol Overview

The RS485/CAN-to-I²C uses 2 Input Protocols and 1 Output Protocol.

Protocol Type		Command Index
Input	RS485 using Modbus	0x01
Input	CAN using modified Modbus	0x02
Output	I²C with SMBus support	0x80

Table 3. Adapter Protocol

The Master/Client Device(s) initiate communication to the target Slave/Server Device(s) by choosing an available and supported Input Protocol of the Adapter (RS485 or CAN). The Master/Client sends an Adapter Command Packet to the Adapter Command Control using the chosen Input Protocol.

The Adapter Command Packet contains the Command Code needed for the Adapter to know which command the Master/Client requests. The Adapter Command Control decodes the requested command and performs it. If the command requested requires data transfer to a target Slave/Server through I²C, the Adapter Command Decoder passes the parameter to the Output Protocol. The Output Protocol then performs the transaction, and if successful – it transfers the resulting Output, if any, to the Adapter Command Control.

After performing the requested command, the Adapter Command Control then creates an Adapter Response Packet. The Adapter Response Packet contains Error Codes in determining the success of the command execution, and resulting output, if any. The Adapter Command Control then sends the Adapter Response Packet to the Master/Client through the same Input Protocol used in transferring the Adapter Command Packet.

If the Master/Client sends another transaction while the adapter is still processing the previous command, it will send a busy signal reply to the Master/Client.

6.2 Protocol Transaction

The Adapter Protocol defines the data packet sent by the Master/Client Device(s) to make the Adapter send data to the target Slave/Server Device(s) or control adapter functions. Master/Client Device(s) to Adapter transaction starts with the Master/Client Device(s) sending the Adapter an Adapter Command Packet, which is composed of the Command Code and its Parameters. If the Adapter successfully receives the Adapter Command Packet, the Adapter will perform the command requested. Upon completion of the command, the Adapter will respond by sending an Adapter Response Packet to the Master/Client Device(s) containing the Command Code, Error Code (to determine a successful completion or not), and other parameters requested. Figure 8 illustrates this transaction.

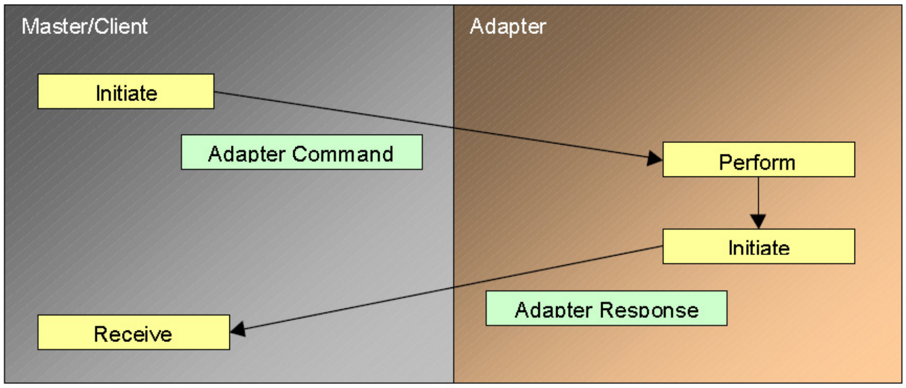


Figure 8. Adapter Protocol

6.3 Adapter Command and Response Packets

The Command Packet, as shown in Table 4, is composed of 2 Bytes of Command Code and optional 64 Bytes Parameter. The 1st Byte of the Command Code is the **Command Index**, which serves as grouping for the Adapter Commands, and the 2nd Byte is the **Command Function**, which specifies the function needed under the group indicated in the 1st Byte. The 2 Byte Command Code is then followed by optional **Parameter Bytes**. The length and content of the Parameter Bytes (if any) depends on the command requested and is included in the definition of the Command. The Command Parameters can have a size of up to 64 Bytes.

Byte 1 Command Index	Byte 2 Command Function	Byte 3 to 66 Command Parameters

Table 4. Command Packet

The Response Packet data frame, as shown in Table 5, is composed of 2 Bytes of Command Code, a 1 Byte Error Code, and optional 64 Bytes Output. The 2 Bytes of Command Code is the same Command Code received by the Adapter in the Adapter Command Packet. The 3rd Byte is the Error Code, which is used to determine if the command execution succeeds, or what type of error occurred. The Error Code is then followed by optional Command Output Bytes, provided that the Error Code returns no error. Depending on the error that occurred, there may be no Command Output. The length and content of the Command Output Bytes (if any) depends on the command requested and is included in the definition of the Command. The Command Output can have a size of up to 64 Bytes.

Byte 1 Command Index	Byte 2 Command Function	Byte 3 Error Code	Byte 4 to 67 Command Output

Table 5. Response Packet

6.3.1 Command Index

The Command Index is used to group together the available functions into Command Types. Table 6 below shows the summary and description of the Command Index grouped in each Command Type.

Command Index	Command Type	Description
0x00	Adapter Control	Control adapter specific functions and active input protocol control functions.
0x01 to 0x7F	Input Protocol Control	Controls input protocol specific functions. 0x01 – for RS485 using Modbus 0x02 – for CAN using modified Modbus
0x80 to 0xFF	Output Protocol Control	Controls output protocol specific functions and sending data through the output protocol. 0x80 - I²C with SMBus support

Table 6. Command Index Grouping

Adapter Control handles Functions for general Adapter Control. This includes: Adapter reset, Digital Output control (such as LED), Digital Input reading, etc. Adapter Control also contains functions in controlling the Active Input and Output Protocol, such as current Active Protocol reporting and switching. Command Index 0x00 is reserved for Adapter Control Command Type.

Input Protocol Control handles Functions for controlling the available Input Protocols. This includes: Baud/Clock rate control, Input Protocol Reset, etc. Command Index 0x01–0x7F is reserved for Input Protocol Control Command Type.

Output Protocol Control handles Functions for controlling the available Output Protocols. This includes: Baud/Clock rate control, Output Protocol Enabling/Disabling, Output Protocol Reset, etc. This also contains the specific commands to be able to send data using the specific Output Protocol, such as Reading and Writing data. Command Index 0x80–0xFF is reserved for Output Protocol Control Command Type.

6.3.2 Error Code

The Error Code is returned in the Adapter Response Packet to determine if the Command requested from the Adapter Command Packet is successful, or not. If not successful, the Error Code returned will help determine what error occurred.

For most Adapter Commands, Error Codes 0x00 to 0x04 are used. For these Error Codes, the Adapter Response Packet will always return no Command Output. Error Codes 0x05 to 0xFF are reserved to be define by the specific Adapter Command Function. Depending on the Command Function, Error Codes for this range may or may not return a Command Output in the Adapter Response Packet (see specific Command Function description). Table 7 below summarizes the Error Codes.

Error Code	Error Type
0x00	No Error
0x01	Inactive Input Protocol
0x02	Invalid Command Index
0x03	Invalid Command Function
0x04	Invalid Command Parameter
0x05	Inactive Output Protocol
0x06-0xFF	Command Function Defined

Table 7. Error Code Summary

The order by which Error Code are determined are as follows:

1. Checks if the Adapter Command Packet is received from the currently active Input Protocol. If the Adapter Command Packet is received from and Inactive Input Protocol, Error Code 0x01 is returned.

2. Check if the Command Index is valid and supported. If the Command Index is not supported or invalid, then Error Code 0x02 is returned.

3. Check if the Command Function is valid and supported. If the Command Function is not supported or invalid, then Error Code 0x03 is returned.

4. Check if the Command Parameter if valid and enough for the Command Function to execute. This includes too many/less Command Parameter Bytes, Command Parameter given if none is needed, invalid format, and invalid range. If the Command Parameter is not valid enough for the Command Function to execute, then Error Code 0x04 is returned. Note that some invalid Command Parameter may still allow the Command Function to execute. During this case, an Error Code of 0x05–0xFF is to be used. The Command Function definition is to clarify which cases of Invalid Command Parameter will return an Error Code 0x04, as well as when will a different Error Code (if any) will be returned and the corresponding response and Output.

5. The Command Function will then execute, and detect Function specific Errors (including some Invalid Command Parameter described above). If any error is found, the corresponding 0x05–0xFF will be returned, along with the corresponding Command Output (if any), as defined in the Command Function description. If no error is found after Command Function execution, Error Code 0x00 is returned, along with the corresponding Command Output (if any), as defined in the Command Function description.

6.4 Adapter Control Commands

Table 8 shows the supported Adapter Control Functions.

Function Code	Function Name	Parameter Length (Bytes)	Output Length (Bytes)
0x00	Adapter Version	0	3
0x02-0x0F	Reserved	---	---
0x10	Get Active Input Protocol	0	1
0x11	Set Active Input Protocol	1	1
0x12-0x1F	Reserved	---	---
0x20	Get Active Output Protocol	0	1
0x21	Set Active Output Protocol	1	1
0x22-0x1F	Reserved	---	---
0xF0	Go into Bootloader Mode	0	---
0xFF	Adapter Reset	0	See Description

Table 8. Adapter Control Commands

0x00 - Adapter Version

Parameter	None
Description	Returns current Adapter Firmware Version. The 3 Bytes returned are the Adapter Firmware Major, Minor and Test Version in decimal format. i.e. An Adapter Firmware with version 01.02.56 will return the following (in hex): Byte 1 – 0x01 Byte 2 – 0x02 Byte 3 – 0x38
Output	Byte 1 – Adapter Firmware Major Version. Byte 2 – Adapter Firmware Minor Version. Byte 3 – Adapter Firmware Test Version.
Additional Error Codes	0x05 – Not Applicable, None Output Protocol Function.

0x10 – Get Active Input Protocol

Parameter	None
Description	Returns 1 Byte of data containing the Command Index of the currently Active Input Protocol. If there is no Active Input Protocol, a 0x00 will be returned as Output. This Function should be available even on an Inactive Input Protocol, but should not cause that Input Protocol to be Active. Thus, Error Code 0x01 is ignored for this Function. See Section 6.5 for details on how Active Input Protocol is determined.
Output	Byte 1 – Current Active Input Protocol (0x01–0x7F).
Additional Error Codes	0x01 – Not applicable, see Description. 0x05 – Not Applicable, None Output Protocol Function.

0x11 – Set Active Input Protocol

Parameter	Byte 1 – Command Index of Input Protocol to set as Active (0x00, 0x01–0x7F).
Description	Sets the Active Input Protocol to the Input Protocol of the Command Index provided in the Parameter. Command Index provided must be within valid range and supported, or a 0x00 (None) for the function to execute, or an Invalid Command Parameter will occur. The current Active Input Protocol must also be Idle (i.e. not in the middle of a transaction) for it to be changed, otherwise, an Active Input Protocol Collision Error (0x06) will occur. On an Active Input Protocol Collision, the current Active Input Protocol will not be changed and will be returned as the Output. This Function should be available even on Inactive Input Protocol; thus, Error Code 0x01 is ignored on this Function. See Section 6.5 for details on how Active Input Protocol is determined.
Output	Byte 1 – Actual Active Input Protocol set (0x00, 0x01–0x7F).
Additional Error Codes	0x01 – Not applicable, see Description. 0x04 – Will not respond if Input Protocol given is outside of valid range (0x00, 0x01–0x7F), or not supported. 0x05 – Not Applicable, None Output Protocol Function. 0x06 – Active Input Protocol Collision. Current Active Input Protocol will not change, and will be returned as Output.

0x20 – Get Active Output Protocol

Parameter	None
Description	Returns 1 Byte of data containing the Command Index of the currently Active Output Protocol. If there is no Active Output Protocol, a 0x00 will be returned as Output. See Section 6.8 for details on how Active Output Protocol is determined.
Output	Byte 1 – Current Active Output Protocol (0x80–0xFF).
Additional Error Codes	0x05 – Not Applicable, None Output Protocol Function.

0x21 – Set Active Output Protocol

Parameter	Byte 1 – Command Index of Output Protocol to set as Active (0x00, 0x80–0xFF).
Description	Sets the Active Output Protocol to the Output Protocol of the Command Index provided in the Parameter. Command Index provided must be within valid range and supported, or a 0x00 (None) for the function to execute, or an Invalid Command Parameter will occur. The current Active Output Protocol must also be Idle (i.e. not in the middle of a transaction) for it to be changed, otherwise, an Active Output Protocol Collision Error (0x06) will occur. On an Active Output Protocol Collision, the current Active Output Protocol will not be changed and will be returned as the Output. See Section 6.8 for details on how Active Output Protocol is determined.
Output	Byte 1 – Actual Active Output Protocol set (0x00, 0x80–0xFF).
Additional Error Codes	0x04 – Will not respond if Output Protocol given is outside of valid range (0x00, 0x80–0xFF), or not supported. 0x05 – Not Applicable, None Output Protocol Function. 0x06 – Active Output Protocol Collision. Current Active Output Protocol will not change, and will be returned as Output.

0xF0 – Go into Bootloader Mode

Parameter	None
Description	It writes to the jump key that the adapter needs to go to bootloader mode. And resets the microcontroller. It then restarts and goes into bootloader mode.
Output	None
Additional Error Codes	None

0xFF – Adapter Reset

Parameter	None
Description	Performs software reset for the Adapter. This resets buffers and settings to default used by all Input, Adapter and Output Protocols. Since performing reset will not make returning of the Adapter Response Packet possible, the Master/Client should not request for Adapter Response Packet. The Master/Client should perform necessary profile/configuration clearing on its side if needed.
Output	See Description
Additional Error Codes	See Description

6.5 Active Input Protocol

Only one Input Protocol at a given instance can perform transfer of Adapter Command/Response Packets to the Adapter Protocol. The Input Protocol allowed for transfer is called the Active Input Protocol.

Upon initialization of the Adapter, the Active Input Protocol is set to None, flagged as 0x00 when checked using the Get Active Input Protocol function. Each Input Protocol that is available and ready for will then continuously listen/wait for transfer from the Master/Client. Once any of the Input Protocol received a successful transaction from the Master/Client (as defined on particular Input Parameter), the Adapter Protocol will then set that Input Protocol as the Active Input protocol. Once an Input Protocol becomes Active, any successful transaction from the other present Input Protocol will return an Inactive Input Protocol Error with few exceptions. The Active Input Protocol will remain as the Active Input Protocol until any of the following happens:

- No successful transaction occurs (idle) on the Active Input Protocol for at least 10 seconds. In this case, the Active Input Protocol will be go to None (0x00).
- An Adapter Reset function is commanded. In this case, the Active Input Protocol will be re–initialized and returned to None (0x00).
- A Set Active Input Protocol command was commanded and successfully (No Error, 0x00) performed. In this case, the Active Input Protocol is set to the new Input Protocol set in the Parameter.

A specific Function under the Command Index of the Input Protocol causes change in Active Input Protocol. New Active Input Protocol will depend on the definition of the Function.

6.6 Input Protocol – RS485 using Modbus (0x01)

Input Protocol 0x01 is assigned to the RS485 bus using the Modbus protocol. RTU (Remote Terminal Unit) mode is used for Modbus.

- Data are sent per byte and should not have more than 1.5 characters between each byte.
- If more than 3.5 characters spacing between next byte is achieved, the next byte is considered the next packet.

6.6.1 Server Address

Adapter Server Address when using this Input Protocol is determined by 3 Address Bits to set the Server Address.

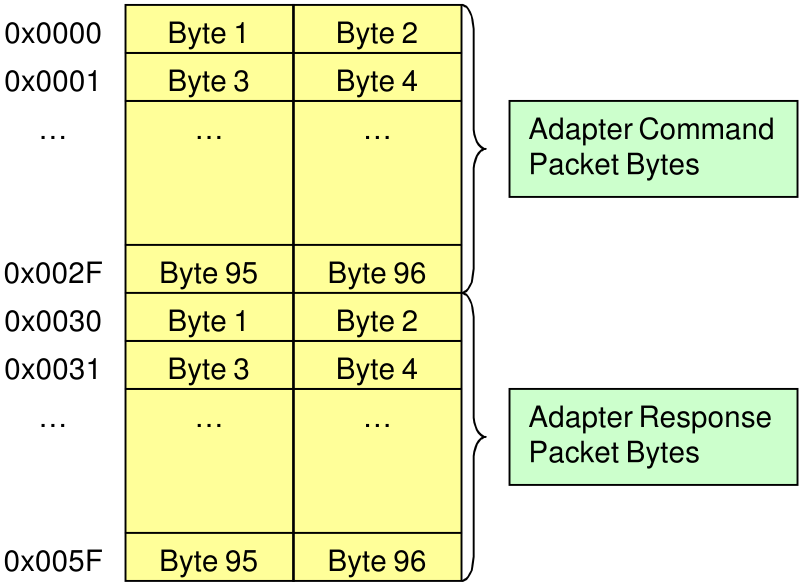
Table 9 below shows the summary of the Address Bit Logic vs. Adapter Server Address.

Server Address	A2	A1	A0
0x30 (0b00110000)	0	0	0
0x32 (0b00110010)	0	0	1
0x34 (0b00110100)	0	1	0
0x36 (0b00110110)	0	1	1
0x38 (0b00111000)	1	0	0
0x3A (0b00111010)	1	0	1
0x3C (0b00111100)	1	1	0
0x3E (0b00111110)	1	1	1

Table 9. Modbus Server Address Summary

6.6.2 Adapter Command/Response Packet Holding Registers

Holding registers 0x0000–0x003F are assigned to the Adapter Command Packets, and 0x0040–0x007F stores is assigned for the Adapter Response Packet. Note that each holding register holds 2 Bytes. In counting the Byte order, the MSB comes first, followed by the LSB. The mapping will then be: in 0x0000 of the holding register MSB will be Byte 1 of the Adapter Command Packet, 0x0000 LSB will be Byte 2, 0x0001 MSB will be Byte 3, 0x0001 LSB will be Byte 4 and so on; 0x0030 of the holding register MSB will be Byte 1 of the Adapter Response Packet, 0x0030 LSB will be Byte 2, 0x0031 MSB will be Byte 3, 0x0031 LSB will be Byte 4 and so on. Figure 9 below shows an illustration of the holding register mapping.



0x0000	Byte 1	Byte 2
0x0001	Byte 3	Byte 4
…	…	…
0x002F	Byte 95	Byte 96
0x0030	Byte 1	Byte 2
0x0031	Byte 3	Byte 4
…	…	…
0x005F	Byte 95	Byte 96

Figure 9. Modbus Holding Register Mapping

This Input Protocol will use the Modbus functions Read Holding Registers (0x03), Write Multiple Registers (0x10), and Read/Write Multiple Registers (0x17) to transfer the Adapter Command/Response Packet. The Write Multiple Registers is used to write the Adapter Command Packet (0x0000–0x002F). The Read Holding Registers is used to read the Adapter Response Packet (0x0030–0x005F). The Read/Write Multiple Registers can be used to perform a Write of the Adapter Command Packet followed by a Read of the Adapter Response Packet sequence. Writing and reading to the Holding Register always starts in the first byte.

6.6.2.1 Supported Modbus Functions

0x03 Read Holding Register:

Request

Function Code	1 Byte	0x03
Starting Address	2 Bytes	0x0030–0x005F
Quantity of Registers	2 Bytes	1–64

Response

Function Code	1 Byte	0x03
Byte Count	1 Byte	1–128
Register Value	1–128 Bytes	Adapter Response Packet

0x10 Write Multiple Registers:

Request

Function Code	1 Byte	0x10
Starting Address	2 Bytes	0x0000–0x002F
Quantity of Registers	2 Bytes	1–64
Byte Count	1 Bytes	1–128
Register Value	1–128 Bytes	Address Command Packet

Response

Function Code	1 Byte	0x10
Starting Address	2 Bytes	0x0030–0x005F
Quantity of Registers	2 Bytes	1–64

0x17 Read/Write Multiple Registers:

Request

Function Code	1 Byte	0x17
Read Starting Address	2 Bytes	0x0030–0x005F
Quantity to Read	2 Bytes	1–64
Write Starting Address	2 Bytes	0x0000–0x002F
Quantity of Write	2 Bytes	1–64
Write Byte Count	1 Bytes	1–128
Write Registers Value	1–128 Bytes	Address Command Packet

Response

Function Code	1 Byte	0x17
Byte Count	1 Byte	1–128
Read Registers Value	1–128 Bytes	Adapter Response Packet

6.6.3 Input Protocol Control Functions

Table 10 below shows the supported Functions for Input Protocol RS485 using Modbus.

Function Code	Function Name	Parameter Length (Bytes)	Output Length (Bytes)
0x00	Input Protocol Description	0	64
0x01	Get EUSART Baud Rate	0	1
0x02	Set EUSART Baud Rate	1	1
0x09	Get Read/Write Timeout	0	1
0x0A	Set Read/Write Timeout	1	1
0xFF	Input Protocol Reset	0	See Description

Table 10. Supported RS485 using Modbus Functions

0x00 – Input Protocol Description

Parameter	None
Description	Returns a string of characters that will describe the Input Protocol assigned in the Command Index. The data will be in ASCII format, with a New Line character ('\n', 0x0A) at the end. Bytes beyond the New Line will be padded with 0xFF.
Output	Byte 1–0x52 (R) Byte 2–0x53 (S) Byte 3–0x34 (4) Byte 4–0x38 (8) Byte 5–0x35 (5) Byte 6–0x20 ( ) Byte 7–0x75 (u) Byte 8–0x73 (s) Byte 9–0x69 (i) Byte 10–0x6E (n) Byte 11–0x67 (g) Byte 12–0x20 ( ) Byte 13–0x4D (M) Byte 14–0x6F (o) Byte 15–0x64 (d) Byte 16–0x62 (b) Byte 17–0x75 (u) Byte 18–0x73 (s) Byte 19–0x0A (\n) Byte 20–64–0xFF (pad)
Additional Error Codes	0x05 – Not Applicable, None Output Protocol Function.

0x01 – Get EUSART Baud Rate

Parameter	None
Description	Returns the current 1 Byte code for EUSART Baud. Possible Outputs will be: 0x01–0.3 Kbps 0x02–1.2 Kbps 0x03–2.4 Kbps 0x04–9.6 Kbps 0x05–19.2 Kbps 0x06–57.6 Kbps 0x07–115.2 Kbps
Output	Byte 1 – Code for current EUSART Baud Rate.
Additional Error Codes	None.

0x02 – Set EUSART Baud Rate

Parameter	Byte 1 – Code for desired EUSART Baud Rate.
Description	Sets current EUSART Baud Rate to the Desired EUSART Baud Rate in the parameter. Sending 0x00 will enable the Auto-Baud Detection. Possible Parameters will be: 0x00–Auto-detect 0x01–0.3 Kbps 0x02–1.2 Kbps 0x03–2.4 Kbps 0x04–9.6 Kbps 0x05–19.2 Kbps 0x06–57.6 Kbps 0x07–115.2 Kbps Other values will return an Invalid Parameter error (0x04). Output will return the code of EUSART Baud Rate set.
Output	Byte 1 – Code for current EUSART Baud Rate.
Additional Error Codes	0x04 – Will not respond if EUSART Baud Rate given is not of valid value (see Description).

0x09 – Get Read/Write Timeout

Parameter	None
Description	Returns the read timeout of RS485. (currently not used) Possible Outputs will be: 0x01–100ms 0x02–200ms : : 0xFF–25.5s
Output	Byte 1 – Actual Read Timeout set.
Additional Error Codes	None.

0x0A – Set Read/Write Timeout

Parameter	Byte 1 – time of read timeout before error (0x01 – 0xFF)
Description	Sets the read timeout of RS485. It is defaulted to 1 second if not set. It is incremented per 100ms. (currently not used) i.e. If the desired read timeout is 500ms, A value of 5 should be written in the parameter.
Output	Byte 1 – Actual Read Timeout set.
Additional Error Codes	0x04 – Will not respond if Read timeout given is not of valid value (see Description).

0xFF – Input Protocol Reset

Parameter	None.
Description	Performs software reset for the Input Protocol. This resets buffers and settings to default used by all Input Protocol. Since performing reset will not make returning of the Adapter Response Packet possible, the Master/Client should not request for Adapter Response Packet. The Master/Client should perform necessary profile/configuration clearing on its side if needed.
Output	See Description.
Additional Error Codes	See Description.

6.6.4 RS485 Transaction Error

Defined here are the Error Codes that will be used by functions in this Input Protocol that involves RS485 transfer (functions that refer in this in definition). See Table 11 below for Error Codes and description.

Error Code	Error Code	Commands
0x50	Read Timeout Error	A timeout occurred when reading in the CAN bus.

Table 11. RS485 Transfer Error Code Summary

6.7 Input Protocol – CAN using Modified Modbus (0x02)

Input Protocol 0x02 is assigned to the CAN bus using the as Modbus frame format, with modifications to take advantage of CAN higher layer features. RTU (Remote Terminal Unit) format is also for Modbus. Changes are as follows:

- The Server Address will be placed in the Identifier, instead as the 1st Byte of the Modbus Frame. A Standard (11-bit) Identifier will be used. Since Server Address is 8-bit, the last 3 bits (bits 8-10) of the Standard Identifier will be padded with 0's, and bits 0-7 will be the Server Address. Hence, the Modbus Frame will start with the Function Code.
- No Error Check Byte at the end of the Modbus Frame. The CRC checking in the CAN Frames will be used for data integrity checking.
- Since CAN Data Frames can have a maximum of 8 Bytes, and Modbus Frames can exceed 8 Bytes, the Client is to keep on sending the Modbus Frames in 8-Byte chunks. The CAN Input Protocol is to collect this chunk until a Data Frame with 0 Bytes in the Control Field Data Length Code (DLC) is received from the Client. Once the 0 Byte DLC Data Frame is received, the Modbus Frame will then be process and its function executed. The Client will then send a Remote Frame (assuming Data Frame is received by Server successfully) to request receipt of the Modbus Response Frame (which contains the Adapter Response Frame).
- The CAN Output Protocol will also send a Data Frame with 0 Bytes in the Control Field Data Length Code (DLC) to signal the Client that the transmission is complete.

6.7.1 Server Address

Adapter Server Address when using this Input Protocol is determined by 3 Address Bits to set the Server Address. The Address Bits is to be shared with that used by RS485 using Modbus Input Protocol. Hence, Modbus Server Address for both RS485 and CAN will be the same. Table 12 below shows the summary of the Address Bit Logic vs. Adapter Server Address. Note the 3-bit 0's pad on bits 8-10, as discussed in Section 6.7 above.

Server Address	A2	A1	A0
0x30 (0b00000110000)	0	0	0
0x32 (0b00000110010)	0	0	1
0x34 (0b00000110100)	0	1	0
0x36 (0b00000110110)	0	1	1
0x38 (0b00000111000)	1	0	0
0x3A (0b00000111010)	1	0	1
0x3C (0b00000111100)	1	1	0
0x3E (0b00000111110)	1	1	1

Table 12. Modbus Server Address Summary

6.7.2 Adapter Command/Response Packet Holding Registers

Holding Register mapping for CAN using modified Modbus Input Protocol is the same that used for RS485 using Modbus Input Protocol. Note that in MCU memory, the Holding Registers for CAN using modified Modbus Input Protocol is separate with that used by RS485 using Modbus Input Protocol. Refer to section 6.6.2 for Holding Register mapping details.

6.7.2.1 Supported Modbus Functions

CAN using modified Modbus Input Protocol supports the same Modbus Functions used for RS485 using Modbus Input Protocol. Refer to section 6.6.2.1 for supported Modbus function details.

6.7.3 Input Protocol Control Functions

Table 13 below shows the supported Functions for Input Protocol CAN using modified Modbus.

Function Code	Function Name	Parameter Length (Bytes)	Output Length (Bytes)
0x00	Input Protocol Description	0	64
0x01	Get CAN Baud Rate	0	1
0x02	Set CAN Baud Rate	1	1
0x09	Get Read Timeout	1	1
0x0A	Set Read Timeout	1	1
0xFF	Input Protocol Reset	0	See Description

Table 13. Supported CAN using modified Modbus

0x00 – Input Protocol Description

Parameter	None
Description	Returns a string of characters that will describe the Input Protocol assigned in the Command Index. The data will be in ASCII format, with a New Line character ('\n', 0x0A) at the end. Bytes beyond the New Line will be padded with 0xFF.
Output	Byte 1 - 0x43 (C) Byte 2 - 0x41 (A) Byte 3 - 0x4E (N) Byte 4 - 0x20 ( ) Byte 5 - 0x75 (u) Byte 6 - 0x73 (s) Byte 7 - 0x69 (i) Byte 8 - 0x6E (n) Byte 9 - 0x67 (g) Byte 10 - 0x20 ( ) Byte 11 - 0x6D (m) Byte 12 - 0x6F (o) Byte 13 - 0x64 (d) Byte 14 - 0x69 (i) Byte 15 - 0x66 (f) Byte 16 - 0x69 (i) Byte 17 - 0x65 (e) Byte 18 - 0x64 (d) Byte 19 - 0x20 ( ) Byte 20 - 0x4D (M) Byte 21 - 0x6F (o) Byte 22 - 0x64 (d) Byte 23 - 0x62 (b) Byte 24 - 0x75 (u) Byte 25 - 0x73 (s) Byte 26–0x0A (\n) Byte 27–64–0xFF (pad)
Additional Error Codes	0x05–Not Applicable, None Output Protocol Function.

0x01 – Get CAN Baud Rate

Parameter	None
Description	Returns the current 1 Byte code for CAN Baud. Possible Outputs will be: 0x01 – 10 Kbps 0x02 – 20 Kbps 0x03 – 50 Kbps 0x04 – 125 Kbps 0x05 – 250 Kbps 0x06 – 500 Kbps 0x07 – 800 Kbps 0x08 – 1 Mbps
Output	Byte 1 – Code for current CAN Baud Rate.
Additional Error Codes	None.

0x02 – Set CAN Baud Rate

Parameter	Byte 1 – Code for desired CAN Baud Rate.
Description	Sets current CAN Baud Rate to the Desired CAN Baud Rate in the parameter. Possible Parameters will be: 0x01 – 10 Kbps 0x02 – 20 Kbps 0x03 – 50 Kbps 0x04 – 125 Kbps 0x05 – 250 Kbps 0x06 – 500 Kbps 0x07 – 800 Kbps 0x08 – 1 Mbps Other values will return an Invalid Parameter error (0x04). Output will return the code of CAN Baud Rate set.
Output	Byte 1 – Code for current CAN Baud Rate.
Additional Error Codes	0x04 – Will not respond if CAN Baud Rate given is not of valid value (see Description).

0x09 – Get Read Timeout

Parameter	None
Description	Returns the read timeout of CAN. Possible Outputs will be: 0x01 – 100ms 0x02 – 200ms : : 0xFF – 25.5s
Output	Byte 1 – Actual Read Timeout set.
Additional Error Codes	None.

0x0A – Set Read Timeout

Parameter	Byte 1 – time of read timeout before error (0x01 – 0xFF)
Description	Sets the read timeout of CAN. It is defaulted to 1 second if not set. It is incremented per 100ms. I.e. If the desired read timeout is 500ms, a value of 5 should be written in the parameter.
Output	Byte 1 – Actual Read Timeout set.
Additional Error Codes	0x04 – Will not respond if Read timeout given is not of valid value (see Description).

0xFF – Input Protocol Reset

Parameter	None.
Description	Performs software reset for the Input Protocol. This resets buffers and settings to default used by all Input Protocol. Since performing reset will not make returning of the Adapter Response Packet possible, the Master/Client should not request for Adapter Response Packet. The Master/Client should perform necessary profile/configuration clearing on its side if needed.
Output	See Description.
Additional Error Codes	See Description.

6.7.4 CAN Transaction Error

Defined here are the Error Codes that will be used by functions in this Input Protocol that involves CAN transfer (functions that refer in this in definition). See Table 14 below for Error Codes and description.

Error Code	Error Type	Commands
0x60	Read Timeout Error	A timeout occurred when reading in the CAN bus. (DLC zero is not sent after read timeout is reached)

Table 14. CAN Transfer Error Code Summary

6.8 Active Output Protocol

For cases of multiple Output Protocols defined in the Adapter, only one Output Protocol at a given instance, can perform transfer of Adapter Command/Response Packets to the Adapter Protocol. The Output Protocol allowed for transfer is called the Active Output Protocol. Upon initialization of the Adapter, the Active Output Protocol is set to None, flagged as 0x00 when checked using the Get Active Output Protocol function (see Section 6.4). Once any of the Output Protocol has performed a successful transaction to the Slave/Server (as defined on particular Output Parameter), the Adapter Protocol will then set that Output Protocol as the Active Output protocol. Once an Output Protocol becomes Active, any transaction request for Slave/Server using other present Output Protocol will return an Inactive Output Protocol Error (see Section 6.3.2). The Active Output Protocol will remain as the Active Output Protocol until any of the following happens:

- No successful transaction occurs (idle) on the Active Output Protocol for at least 10 secs. In this case, the Active Output Protocol will be go to None (0x00).
- An Adapter Reset function is commanded. In this case, the Active Output Protocol will be re–initialized and returned to None (0x00).
- A Set Active Output Protocol command was commanded and successfully (No Error, 0x00) performed. In this case, the Active Output Protocol is set to the new Output Protocol set in the Parameter. See Section 6.4.

A specific Function under the Command Index of the Output Protocol causes change in Active Output Protocol. New Active Output Protocol will depend on the definition of the Function.

6.9 Output Protocol – I²C with SMBus Support (0x80)

Table 15 below shows the supported Functions for Output Protocol I²C with SMBus Support.

Function Code	Function Name	Parameter Length (Bytes)	Output Length (Bytes)
0x00	Output Protocol Description	0	64
0x01	Get I²C Frequency	0	2
0x02	Set I²C Frequency	2	2
0x10	I²C Write	Variable, 3–64	0
0x11	I²C Read	3	Variable, 1–64
0x20	SMBus Quick Command	1	0
0x21	SMBus Send Byte	3	0
0x22	SMBus Receive Byte	2	1
0x23	SMBus Write Byte/Word	Variable, 5–6	0
0x24	SMBus Read Byte/Word	4	Variable, 1–2
0x25	SMBus Block Write	Variable, 5–36	0
0x26	SMBus Block Read	3	Variable, 2–33
0x27	SMBus Process Call	Variable, 5–36	Variable, 2–33
0xFF	Output Protocol Reset	0	0

Table 15. Supported I²C with SMBus Output Protocol Functions

0x00 – Output Protocol Description

Parameter	None
Description	Returns a string of characters that will describe the Output Protocol assigned in the Command Index. The data will be in ASCII format, with a New Line character (‘\n’, 0x0A) at the end. Bytes beyond the New Line will be padded with 0xFF.
Output	Byte 1 – 0x49 (I) Byte 2 – 0x32 (2) Byte 3 – 0x43 (C) Byte 4 – 0x20 ( ) Byte 5 – 0x75 (u) Byte 6 – 0x73 (s) Byte 7 – 0x69 (i) Byte 8 – 0x6E (n) Byte 9 – 0x67 (g) Byte 10 – 0x20 ( ) Byte 11 – 0x53 (S) Byte 12 – 0x4D (M) Byte 13 – 0x42 (B) Byte 15 – 0x75 (u) Byte 16 – 0x73 (s) Byte 17 – 0x0A (\n) Byte 18 – 64–0xFF (pad)
Additional Error Codes	None.

0x01 – Get I²C Frequency

Parameter	None
Description	Returns the current 2 Bytes for I²C SCL frequency in KHz set in the configuration. Possible Output is from 10 KHz to 400 KHz. Default value is 100 KHz. Returned frequency accuracy is guaranteed only for frequencies from 10 KHz–100 KHz to be within 2%. Accuracy above 100 KHz is not guaranteed. I.e. I²C SCL frequency of 100 KHz will return: Byte 1–0x64. Byte 2–0x00.
Output	Byte 1 – I²C SCL frequency LSB. Byte 2 – I²C SCL frequency MSB. I²C SCL frequency is in KHz and is of Unsigned data format.
Additional Error Codes	None.

0x02 – Set I²C Frequency

Parameter	Byte 1 – Desired I²C SCL frequency LSB. Byte 2 – Desired I²C SCL frequency MSB. Desired I²C SCL frequency is in KHz and is of Unsigned data format.
Description	Sets current I²C SCL frequency to the Desired I²C SCL frequency in the parameter. Accepted Parameter is from 10 KHz to 400 KHz. Values outside if this range will result in and Invalid Parameter Error (0x04). While I²C SCL frequency of up to 400 KHz is accepted, actual I²C SCL frequency accuracy is only guaranteed up to 2% at 10 KHz–400 KHz. Accuracy at I²C SCL frequency of above 100 KHz is no guaranteed. I²C communication at I²C SCL frequency of above 100 KHz is also not guaranteed. Output will return the I²C SCL frequency set.
Output	Byte 1 – I²C SCL frequency (in KHz) LSB. Byte 2 – I²C SCL frequency (in KHz) MSB.
Additional Error Codes	0x04 – Will not respond if I²C SCL frequency given is outside of valid range (10 KHz–400 KHz).

0x10 – I²C Write

Parameter	Byte 1 – I²C Address. Byte 2 – Include Stop Bit. Byte 3 – Number of Data Bytes to write. Byte 4 – 64-Data Bytes to write. I²C Address uses 7-bit Addressing. Include Stop Bit is of Boolean data type. Number of Data Bytes is of Unsigned data format.
Description	Sends a Number of Data Bytes as stated in the parameter to the provided I²C Address. The I²C Address uses 7-bit addressing. In the I²C addressing, the 8th bit is not included in the addressing (masked). Number of Data Bytes accepted is from 0–61. Requesting to send a Number of Data Bytes outside of this range will result in an Invalid Parameter (0x04) error. The number of Data Bytes provided must also match the value of the Number of Data Bytes, or an Invalid Parameter (0x04) error will occur. Number of Data Bytes can be 0, in this case, there must be no data after Byte 3. The Data Bytes to be written (if any) will be taken from Bytes 4-64. Byte 4 will be sent 1st, followed by Byte 5, and so forth, until all Data Bytes is written. Byte 2 will be interpreted as a Boolean data type, and will be used to determine if a Stop Bit is sent at the end of writing. A Boolean value of TRUE will cause the Stop Bit to be sent, and a value of FALSE will not.
Output	None.
Additional Error Codes	0x04 – Will not respond if the Number of Data Bytes given is outside of valid range (0-64); or number of Data Bytes provided does match with the provided Number of Data Bytes to write. Refer to Section 6.9.2 for I²C Transaction Errors.

0x11 – I²C Read

Parameter	Byte 1 – I²C Address. Byte 2 – Include Stop Bit. Byte 3 – Number of Data Bytes to read. I²C Address uses 7-bit Addressing. Include Stop Bit is of Boolean data type. Number of Data Bytes is of Unsigned data format.
Description	Reads a Number of Data Bytes as stated in the parameter to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Number of Data Bytes accepted is from 1–64. Requesting to read data outside of this range will result in an Invalid Parameter (0x04) error. The Data Bytes received will be placed at Bytes 1–64 of the Output. 1st Data Byte read is placed to Byte 1, 2nd Data Byte received to Byte 2, and so forth, until a number of Data Byte equal to the Number of Data Bytes is read. Byte 2 will be interpreted as a Boolean data type, and will be used to determine if a Stop Bit is sent send at the end of writing. A Boolean value of TRUE will cause the Stop Bit to be sent, and a value of FALSE will not.
Output	Byte 1-64 – Data Bytes read.
Additional Error Codes	0x04 – Will not respond if the Number of Data Bytes given is outside of valid range (1–64). Refer to Section 6.9.2 for I²C Transaction Errors.

0x20 – SMBus Quick Command

Parameter	Byte 1 – I²C Address. I²C Address uses 7-bit Addressing.
Description	Performs a Quick Command, as defined in the SMBus protocol, to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked).
Output	None.
Additional Error Codes	Refer to Section 6.9.2 for I²C Transaction Errors.

0x21 – SMBus Send Byte

Parameter	Byte 1 – I²C Address. Byte 2 – Data Byte to send. Byte 3 – PEC Enable. I²C Address uses 7-bit Addressing. PEC Enable is of Boolean data type.
Description	Performs a Send Byte, as defined in the SMBus protocol, to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). The Data Byte to send is in Byte 2. Byte 3 flags if PEC will be placed at the end of the packet (TRUE) or not (FALSE).
Output	None.
Additional Error Codes	Refer to Section 6.9.2 for I²C Transaction Errors.

0x22 – SMBus Receive Byte

Parameter	Byte 1 – I²C Address. Byte 2 – PEC Enable. I²C Address uses 7-bit Addressing. PEC Enable is of Boolean data type.
Description	Performs a Receive Byte, as defined in the SMBus protocol, to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Byte 3 flags if PEC of the received packet will be checked (TRUE) or not (FALSE). The Data Byte read is placed at Byte 1 of the Output.
Output	Byte 1 – Data Byte Received.
Additional Error Codes	Refer to Section 6.9.2 for I²C Transaction Errors.

0x23 – SMBus Write Byte/Word

Parameter	Byte 1 – I²C Address. Byte 2 – Command Code. Byte 3 – Number of Data Bytes to write. Byte 4 – PEC Enable. For Write Byte: Byte 5 – Data Byte to write. For Write Word: Byte 5 – Data Word to write LSB. Byte 6 – Data Word to write MSB. I²C Address uses 7-bit Addressing. Number of Data Bytes is of Unsigned data format. PEC Enable is of Boolean data type.
Description	Performs a Write Byte/Word, as defined in the SMBus protocol, to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Byte 2 contains Command Code to be used. Byte 3 defines the Number of Data Bytes to be written. Number of Data Bytes to be written can only be a value of 1 (for Write Byte) or 2 (for Write Word), otherwise an Invalid Parameter (0x04) error will occur. The Data Bytes to be written will be taken from Bytes 5–6. For Write Byte, Byte 5 will contain the Data Byte, and there must be no Byte 6. For Write Word, Byte 5 will contain the LSB of the Data Word, and Byte 6 will contain the MSB. Non–conformance will result in an Invalid Parameter (0x04) error. Byte 4 flags if PEC will be placed at the end of the packet (TRUE) or not (FALSE).
Output	None.
Additional Error Codes	0x04 – Will not respond if the Number of Data Bytes given is outside of valid range (1–2); or number of Data Bytes provided does match with the provided Number of Data Bytes to write. Refer to Section 6.9.2 for I²C Transaction Errors.

0x24 – SMBus Read Byte/Word

Parameter	Byte 1 – I²C Address. Byte 2 – Command Code. Byte 3 – Number of Data Bytes to read. Byte 4 – PEC Enable. I²C Address uses 7-bit Addressing. Number of Data Bytes is of Unsigned data format. PEC Enable is of Boolean data type.
Description	Performs a Read Byte/Word, as defined in the SMBus protocol, to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Byte 2 contains Command Code to be used. Byte 3 defines the Number of Data Bytes to be read. Number of Data Bytes to be read can only be a value of 1 (for Read Byte) or 2 (for Read Word), otherwise an Invalid Parameter (0x04) error will occur. Byte 4 flags if the PEC of the read packet will be checked (TRUE) or not (FALSE). The Data Bytes read will be placed to Bytes 5–6. For Read Byte, the Data Byte will be placed to Byte 5 and there will be no Byte 6. For Write Word, the LSB of the Data Word will be placed to Byte 5, and the MSB will be placed to Byte 6.
Output	For Read Byte: Byte 1 – Data Byte read. For Read Word: Byte 1 – Data Word read LSB. Byte 2 – Data Word read MSB.
Additional Error Codes	0x04 – Will not respond if the Number of Data Bytes given is outside of valid range (1–2). Refer to Section 6.9.2 for I²C Transaction Errors.

0x25 – SMBus Block Write

Parameter	Byte 1 – I²C Address. Byte 2 – Command Code. Byte 3 – Number of Data Bytes to write. Byte 4 – PEC Enable. Byte 5 – 36–Data Bytes to be written. I²C Address uses 7-bit Addressing. Number of Data Bytes is of Unsigned data format. PEC Enable is of Boolean data type.
Description	Performs a Block Write, as defined in the SMBus protocol, to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Byte 2 contains Command Code to be used. Byte 3 defines the Number of Data Bytes to be written. Number of Data Bytes to be written can be a value from 1–32, otherwise an Invalid Parameter (0x04) error will occur. The number of the Data Bytes provided must also match the value of the Number of Data Bytes, or an Invalid Parameter (0x04) error will occur. The Data Bytes to be written will be taken from Bytes 5–36. Byte 5 will be sent 1st, followed by Byte 6, and so forth, until all Data Bytes is written. Byte 4 flags if PEC will be placed at the end of the packet (TRUE) or not (FALSE).
Output	None.
Additional Error Codes	0x04 – Will not respond if the Number of Data Bytes given is outside of valid range (1–32); or number of Data Bytes provided does match with the provided Number of Data Bytes to write. Refer to Section 6.9.2 for I²C Transaction Errors.

0x26 – SMBus Block Read

Parameter	Byte 1 – I²C Address. Byte 2 – Command Code. Byte 3 – Data bytes to read Byte 4 – PEC Enable. I²C Address uses 7-bit Addressing. Number of Data Bytes is of Unsigned data format. PEC Enable is of Boolean data type.
Description	Performs a Block Read, as defined in the SMBus protocol, to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Byte 2 contains Command Code to be used. Byte 3 flags if the PEC of the read packet will be checked (TRUE) or not (FALSE). The Number of Data Bytes read will be placed to Byte 1 of the Output. The Data Bytes read will be placed at Bytes 2–33 of the Output. 1st Data Byte read is placed to Byte 1, 2nd Data Byte received to Byte 2, and so forth, until a number of Data Byte equal to the Number of Data Bytes is read.
Output	Byte 1 – Number of Data Bytes read. Byte 2 – 33 Data Bytes read.
Additional Error Codes	Refer to Section 6.9.2 for I²C Transaction Errors.

0x27 – SMBus Process Call

Parameter	Byte 1 – I²C Address. Byte 2 – Command Code. Byte 3 – Number of Data Bytes to write. Byte 4 – PEC Enable. Byte 5 – 36 Data Bytes to write. I²C Address uses 7-bit Addressing. Number of Data Bytes is of Unsigned data format. PEC Enable is of Boolean data type.
Description	Performs a Process Call, as defined in the SMBus protocol, to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Byte 2 contains Command Code to be used. Byte 3 defines the Number of Data Bytes to be written. Number of Data Bytes to be written can be a value from 1–31, otherwise an Invalid Parameter (0x04) error will occur. The number of the Data Bytes provided must also match the value of the Number of Data Bytes, or an Invalid Parameter (0x04) error will occur. The Data Bytes to be written will be taken from Bytes 5–36. Byte 5 will be sent 1st, followed by Byte 6, and so forth, until all Data Bytes is written. Byte 4 flags if PEC will be placed at the end of the sent packet (TRUE) or not (FALSE). It also flags if the PEC of the read packet will be checked (TRUE) or not (FALSE). The Number of Data Bytes read will be placed to Byte 1 of the Output. The Data Bytes read will be placed at Bytes 2–33 of the Output. 1st Data Byte read is placed to Byte 1, 2nd Data Byte received to Byte 2, and so forth, until a number of Data Byte equal to the Number of Data Bytes is read.
Output	Byte 1 – Number of Data Bytes read. Byte 2 – 33 Data Bytes read.
Additional Error Codes	0x04 – Will not respond if the Number of Data Bytes given is outside of valid range (1–31); or number of Data Bytes provided does match with the provided Number of Data Bytes to write. Refer to Section 6.9.2 for I²C Transaction Errors.

0xFF – Output Protocol Reset

Parameter	None.
Description	Performs software reset for the I²C Module. Clears all TX and RX buffers, and resets I²C SCL clock to 100 KHz.
Output	None.
Additional Error Codes	None.

6.9.2 I²C Transaction Error

Defined here are the Error Codes that will be used by functions in this Output Protocol that involves I²C transfer (functions that refer in this in definition). See Table 16 below for Error Codes and description.

Error Code	Error Type	Commands
0x10	Address NACK	NACK occurred when Slave Address was Written to I²C bus.
0x11	Data NACK	NACK occurred when Slave Address was Written to I²C bus.
0x20	Bus Collision	I²C Bus in use when Start or Stop bit was being sent.
0x21	Write Collision	I²C Bus in use when doing I²C Write.
0x31	Idle Timeout	Timeout occurred when waiting for I²C bus to be idle.
0x32	Start Timeout	Timeout occurred when waiting for sending of Start bit to finish
0x33	Restart Timeout	Timeout occurred when waiting for sending of Restart bit to finish
0x34	Stop Timeout	Timeout occurred when waiting for sending of Stop bit to finish
0x35	Read Timeout	Timeout occurred when waiting for Read to I²C bus to finish
0x36	ACK Timeout	Timeout occurred when waiting for sending of ACK to finish
0x40	Buffer Limit (Hardware buffer)	MCU Hardware Buffer flags overflow
0x41	CRC Error	CRC computation is not equal to the PEC sent

Table 16. I²C Transfer Error Code Summary

7. Interface Protocol – Controller Area Network (CAN)

7.1 Command / Response Packet

Byte 1	Byte 2	Byte 3 to 66
Command Index	Command Function	Command Parameters

Command Packet

Byte 1	Byte 2	Byte 3	Byte 4 to 67
Command Index	Command Function	Error Code	Command Output

Response Packet

7.2 Adapter Protocol Commands

Using Command Index – Adapter Control (0x00).

ADAPTER VERSION (0x00)

This command returns current Adapter Firmware Version. The 3 bytes returned are the Adapter Firmware Major, Minor, and Test Version in decimal format.

Example for CAN:

Send command (in Hex):

CAN ID	Data Length	Command Index	Command code
3E	**2**	**00**	**00**

Then, send command:

CAN ID	Data Length
3E	**0**

Since CAN is 8-bytes data transmission, data length 0 must be send to indicate end of transmission.

Reply (in Decimal):

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6
6	**2**	**0**	**0**	**0**	**1**	**0**	**0**

Output:

The data is interpreted in hexadecimal as version “**01.00.00**”

Byte 4 – 0x01

Byte 5 – 0x00

Byte 6 – 0x00

In reply, Data Byte 1 and Data Byte 2 returned the command index and command code in decimal form while Data Byte 3 indicate error code and it should return 0 which pertains “No Error”, or else a transaction error occurred.

7.3 Input Protocol Control Commands

Using Command Index - CAN Modified Modbus (0x02).

INPUT PROTOCOL DESCRIPTION (0x00)

This command returns a string of characters that will describe the Input Protocol assigned in the command index. The data will be in ASCII format, with a new line character ('\n', 0x0A) at the end. Bytes beyond the new line will be padded with 0xFF.

Example for CAN:

Send command (in Hex):

CAN ID	Data Length	Command Index	Command code
3E	**2**	**02**	**00**

Then, send command:

CAN ID	Data Length
3E	**0**

Since CAN is 8-bytes data transmission, data length 0 must be send to indicate end of transmission.

Reply (in Decimal):

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**8**	**2**	**0**	**0**	**67**	**65**	**78**	**32**	**117**

1st 8-data byte

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**8**	**115**	**105**	**110**	**103**	**32**	**109**	**111**	**100**

2nd 8-data byte

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**8**	**105**	**102**	**105**	**101**	**100**	**32**	**77**	**111**

3rd 8-data byte

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**8**	**100**	**98**	**117**	**115**	**10**	**255**	**255**	**255**

4th 8-data byte

Output:

Converting Data Byte 4 (67d) of 1st set of 8 data byte up to Data Byte 4 of the 4th set of 8 data byte to its ASCII equivalent, the result will be a string of "**CAN using modified Modbus**". Succeeding byte will be 255d (FFh).

GET CAN BAUD RATE (0x01)

This command will return the current 1 byte code for CAN Baud rate. The command code for Adapter version for CAN and RS485 Adapter is 0x01.

Example for CAN:

Send command (in Hex):

CAN ID	Data Length	Command Index	Command code
3E	**2**	**02**	**01**

Then, send command:

CAN ID	Data Length
3E	**0**

Since CAN is 8-bytes data transmission, data length 0 must be send to indicate end of transmission.

Reply (in Decimal):

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4
1	**4**	**2**	**1**	**0**	**4**

Output:

CAN Adapter Baud rate is 125 Kbps.

Possible Outputs are: (in hex)

0x01 – 10 Kbps

0x02 – 20 Kbps

0x03 – 50 Kbps

0x04 – 125 Kbps

0x05 – 250 Kbps

0x06 – 500 Kbps

0x07 – 800 Kbps

0x08 – 1 Mbps

7.4 Output Protocol Control Commands

Using Command Index – I²C with SMBus support (0x80).

OUTPUT PROTOCOL DESCRIPTION (0x00)

This command returns a string of characters that will describe the Output Protocol assigned in the command index. The data will be in ASCII format, with a new line character ('\n', 0x0A) at the end. Bytes beyond the new line will be padded with 0xFF.

Example for CAN:

Send command (in Hex):

CAN ID	Data Length	Command Index	Command code
3E	**2**	**80**	**00**

Then, send command:

CAN ID	Data Length
3E	**0**

Since CAN is 8-bytes data transmission, data length 0 must be send to indicate end of transmission.

Reply (in Decimal):

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**8**	**128**	**0**	**0**	**73**	**50**	**67**	**32**	**117**

1st 8-data byte

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**8**	**115**	**105**	**110**	**103**	**32**	**83**	**77**	**66**

2nd 8-data byte

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**8**	**117**	**115**	**10**	**0**	**255**	**255**	**255**	**255**

3rd 8-data byte

Output:

Converting Data Byte 4 (73d) of 1st set of 8 data bytes up to Data Byte 2 of the 3rd set of 8 data byte to its ASCII equivalent, the result will be "**I²C using SMBus**". Succeeding byte will be 255d (FFh).

GET I²C FREQUENCY (0x01)

This command will return the current 2 bytes I²C SCL frequency in kHz set in the configuration. Possible output is from 10kHz to 400kHz. Default value is 100kHz. Returned frequency accuracy is guaranteed only for frequencies from 10kHz-100kHz to be within 2%. Accuracy above 100kHz is not guaranteed.

Example for CAN:

Send command (in Hex):

CAN ID	Data Length	Command Index	Command code
3E	**2**	**80**	**01**

Then, send command:

CAN ID	Data Length
3E	**0**

Since CAN is 8-bytes data transmission, data length 0 must be send to indicate end of transmission.

Reply (in Decimal):

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5
1	**8**	**128**	**1**	**0**	**100**	**0**

Output:

The default I²C frequency of the Adapter is 100kHz. Data Byte 4 is the I²C SCL frequency LSB and Data Byte 5 is the I²C SCL frequency MSB. I²C SCL frequency is in kHz and is have unsigned data format.

I²C WRITE (0x10) AND READ (0x11)

Reads a Number of Data Bytes as stated in the parameter to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Number of Data Bytes accepted is from 1–64. Requesting to read data outside of this range will result in an Invalid Parameter (0x04) error.

The Data Bytes received will be placed at Bytes 1–64 of the Output. 1st Data Byte read is placed to Byte 1, 2^{nd} Data Byte received to Byte 2, and so forth, until a number of Data Byte equal to the Number of Data Bytes is read.

Byte 2 will be interpreted as a Boolean data type, and will be used to determine if a Stop Bit is sent send at the end of writing. A Boolean value of TRUE will cause the Stop Bit to be sent, and a value of FALSE will not. I²C Read involves I²C write to determine which location on the memory map you are going to read.

READ MODEL NUMBER (From EEPROM of Power Supply)

Example for CAN:

Send I²C Write command (in Hex):

CAN ID	Data Length	Command Index	Command code	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4
3E	**6**	**80**	**10**	**AE**	**01**	**01**	**19**

Command Index for Output Protocol is 0x80 and Command Code for I²C Write is 0x10. Data Byte 1 indicates the I²C address or the external EEPROM address (0xA0, 0xA1, 0xA2 floating), Data Byte 2 tells to include stop bit and Data Byte 3 is the number of data bytes to write. Model Number is located at EEPROM offset 0x19-0x24 (12 bytes). Data bytes 4-64 are the data bytes to write.

Then, send command:

CAN ID	Data Length
3E	**0**

Since CAN is 8-bytes data transmission, data length 0 must be send to indicate end of transmission.

Reply (in Decimal):

ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3
1	**3**	**128**	**16**	**0**

Output:

Data Byte 1 and 2 returned the command index and command code respectively and Data Byte 3 gives a value of 0 that indicates No error or successful write.

Send I²C Read command (in Hex):

CAN ID	Data Length	Command Index	Command Code	Data Byte 1	Data Byte 2	Data Byte 3
3E	**5**	**80**	**11**	**AE**	**01**	**0C**

Then, send command:

CAN ID	Data Length
3E	**0**

Since CAN is 8-bytes data transmission, data length 0 must be send to indicate end of transmission.

Reply (in Decimal):

CAN ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**8**	**128**	**17**	**0**	**55**	**51**	**45**	**53**	**56**

1st 8 data byte

CAN ID	Data Length	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
1	**8**	**48**	**45**	**48**	**48**	**48**	**49**	**73**

2nd 8 data byte

Output:

Converting Data Byte 4 (55d) of 1st set of 8 data byte up to Data Byte 5 of the 2nd set of 8 data byte to its ASCII equivalent, the result will be “**73-580-0001I**”.

8. Interface Protocol – RS485

8.1 Command / Response Packet

Byte 1 **Command Index**	**Byte 2** **Command Function**	**Byte 3 to 66** **Command Parameters**

Command Packet

Byte 1 **Command Index**	**Byte 2** **Command Function**	**Byte 3** **Error Code**	**Byte 4 to 67** **Command Output**

Response Packet

8.

8.2 Adapter Control Commands

Using Command Index - Adapter Control (0x00).

ADAPTER VERSION (0x00)

This command returns current Adapter Firmware Version. The 3 bytes returned are the Adapter Firmware Major, Minor, and Test Version in decimal format.

Example for RS485:

Send command (in Hex):

Address	Command Index	Command Code
3E	**00**	**00**
Decimal: **62,0,0,**		

Reply (in Decimal):

Command Index	Command Code	Error Code	Read Data	Read Data	Read Data
Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6
0	**0**	**0**	**1**	**0**	**0**

Output:

The data is interpreted in hexadecimal as version “**01.00.00**”

Byte 4 – 0x01

Byte 5 – 0x00

Byte 6 – 0x00

In reply, Data Byte 1 and Data Byte 2 returned the command index and command code in decimal form while Data Byte 3 indicate error code and it should return 0 which pertains “No Error”, or else a transaction error occurred.

8.3 Input Protocol Control Commands

Using Command Index - RS485 using Modbus (0x01).

INPUT PROTOCOL DESCRIPTION (0x00)

This command returns a string of characters that will describe the Input Protocol assigned in the command index. The data will be in ASCII format, with a new line character ('\n', 0x0A. 10d) at the end. Bytes beyond the new line will be padded with 0xFF.

Example for RS485:

Send command (in Hex):

Address	Command Index	Command Code
3E	**01**	**00**
Decimal: **62,1,0,**		

Reply (in Decimal):

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
1	**0**	**0**	**82**	**83**	**52**	**56**	**53**

1st 8-data byte

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
32	**117**	**115**	**105**	**110**	**103**	**032**	**77**

2nd 8-data byte

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
111	**100**	**98**	**117**	**115**	***10***	**255**	**255**

3rd 8-data byte

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
255	**255**	**255**	**255**	**255**	**255**	**255**	**255**

4th 8-data byte

Output:

Converting Data Byte 4 (82d) of 1st set of 8 data byte up to Data Byte 5 of the 3rd set of 8 data byte to its ASCII equivalent, the result will be a string of "**RS485 using Modbus**". Succeeding byte will be 255d (FFh).

GET CAN BAUD RATE (0x01)

This command will return the current 1 byte code for RS485 Baud rate. The command code for Adapter version for RS485 Adapter is 0x01.

The Default RS485 Setting – Baud Rate: 9.6 kbps, Data: 8, Parity: None, Stop:1

Example for RS485:

Send command (in Hex):

Address	Command Index	Command Code
3E	**01**	**01**
Decimal: **62,1,1,**		

Reply (in Decimal):

Command Index	Command Code	Error Code	Read Data	Read Data
Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5
1	**1**	**0**	**4**	**0**

Output:

RS485 Adapter Baud rate is 9.6 Kbps.

Default Output is: (in hex)

0x04 – 9.6 Kbps

8.4 Output Protocol Control Commands

Using Command Index – I²C with SMBus Support (0x80).

OUTPUT PROTOCOL DESCRIPTION (0x00)

This command returns a string of characters that will describe the Output Protocol assigned in the command index. The data will be in ASCII format, with a new line character ('\n', 0x0A, 10d) at the end. Bytes beyond the new line will be padded with 0xFF.

Example for RS485:

Send command (in Hex):

Address	Command Index	Command Code
0x3E	**0x80**	**0x00**
Decimal: **62,128,0,**		

Reply (in Decimal):

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
128	**0**	**0**	**73**	**50**	**67**	**32**	**117**

1st 8-data byte

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
115	**105**	**110**	**103**	**32**	**83**	**77**	**66**

2nd 8-data byte

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
117	**115**	***10***	**0**	**255**	**255**	**255**	**255**

3rd 8-data byte

Output:

Converting Data Byte 4 (73d) of 1st set of 8 data bytes up to Data Byte 2 of the 3rd set of 8 data byte to its ASCII equivalent, the result will be "**I²C using SMBus**". Succeeding byte will be 255d (FFh).

GET I²C FREQUENCY (0x01)

This command will return the current 2 bytes I²C SCL frequency in kHz set in the configuration. Possible output is from 10kHz to 400kHz. Default value is 100kHz. Returned frequency accuracy is guaranteed only for frequencies from 10kHz-100kHz to be within 2%. Accuracy above 100kHz is not guaranteed.

Example for RS485:

Send command (in Hex):

Address	Command Index	Command code
3E	**80**	**01**
Decimal: **62,128,1**		

Reply (in Decimal):

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5
128	**1**	**0**	**100**	**0**

Output:

The default I²C frequency of the Adapter is "**100kHz**". Data Byte 4 is the I²C SCL frequency LSB and Data Byte 5 is the I²C SCL frequency MSB. I²C SCL frequency is in kHz and is have unsigned data format.

I²C WRITE (0x10) AND READ (0x11)

Reads a Number of Data Bytes as stated in the parameter to the provided I²C Address. In the I²C addressing, the 8th bit is not included in the addressing (masked). Number of Data Bytes accepted is from 1–64. Requesting to read data outside of this range will result in an Invalid Parameter (0x04) error.

The Data Bytes received will be placed at Bytes 1–64 of the Output. 1st Data Byte read is placed to Byte 1, 2nd Data Byte received to Byte 2, and so forth, until a number of Data Byte equal to the Number of Data Bytes is read.

Byte 2 will be interpreted as a Boolean data type, and will be used to determine if a Stop Bit is sent send at the end of writing. A Boolean value of TRUE will cause the Stop Bit to be sent, and a value of FALSE will not. I²C Read involves I²C write to determine which location on the memory map you are going to read.

READ MODEL NUMBER (From EEPROM of Power Supply)

Example for RS485:

Send I²C Write command (in Hex):

Address	Command Index	Command code	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4
3E	**80**	**10**	**AE**	**1**	**1**	**19**
62,128,16,174,1,1,25,						

Command Index for Output Protocol is 0x80 and Command Code for I²C Write is 0x10. Data Byte 1 indicates the I²C address or the external EEPROM address (0xA0, 0xA1, 0xA2 floating), Data Byte 2 tells to include stop bit and Data Byte 3 is the number of data bytes to write. Model Number is located at EEPROM offset 0x19-0x24 (12 bytes).

Data bytes 4-64 are the data bytes to write.

Reply (in Decimal):

Data Byte 1	Data Byte 2	Data Byte 3
128	**16**	**0**

Output:

Data Byte 1 and 2 returned the command index and command code respectively and Data Byte 3 gives a value of 0 that indicates No error or successful write.

Send I²C Read command (in Hex):

Address	Command Index	Command Code	Data Byte 1	Data Byte 2	Data Byte 3
3E	**80**	**11**	**AE**	**01**	**0C**
62,128,17,174,1,12,					

Reply (in Decimal):

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
128	**17**	**0**	**51**	**55**	**45**	**53**	**56**

1st 8 data byte

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
48	**45**	**48**	**48**	**48**	**49**	**73**

2nd 8 data byte

Output:

Converting Data Byte 4 (51d) of 1st set of 8 data byte up to Data Byte 7 of the 2nd set of 8 data byte to its ASCII equivalent, the result will be “**73-580-00001I**”.

9. Interface Protocol – PMBus (For iMP Series Power Supplies)

9.1 Introduction

This iMP Protocol is compliant with the PMBus Power System Management Protocol Specification Part I Revision 1.0, and the PMBus Power System Management Protocol Specification Part II Revision 1.0. Note that the PMBus is based on the System Management Bus (SMBus) Specification. For supplementary information regarding the PMBus and SMBus specifications, these documents will be referred to, and are considered part of this protocol:

Reference 1	PMBus™ Power System Management Protocol Specification, Part I – General Requirements, Transport And Electrical Interface, Revision 1.0 www.powerSIG.org
Reference 2	PMBus™ Power System Management Protocol Specification, Part II – Command Language, Revision 1.0 www.powerSIG.org
Reference 3	System Management Bus Specification, Revision 1.1 www.sbs-forum.org
Reference 4	System Management Bus (SMBus) Specification, Version 2.0 www.sbs-forum.org

9.2 SMBus Compliance

Packet Error Checking

Packet Error Checking (PEC) is optional in SMBus. The iMP secondary firmware version 2.06 or later supports PEC.

Ack/Nack

This version of the protocol does not issue a Nack in any case of errors above the data link layer. Communication reliability can be achieved though read-back and/or through fault flags.

Bus Protocols

This protocol supports all SMBus bus protocols except the Quick Command and the Host Notify Protocol. Shown below are excerpts from the System Management Bus (SMBus) Specification, Version 2.0 document for easy reference (figure numbers are with reference to the said document).

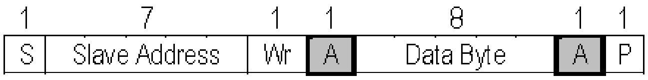


Figure 5-3: Send byte protocol

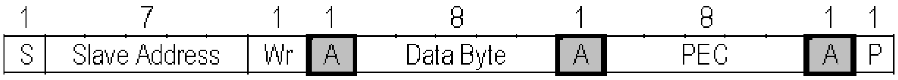


Figure 5-4: Send byte protocol with PEC

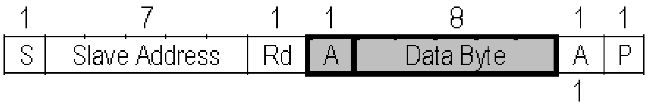


Figure 5-5: Receive byte protocol

Figure 5-6: Receive byte protocol with PEC

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command Code	A	Data Byte	A	P

Figure 5-7: Write byte protocol

1	7	1	1	8	1	8	1	8	1	1
S	Slave Address	Wr	A	Command Code	A	Data Byte Low	A	Data Byte High	A	P

Figure 5-8: Write Word Protocol

1	7	1	1	8	1	8	1	8	1	1
S	Slave Address	Wr	A	Command Code	A	Data Byte	A	PEC	A	P

Figure 5-9: Write byte protocol with PEC

1	7	1	1	8	1	8	1	8	1	
S	Slave Address	Wr	A	Command Code	A	Data Byte Low	A	Data Byte High	A	...

8	1	1
PEC	A	P

Figure 5-10: Write Word Protocol with PEC

1	7	1	1	8	1	1	7	1	1	8	1	
S	Slave Address	Wr	A	Command Code	A	S	Slave Address	Rd	A	Data Byte Low	A	...

8	1	1
Data Byte High	A	P
	1	

Figure 5-13: Read word protocol

1	7	1	1	8	1	1	7	1	1	8	1	
S	Slave Address	Wr	A	Command Code	A	S	Slave Address	Rd	A	Data Byte Low	A	...

8	1	8	1	1
Data Byte High	A	PEC	A	P
			1	

Figure 5-14: Read word protocol with PEC

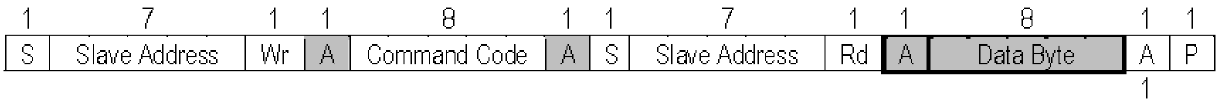


Figure 5-11: Read Byte Protocol

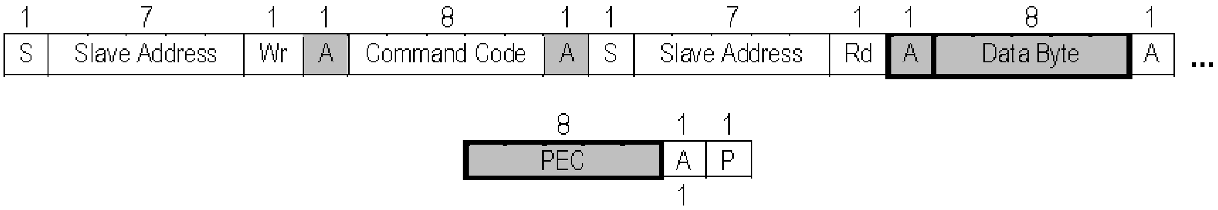


Figure 5-12: Read byte protocol with PEC

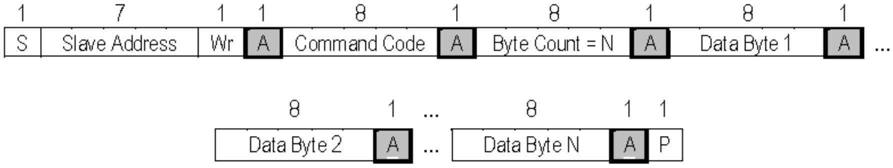


Figure 5-17: Block Write

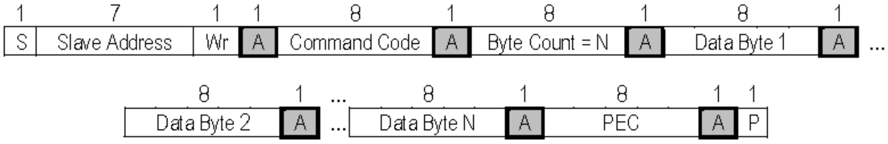


Figure 5-18: Block Write with PEC

Slave Device Addressing

This protocol's addressing system is not dynamic, and is based only on the three input signals A0, A1, and A2.

The iMP valid addresses are:

A2	A1	A0	Write Address	Read Address
0	0	0	30	31
0	0	1	32	33
0	1	0	34	35
0	1	1	36	37
1	0	0	38	39
1	0	1	3A	3B
1	1	0	3C	3D
1	1	1	3E	3F

Table 17. iMP Device Addressing

This protocol also supports the General Call address (00h) for write operation only.

Optional SMBus Signals

This protocol does not use any of the Optional SMBus signals.

9.3 PMBus Compliance

Command Error

This protocol does not support NACKing the command code or data bytes as they are being received. For any command error, the CML fault bit in the STATUS_BYTE register is set. The STATUS_CML register is not supported however. To see the detailed description of the fault, the CASE_FAULT_BYTE (Manufacturer Specific) register must be read. It contains the Command Error and the Command Disabled flags which describes the kind of communication error encountered.

Control Signal

The control signal in the iMP Power Supply unit is replaced by two signals, the INHIBIT_ENABLE_0 and the INHIBIT_ENABLE_1 signals. The logic of these signals are controlled by the ON_OFF_CONFIG register bit1 (this bit is originally the Control Pin Polarity flag, in this protocol however this is the Global Inhibit Flag). Depending on the value of this bit, the logic of these signals are described below:

INHIBIT_ENABLE_0	**INHIBIT_ENABLE_1**	**GLOBAL INHIBIT = TRUE**	**GLOBAL INHIBIT = FALSE**
0	0	Modules ON	Modules OFF
0	1	Modules OFF	Modules ON
1	0	Modules ON	Modules OFF
1	1	Modules ON	Modules OFF

Table 18. Control Signal

Parameter Reading

Due to the complexity of the internal communication system of the power supply, not all parameters that can be written is readable. This is specially the case for commands targeted to an individual module, including the PMbus standard commands TON_DELAY, VOUT_COMMAND, and IOUT_OC_FAULT_LIMIT. Since the Page register may change, the previous parameter may not be applicable anymore.

Memory Model

There are four types of memory locations from which the configuration of the PSU can be defined. They are the Operating Memory, the User Configuration Memory, the Factory Default Configuration Memory, and the Firmware Default Configuration Memory locations.

Operating Memory (R/W, Volatile)

The settings that are used by the PSU are stored in a volatile memory location called Operating Memory. This memory location is the working memory of the PSU. Upon start-up, previously defined settings are loaded into the Operating Memory from either of three nonvolatile memory locations (see succeeding items below). Configuration commands only affect this memory area. Although the contents of this memory space are not saved at turnoff, they can be saved in either the User or Factory Default Configuration Memory locations using the commands STORE_DEFAULT_ALL, RESTORE_DEFAULT_ALL, STORE_USER_ALL, and RESTORE_USER_ALL.

User Configuration Memory (R/W, Non-volatile)

This memory location contains the main configuration setup that will be loaded at the Operating Memory every time the PSU is powered. If any error is detected on this storage area, the User Configuration Error flag is set in the CASE_FAULT_BYTE, and the CML fault bit in the STATUS_BYTE register is set. This memory location is non-volatile.

Factory Default Configuration Memory (R/W, Non-volatile)

This memory location contains the factory configuration setup that will only be loaded at the Operating Memory if an error on the User Configuration Memory is detected upon power-up. If any error is detected on this storage area, the Default Configuration Error flag is set in the CASE_FAULT_BYTE, and the CML fault bit in the STATUS_BYTE register is set. This memory location is non-volatile.

Firmware Default Configuration Memory (Read-Only, Non-volatile)

This memory location is embedded on the firmware and is only loaded at the Operating Memory if both the User Configuration Memory and the Default Configuration Memory encountered errors upon power-up. This memory location is non-volatile.

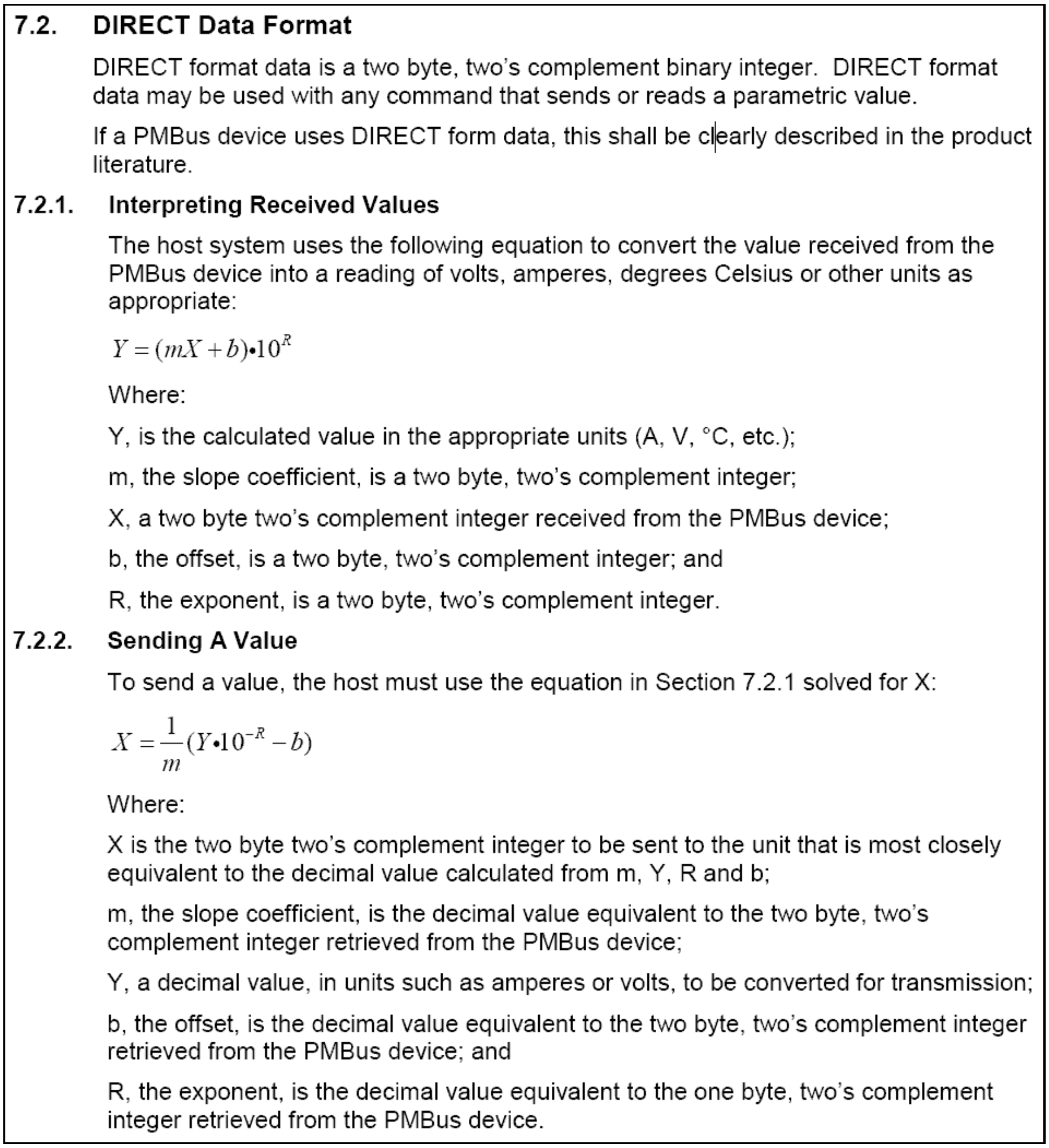
Status Registers

This protocol will use only the STATUS_BYTE register. In conjunction with this, there is also a CASE_FAULT_BYTE (Manufacturer Specific), a MODULE_COMMUNICATION_ERROR_BYTE (Manufacturer Specific), and a CASE_STATUS_BYTE (Manufacturer Specific) register. Only the CASE_STATUS_BYTE register is real-time, since the others requires the CLEAR_FAULTS command to reset the flags.

Data Formats

This protocol will use only the Direct Data Format for all parameters except for the Case Temperature related parameters, which use a manufacturer-specific data format that has a resolution of 0.25 degrees Celsius. Refer to each parameter's associated commands for the details on the data format. Access to the coefficients is not supported as the data format is fixed and already described in the specifications. This format is also applicable to the Output Voltage Related Parameters. Shown below is an excerpt from the PMBus™ Power System Management Protocol Specification, Part II – Command Language, Revision 1.0 document for easy reference (section numbers are with reference to the said document).

From Reference 2 - PMBus™ Power System Management Protocol Specification, Part II



7.2. DIRECT Data Format

DIRECT format data is a two byte, two's complement binary integer. DIRECT format data may be used with any command that sends or reads a parametric value.

If a PMBus device uses DIRECT form data, this shall be clearly described in the product literature.

7.2.1. Interpreting Received Values

The host system uses the following equation to convert the value received from the PMBus device into a reading of volts, amperes, degrees Celsius or other units as appropriate:

$$Y = (mX + b)\bullet 10^{R}$$

Where:

Y, is the calculated value in the appropriate units (A, V, °C, etc.);

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

X, a two byte two's complement integer received from the PMBus device;

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

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

7.2.2. Sending A Value

To send a value, the host must use the equation in Section 7.2.1 solved for X:

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

Where:

X is the two byte two's complement integer to be sent to the unit that is most closely equivalent to the decimal value calculated from m, Y, R and b;

m, the slope coefficient, is the decimal value equivalent to the two byte, two's complement integer retrieved from the PMBus device;

Y, a decimal value, in units such as amperes or volts, to be converted for transmission;

b, the offset, is the decimal value equivalent to the two byte, two's complement integer retrieved from the PMBus device; and

R, the exponent, is the decimal value equivalent to the one byte, two's complement integer retrieved from the PMBus device.

Figure 10. Direct Data Format

9.4 Supported PMBus Standard Register

Listed below are the PMBus standard registers that are supported by the PSU. Details are described in the PMBus™ Power System Management Protocol Specification, Part II – Command Language, Revision 1.0 document, but excerpts are shown for easy reference (table numbers are with reference to the said document).

STATUS_BYTE

This device uses the PMBus standard STATUS_BYTE register and all applicable flags. This register reflects all the other faults such that:

a. Any Module Over Voltage Protection fault sets the STATUS_BYTE VOUT_OV Flag
b. Any Module Over Current Protection fault sets the STATUS_BYTE IOUT_OC Flag
c. Any Module Over Temperature Protection fault, Case Over Temperature Protection fault, Case Over Temperature Protection warning, or Primary Over Temperature Protection warning, sets the STATUS_BYTE TEMPERATURE Flag
d. Calling a Disabled Command, Command Errors, Fault in the Default or User Memory Locations, or errors in any of the smart module internal UART communication buses, sets the STATUS_BYTE CML Flag
e. Any module UVP condition, module system fault, or an Over Power Limit Protection fault sets the STATUS_BYTE OTHER Flag
f. The following commands will set the BUSY flag while the command operation is ongoing:

VOUT_COMMAND
IOUT_OC_FAULT_LIMIT
TON_DELAY
EXTRACT_MODULE_VERSION
EXTRACT_MODULE_CONFIG_BYTES
IOUT_SENSOR_CALIBRATION
OVP_LIMIT_PERCENT
UVP_LIMIT_PERCENT
MODULE_OTP_LIMIT
MODULE_CONFIG_FLAGS
LOAD_PREDEFINED_SETTING
MODULE_VSCALE_CALIBRATION
DIRECT_MODULE_ACCESS

g. The same commands in item f above will be temporarily disabled while the BUSY flag is set, along with these commands:

READ_MODULE_VERSION
READ_MODULE_CONFIG_BYTES
READ_DIRECT_MODULE_ACCESS_REPLY

BIT	STATUS BIT NAME	DESCRIPTION
7	BUSY	A fault was declared because the device was busy and unable to response.
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	An output overvoltage fault has occurred.
4	IOUT_OC	An output overcurrent fault has occurred.
3	VIN_UV	An input undervoltage fault has occurred
2	TEMPERATURE	A temperature fault or warning has occurred.
1	CML	A communications, memory, or logic fault has occurred.
0	OTHER	A fault or warning not listed in bits [7:1] has occurred.

Table 19. STATUS_BYTE Register

WRITE_PROTECT

This device uses the PMBus standard WRITE_PROTECT register and all applicable flags.

DATA BYTE VALUE	DESCRIPTION
1000 0000	Disable all writes except to the WRITE_PROTECT command.
0100 0000	Disable all writes except to the WRITE_PROTECT, OPERATION, and PAGE commands.
0010 0000	Disable all writes except to the WRITE_PROTECT, OPERATION, and PAGE, ON_OFF_CONFIG and VOUT_COMMAND commands.
0000 0000	Enable writes to all commands.

Table 20. WRITE_PROTECT Register

OPERATION

This device uses only bit 7 of the PMBus standard OPERATION register. The other flags are disregarded.

BITS [7:6]	BITS [5:4]	BITS [3:2]	BITS [1:0]	Unit On or Off	Margin State
00	XX	XX	XX	Immediate Off (No Sequencing)	N/A
01	XX	XX	XX	Soft Off (With Sequencing)	N/A
10	00	XX	XX	On	Off
10	01	01	XX	On	Margin Low (Ignore Fault)
10	01	10	XX	On	Margin Low (Act on Fault)
10	10	01	XX	On	Margin High (Ignore Fault)
10	10	10	XX	On	Margin High (Act on Fault)

Table 21. OPERATION Register

ON_OFF_CONFIG

This device uses the PMBus standard ON_OFF_CONFIG register and all applicable flags except for bit 0. The Control Pin Polarity flag is used as the Global Inhibit flag instead.

BIT	PURPOSE	VALUE	DESCRIPTION
[7:5]		N/A	Reserved for Future Use.
4	Sets the default to either operate any time power is present or for the on/off to be controlled by CONTROL pin and serial bus commands	0	Unit powers up any time power is present regardless of state of the CONTROL pin.
		1	Unit does not power up until commanded by the CONTROL pin and OPERATION command (as programmed in bits [3:0].
3	Controls how the unit responds to commands received via the serial bus	0	Unit ignores the on/off portion of the OPERATION command from the serial bus
		1	To start, the unit requires that the on/off portion of the OPERATION command is instructing the unit to run.
2	Controls how the unit responds to the CONTROL pin	0	Unit ignores the CONTROL pin (on/off controlled only the OPERATION command).
		1	Unit requires the CONTROL pin to be asserted to start the unit. Depending on bit [3], the OPERATION command may also be required to instruct the device to start before the output is energized.
1	Polarity of the CONTROL pin	0	Active low (Pull pin low to start the unit).
		1	Active high (Pull pin high to start the unit).
0	CONTROL pin action when commanding the unit to turn off	0	Use the programmed turn off delay and fall time.
		1	Turn off the output and stop transferring energy to the output as fast as possible. The device's product literature shall specify whether or not the device sinks current to decrease the output voltage fall time.

Table 22. ON_OFF_CONFIG Register

9.5 Manufacturer Specific Register

PSU CONFIG

This register is stored in a non-volatile memory (User or Default Configuration Memory), and contains different configuration flags that controls the fan speed, external EEPROM access, and start-up state.

BIT	FLAG	R/W	DESCRIPTION
0	Fan Alarm Disabled	R/W	If this flag is set, fan fault detection is disabled (useful for PSU without fans).
1	Fan Off at Standby	R/W	At standby, fans operate at quiet mode by default. If this flag is set, the fans are turned off at standby mode instead.
2	Fan Direction Reversed Config	R/W	Fan speed is based on the hottest temperature reading, and reaches the maximum at 50 degrees Celsius. If this flag is set, reverse fan air flow is assumed and fan speed reaches the maximum at 40 degrees Celsius.
3	Full Speed Override*	R/W	If this flag is set, fan speed is set to maximum.
4	Half Speed Override*	R/W	If this flag is set, fan PWM duty is set to half.
5	Fan Voltage Override*	R Only	If this flag is set, fan speed is set according to a requested fan voltage. This flag can be controlled only by use of the VFAN_1 command.
6	FRU EEPROM Write Enabled	R/W	If this flag is set, external write to the FRU EEPROM is allowed.
7	Startup Operation Mode On	R/W	If this flag is set, the initial value of the OPERATION register is set to ON mode

*These fan overrides work in conjunction with the default temperature-based fan control. Whichever results to the highest fan speed will take control of the fans.

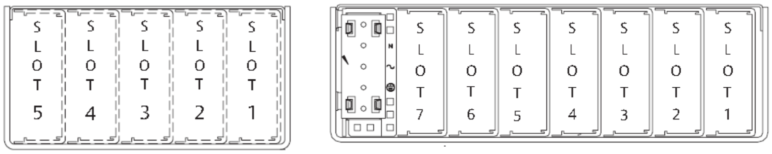
Table 23. PSU_CONFIG Register

ACTIVE SLOTS

This register contains the configuration of the module slots. The eight bits refer to the eight module slots. If a bit is set, then the corresponding slot contains a module (smart or otherwise). This register is stored in a non-volatile memory (User or Default Configuration Memory).

BIT	MODULE SLOT POSITION
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7*	---

*Unused bit.



iMP4

iMP8, iMP1

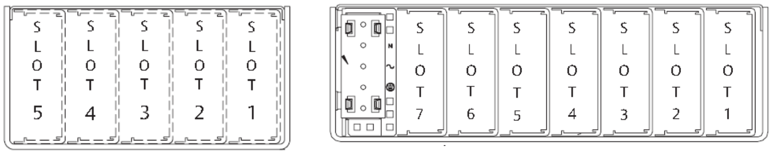
Figure 11. iMP Slot Configuration

SMART MODULES

This register contains the configuration of the smart modules. The eight bits refer to the eight module slots. If a bit is set, then the corresponding slot contains a smart module. This register is stored in a non-volatile memory (User or Default Configuration Memory).

BIT	MODULE SLOT POSITION
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7*	---

*Unused bit.



iMP4

iMP8, iMP1

Figure 12. iMP Slot Configuration

PSU_SETUP

This register contains flags regarding the current status of the PSU configuration access.

BIT	FLAG	R/W	DESCRIPTION
0	Configuration Data Status 0	R Only	These flags determine the state of the Operating Memory 11 - User Configuration Data was loaded 10 - Default Configuration Data was loaded 01 - Firmware Default Configuration Data was loaded 00 - Configuration Data in the Operating Memory was Updated
1	Configuration Data Status 1	R Only	
2	Factory Setup Enabled	R Only	CONFIDENTIAL: This flag determine if the Factory Setup Commands are enabled.

Table 24. PSU_SETUP Register

CASE_STATUS_BYTE

This register contains flags about the current status of the PSU. Note that the flags here represents the real-time status, and therefore does not require a separate command to be reset.

BIT	FLAG	R/W	DESCRIPTION
0	Inhibit_Enable_0	R Only	These flags mirrors the Control Signal Input state.
1	Inhibit_Enable_1	R Only	
2	AC OK	R Only	This flag reflects the state of the AC input.
3	Bulk OK	R Only	This flag reflects the state of the Bulk voltage.
4	Global DC OK	R Only	This flag reflects the state of all module outputs.
5	Fan1 OK	R Only	This flag reflects the state of the PSU Fan1.
6	Fan2 OK*	R Only	This flag reflects the state of the PSU Fan2*.
7	PS ON	R Only	This flag reflects the state of the PSU operation.

*For iMP8 and iMP1 series only.

Table 25. CASE_STATUS_BYTE Register

CASE_FAULT_BYTE

This register contains Case fault flags, and must be reset using the CLEAR_FAULTS command.

BIT	FLAG	R/W	DESCRIPTION
0	Case OTP	R Only	Case Over Temperature Limit was reached.
1	Case OTW	R Only	Case Temperature near the Over Temperature Limit.
2	Primary OTW	R Only	Primary Over Temperature Warning Limit was reached.
3	Over Power Fault	R Only	Smart module Power Limit was reached.
4	User Config Error	R Only	User Configuration Memory Data Corrupted.
5	Default Config Error	R Only	Default Configuration Memory Data Corrupted.
6	Disabled Command	R Only	Disabled Command was called by host.
7	Command Error	R Only	General Command error detected.

Table 26. CASE_FAULT_BYTE Register

MODULE_COMMUNICATION_ERROR_BYTE

This register contains fault flags regarding the internal UART-based module communication bus. The eight bits refer to the eight module slots. If a bit is set, then the attempted communication with the installed module in the corresponding slot failed. These flags must be reset using the CLEAR_FAULTS command.

BIT	MODULE SLOT POSITION
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7*	---

*Unused bit.

MODULE_STATUS_FLAGS

This register contains status and fault flags with respect to the module referenced by the current Page. Note that the flags here represents the real-time status, and therefore does not require a separate command to be reset.

BIT	FLAG	R/W	DESCRIPTION
0	Output Enabled	R Only	This flag reflects the state of the module’s operation.
1	UVP Fault	R Only	Module Under voltage condition exists.
2	DC OK	R Only	Module’s output is not within regulation.
3	OCP Fault	R Only	Module Over current fault exists.
4	OTP Fault	R Only	Module Over Temperature condition exists.
5	OTP warning	R Only	Module’s Temperature is near the Temperature Limit within 5°C.
6	OVP Fault	R Only	Module Over voltage fault detected.
7	System Fault	R Only	General module fault detected.

Table 27. MODULE_STATUS_FLAGS Register

MODULE_POWER_VOLTAGE_RANGE_CODE

This register is stored in the non-volatile memory of the module referenced by the Page, and contains information regarding the rated power of the said module as well as the voltage range of the output.

BIT	FLAG	R/W	DESCRIPTION
3-0	Voltage Range	R Only	These bits describe the range of the module's output voltage. 0000 – 2V to 5.5V 0001 – 6V to 12V 0010 – 14V to 20V 0011 – 24V to 36V 0100 – 42V to 60V 0101 – 5V (Fixed) 0110 – 2V to 6V 0111 – 12V to 15V 1000 – 24V to 28V
7-4	Power	R Only	These bits describe the rated power of the module. 0000 – 210W 0001 – 360W 0010 – 144W 0011 – 600W 0100 – 750W 0101 – 1500W

Table 28. MODULE_POWER_VOLTAGE_RANGE_CODE Register

MODULE_CONFIG_FLAGS

This register is stored in the non-volatile memory of the module referenced by the Page, and contains different configuration flags that control certain module operation.

BIT	FLAG	R/W	DESCRIPTION
0	Inhibit High-Asserted	R/W	If this flag is set, the module output is disabled if the inhibit signal is high.
1	Foldback OCP Mode	R/W	If this flag is set, the module's OCP mode is Fold Back, else the OCP mode is Constant Current.
2	UART Mode	R Only	This is always set.

Table 29. MODULE_CONFIG_FLAGS Register

HARDWARE_CODE

This register is stored in the non-volatile memory of the iMP case, and contains detailed information regarding the hardware model.

BIT	FLAG	R/W	DESCRIPTION
3-0	Model Option	R Only	0000 – Standard 0001 – Medicals 0010 – External Fan 0011 – Reverse Fan 0100 – IEC Option
7-4	Model Code	R Only	0000 – (Unused) 0001 – iMP1 0010 – iVS1 0011 – iVS3 0100 – iMP4 0101 – (Reserved) 0110 – iVS6 0111 – (Reserved) 1000 – iMP8 1001 – iVS8 1010 – iVS8H

Table 30. HARDWARE_CODE Register

9.6 Command / Response Packet

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
ID	Command Index	Command Function	Slave Device Address	PMBus Command Code	No. of Bytes	PEC Enable	Data Byte

Command Packet

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
ID	Command Index	Command Function	Error Code	Data Byte LSB	Data Byte MSB

Response Packet

9.7 Supported Standard PMBus Commands

Command Code	Command Name	SMBus Transaction Type	No. of Data Bytes
00h	PAGE	R/W Byte	1
01h	OPERATION	R/W Byte	1
02h	ON_OFF_CONFIG	R/W Byte	1
03h	CLEAR_FAULTS	Send Byte	0
10h	WRITE_PROTECT	R/W Byte	1
11h	STORE_DEFAULT_ALL	Send Byte	0
12h	RESTORE_DEFAULT_ALL	Write Byte	1
15h	STORE_USER_ALL	Send Byte	0
16h	RESTORE_USER_ALL	Write Byte	1
20h	VOUT_MODE	R/W Byte	1
21h	VOUT_COMMAND	R/W Word	2
3Ah	VFAN_1	R/W Word	2
4Fh	OT_FAULT_LIMIT	R/W Word	2
51h	OT_WARN_LIMIT	R/W Word	2
60h	TON_DELAY	R/W Word	2
78h	STATUS_BYTE	Read Byte	1
88h	READ_VIN	Read Word	2
89h	READ_IIN	Read Word	2
8Bh	READ_VOUT	Read Word	2
8Ch	READ_IOUT	Read Word	2
8Dh	READ_TEMPERATURE1	Read Word	2
8Eh	READ_TEMPERATURE2	Read Word	2
8Fh	READ_TEMPERATURE3	Read Word	2
90h	READ_FAN_SPEED_1	Read Word	2
91h	READ_FAN_SPEED_2	Read Word	2
98h	PMBUS_REVISION	Read Byte	1

Table 31. Summary of Supported PMBus Commands

Supported PMBus Standard Commands

PMBus Command	**PAGE**
Command Code	**00h**
Transaction Type	R/W Byte
No. of Data Bytes	1
Description	The iMP PSU can support up to eight different outputs. An internal register serves as an index that can be used by many commands to distinguish between the Modules. This is the Page register which can be accessed through the Page command. The valid values for the Page register are from zero to seven (0 to 7). At power-up, this value is zero.

iMP Series Valid Range	
Slot No.	**Page Register**
1	00h
2	01h
3	02h
4	03h
5	04h
6	05h
7	06h

Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).

Example:	Set the new PAGE register to 02h to access module on slot 3.

RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte**
	3E	80	23	3E	00	01	00	02
in Decimal:	62,128,35,62,0,1,0,2,							

RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**
	1	80	24	00
in Decimal:	1,128,35,0,			

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">OPERATION</td></tr>
<tr><td>Command Code</td><td colspan="8">01h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">R/W Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command can be used to turn the unit on and off (if enabled in the ON_OFF_CONFIG register). Only bit 7 (0 = OFF, 1 = ON) is used by the PSU since Sequencing and Margining are not supported through this protocol. At power-up, the initial value of this register is controlled by the Startup Operation Mode On flag of the PSU_CONFIG register (bit 7).

Data Byte = 80h – PSU ON (default value)
Data Byte = 00h – PSU OFF

Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).</td></tr>
<tr><td>Example 1:</td><td colspan="8">Set the data byte of OPERATION to disable/inhibit all outputs (PSU OFF).</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>01</td><td>01</td><td>00</td><td>00</td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,35,62,1,1,0,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>00</td><td colspan="4"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,35,0,</td></tr>
<tr><td>Example 2:</td><td colspan="8">Set the data byte of OPERATION to enable/activate all outputs (PSU ON).</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>01</td><td>01</td><td>00</td><td>80</td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,35,62,1,1,0,128,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>00</td><td colspan="4"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,35,0,</td></tr>
</table>

Supported PMBus Standard Commands

PMBus Command	**ON_OFF_CONFIG**							
Command Code	**02h**							
Transaction Type	R/W Byte							
No. of Data Bytes	1							
Description	This command can be used to set how the PSU will be turned on, by enabling the OPERATION command or the Control Signals (INHIBIT_ENABLE_0 and INHIBIT_ENABLE_1), or both. The Control Signal polarity (Global Inhibit Flag is this case) can also be set here. The ON_OFF_CONFIG flag for turn off delay and fall time (bit 0) is disregarded. Data Byte = 1Eh (default value) Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).							
Example:	Read the ON_OFF_CONFIG register.							
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte**
	3E	80	24	3E	02	01	00	00
in Decimal:	62,128,36,62,2,1,0,0,							
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte**			
	1	80	24	00	1E			
in Decimal:	1,128,36,0,30,							

Supported PMBus Standard Commands

PMBus Command	**CLEAR_FAULTS**						
Command Code	**03h**						
Transaction Type	Send Byte						
No. of Data Bytes	0						
Description	This command is used to clear the fault flags set in STATUS_BYTE, CASE_FAULT_BYTE, and MODULE_COMMUNICATION_ERROR_BYTE. Note that if the fault condition still exists, the associated flag will be set again. This command will only affect the flags and not the fault condition itself. Note: Use command function 0x21 (Send Byte).						
Example:	Clear all fault flags in all FAULT registers.						
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	21	3E	03	00	00
in Decimal:	62,128,33,62,3,0,0,						
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**			
	1	80	21	00			
in Decimal:	1,128,33,0,						

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">WRITE_PROTECT</td></tr>
<tr><td>Command Code</td><td colspan="8">10h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">R/W Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">The WRITE_PROTECT command can be used to control access to the PSU. Upon power-up, all commands are read-only (as applicable) except for this one. The Write Protection Setting can be updated then to:

1. Lock the serial access by disabling all write commands even the WRITE_PROTECT command
2. Allow only the WRITE_PROTECT, OPERATION and PAGE write commands.
3. Allow only the WRITE_PROTECT, OPERATION, PAGE, ON_OFF_CONFIG and VOUT write commands.
4. Allow only the ON_OFF_CONFIG, VOUT and the WRITE_PROTECT write commands.
5. Lock the serial access only for the ON_OFF_CONFIG and VOUT write commands.
6. Allow all non-factory setup commands (CONFIDENTIAL: factory setup commands are also controlled by the Factory Setup Flag aside from the Write Protect setting).

Data Byte = 81h - Disable all commands except WRITE_PROTECT (default value)
Data Byte = 00h - Enable all commands

Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Unprotect or enable PSU for control access.</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>10</td><td>01</td><td>00</td><td>00</td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,35,62,16,1,0,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>00</td><td colspan="4"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,35,0,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">STORE_DEFAULT_ALL</td></tr>
<tr><td>Command Code</td><td colspan="8">11h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Send Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">0</td></tr>
<tr><td>Description</td><td colspan="8">This command will store the contents of the Operating Memory in the Default Configuration Memory location. This command requires that the Factory Setup Flag is set.
Note: Use command function 0x21 (Send Byte).</td></tr>
<tr><td>Example:</td><td colspan="8">Save the operating memory in the default configuration memory.</td></tr>
<tr><td rowspan="2">RS485 Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>21</td><td>3E</td><td>11</td><td>00</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,33,62,17,0,0,</td></tr>
<tr><td rowspan="2">RS485 Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>21</td><td>00</td><td colspan="4"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,33,0,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">RESTORE_DEFAULT_ALL</td></tr>
<tr><td>Command Code</td><td colspan="8">12h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Write Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command will restore the contents of the Default Configuration Memory location to the Operating Memory.

Note: Use command function 0x23 (Write Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Restore default memory settings to operating memory.</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>12</td><td>01</td><td>00</td><td>00</td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,35,62,18,1,0,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>00</td><td colspan="4"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,35,0,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">STORE_USER_ALL</td></tr>
<tr><td>Command Code</td><td colspan="8">15h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Send Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">0</td></tr>
<tr><td>Description</td><td colspan="8">This command will store the contents of the Operating Memory in the User Configuration Memory location.
Note: Use command function 0x21 (Send Byte).</td></tr>
<tr><td>Example:</td><td colspan="8">Save the operating memory in the user configuration memory.</td></tr>
<tr><td rowspan="2">RS485 Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>21</td><td>3E</td><td>15</td><td>00</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,33,62,21,0,0,</td></tr>
<tr><td rowspan="2">RS485 Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>21</td><td>00</td><td colspan="4"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,33,0,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">RESTORE_USER_ALL</td></tr>
<tr><td>Command Code</td><td colspan="8">16h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Write Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command will restore the contents of the User Configuration Memory location to the Operating Memory.

Note: Use command function 0x23 (Write Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Restore user memory settings to operating memory.</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>16</td><td>01</td><td>00</td><td>00</td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,35,62,22,1,0,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>00</td><td colspan="4"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,35,0,</td></tr>
</table>

Supported PMBus Standard Commands

PMBus Command	**VOUT_MODE**							
Command Code	**20h**							
Transaction Type	R/W Byte							
No. of Data Bytes	1							
Description	This command can be used to read the data format used by the PSU for output voltage related commands. Though the format is fixed (Direct Format), this command was supported for compliance purposes. Data Byte = 40h – direct format (default value) Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).							
Example:	Read the data format of output voltage related commands.							
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte**
	3E	80	24	3E	20	01	00	00
in Decimal:	62,128,36,62,32,1,0,0,							
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte**			
	1	80	23	00	40			
in Decimal:	1,128,36,0,64,							

Supported PMBus Standard Commands

PMBus Command	**VOUT_COMMAND**								
Command Code	**21h**								
Transaction Type	R/W Word								
No. of Data Bytes	2								
Description	This command is used to update the voltage of the output module referenced by the current Page. Direct Data Format: m = 1, b = 0, R = -2 (10mV resolution) Valid Range: Refer to Appendix C the output voltage adjustability range. Note: Use command function 0x23 (Write Word) or 0x24 (Read Word).								
Example:	Set the output voltage to 12V. Data format is 1200d = 4B0h.								
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte LSB**	**Data Byte MSB**
	3E	80	23	3E	21	02	00	B0	04
in Decimal:	62,128,35,62,33,2,0,176,4,								
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**					
	1	80	23	00					
in Decimal:	1,128,35,0,								

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="9">VFAN_1</td></tr>
<tr><td>Command Code</td><td colspan="9">3Ah</td></tr>
<tr><td>Transaction Type</td><td colspan="9">R/W Word</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="9">2</td></tr>
<tr><td>Description</td><td colspan="9">This command will set the Fan Override flag in the PSU_CONFIG register and overrides the PSU fan control with the requested fan voltage (only if this fan voltage is greater than the expected voltage of the fan control logic, which is based on the temperature data). If the parameter is zero, the Fan Override flag is disabled.
Direct Data Format: m = 1, b = 0, R = -2 (10mV resolution)
Valid Range: 6.5 to 12V, 0V to disable
Note: Use command function 0x23 (Write Word) or 0x24 (Read Word).</td></tr>
<tr><td>Example:</td><td colspan="9">Set the fan voltage to 12V level (maximum speed). Data format is 1200d = 4B0h.</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte LSB</td><td>Data Byte MSB</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>3A</td><td>02</td><td>00</td><td>B0</td><td>04</td></tr>
<tr><td>in Decimal:</td><td colspan="9">62,128,35,62,58,2,0,176,4,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="5"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>00</td><td colspan="5"></td></tr>
<tr><td>in Decimal:</td><td colspan="9">1,128,35,0,</td></tr>
</table>

Supported PMBus Standard Commands

PMBus Command	**OT_FAULT_LIMIT**								
Command Code	**4Fh**								
Transaction Type	R/W Word								
No. of Data Bytes	2								
Description	This command is used to set the Over Temperature Limit of the Case. If the value is less than the current OT_WARN_LIMIT, it will also set OT_WARN_LIMIT to this value. Data Format: 0.25 degree Celsius resolution Valid Range: 20 to 90 degree Celsius Note: Use command function 0x23 (Write Word) or 0x24 (Read Word).								
Example:	Set the OTP limit to 85°C. Data format is 85/0.25 = 340d = 154h.								
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte LSB**	**Data Byte MSB**
	3E	80	23	3E	4F	02	00	54	01
in Decimal:	62,128,35,62,79,2,0,84,1,								
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**					
	1	80	23	00					
in Decimal:	1,128,35,0,								

Supported PMBus Standard Commands

PMBus Command	**OT_WARN_LIMIT**
Command Code	**51h**
Transaction Type	R/W Word
No. of Data Bytes	2
Description	This command is used to set the Over Temperature Warning Limit of the Case. Data Format: 0.25 degree Celsius resolution Valid Range: 0 to current OT_FAULT_LIMIT value Note: Use command function 0x23 (Write Word) or 0x24 (Read Word).
Example:	Set the OTP warning limit to 80°C. Data format is 80/0.25 = 320d = 140h.

RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte LSB**	**Data Byte MSB**
	3E	80	23	3E	51	02	00	40	01
in Decimal:	62,128,35,62,81,2,0,64,1,								

RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**
	1	80	23	00
in Decimal:	1,128,35,0,			

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="9">TON_DELAY</td></tr>
<tr><td>Command Code</td><td colspan="9">60h</td></tr>
<tr><td>Transaction Type</td><td colspan="9">R/W Word</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="9">2</td></tr>
<tr><td>Description</td><td colspan="9">This command will set the turn on delay of the output module referenced by the current Page.

Direct Data Format: m = 1, b = 0, R = 0 (1ms resolution)
Valid Range: 0 to 255 ms

Note: Use command function 0x23 (Write Word) or 0x24 (Read Word).</td></tr>
<tr><td>Example:</td><td colspan="9">Set the new turn-on delay to 255ms. Data format is 255ms = 255d = FFh.</td></tr>
<tr><td rowspan="3">RS485
Command Packet:

in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte LSB</td><td>Data Byte MSB</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>60</td><td>02</td><td>00</td><td>FF</td><td>00</td></tr>
<tr><td colspan="9">62,128,35,62,96,2,0,255,0,</td></tr>
<tr><td rowspan="3">RS485
Response Packet:

in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="5"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>00</td><td colspan="5"></td></tr>
<tr><td colspan="9">1,128,35,0,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">STATUS_BYTE</td></tr>
<tr><td>Command Code</td><td colspan="8">78h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Read Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command is used to read the STATUS_BYTE of the PSU. This device uses the PMBus standard STATUS_BYTE register and all applicable flags. This register reflects all the other faults such that:

a. Any Module Over Voltage Protection fault sets the STATUS_BYTE VOUT_OV Flag;
b. Any Module Over Current Protection fault sets the STATUS_BYTE IOUT_OC Flag;
c. Any Module Over Temperature Protection fault, Case Over Temperature Protection fault, Case Over Temperature Protection warning, or Primary Over Temperature Protection warning, sets the STATUS_BYTE TEMPERATURE Flag;
d. Calling a Disabled Command, Command Errors, Fault in the Default or User Memory Locations, or errors in any of the smart module internal UART communication buses, sets the STATUS_BYTE CML Flag;
e. Any module UVP condition, module system fault, or an Over Power Limit Protection fault sets the STATUS_BYTE OTHER Flag;

Data Format: Refer to STATUS_BYTE register (Page 68).

Data Byte = 00h (default value)

Note: Use command function 0x24 (Read Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the status byte of the PSU.</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>78</td><td>01</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,36,62,120,1,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte</td><td colspan="3"></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>00</td><td>00</td><td colspan="3"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,36,0,0,</td></tr>
</table>

Supported PMBus Standard Commands

PMBus Command	**READ_VIN**							
Command Code	**88h**							
Transaction Type	Read Word							
No. of Data Bytes	2							
Description	This command returns the input AC RMS voltage of the PSU. Direct Data Format: m = 1, b = 0, R = -2 (10mV resolution) Note: Use command function 0x24 (Read Word).							
Example:	Read the input AC voltage of the PSU. Data received is 2E98h = 11928d = 119.28Vac.							
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	
	3E	80	24	3E	88	02	00	
in Decimal:	62,128,36,62,136,2,0,							
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte LSB**	**Data Byte MSB**		
	1	80	24	00	98	2E		
in Decimal:	1,128,36,0,152,46,							

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">READ_IIN</td></tr>
<tr><td>Command Code</td><td colspan="8">89h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Read Word</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">2</td></tr>
<tr><td>Description</td><td colspan="8">This command returns the input current of the PSU.
Direct Data Format: m = 1, b = 0, R = -2 (10mA resolution)
Note: Use command function 0x24 (Read Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the input AC current of the PSU. Data received is 033Dh = 829d = 8.29A.</td></tr>
<tr><td rowspan="2">RS485 Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>89</td><td>02</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,36,62,137,2,0,</td></tr>
<tr><td rowspan="2">RS485 Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte LSB</td><td>Data Byte MSB</td><td colspan="2"></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>00</td><td>3D</td><td>03</td><td colspan="2"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,36,0,61,3,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="7">READ_VOUT</td></tr>
<tr><td>Command Code</td><td colspan="7">8Bh</td></tr>
<tr><td>Transaction Type</td><td colspan="7">Read Word</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="7">2</td></tr>
<tr><td>Description</td><td colspan="7">This command returns the output voltage of the module referenced by the current Page.
Direct Data Format: m = 1, b = 0, R = -2 (10mV resolution)
Note: Use command function 0x24 (Read Word).</td></tr>
<tr><td>Example:</td><td colspan="7">Read the output voltage of module. Data received is 04AFh = 1199d = 11.99Vdc.</td></tr>
<tr><td rowspan="3">RS485
Command Packet:

in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>8B</td><td>02</td><td>00</td></tr>
<tr><td colspan="7">62,128,36,62,139,2,0,</td></tr>
<tr><td rowspan="3">RS485
Response Packet:

in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte LSB</td><td>Data Byte MSB</td><td></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>00</td><td>AF</td><td>04</td><td></td></tr>
<tr><td colspan="7">1,128,36,0,175,4,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>**PMBus Command**</td><td colspan="8">**READ_IOUT**</td></tr>
<tr><td>**Command Code**</td><td colspan="8">**8Ch**</td></tr>
<tr><td>**Transaction Type**</td><td colspan="8">Read Word</td></tr>
<tr><td>**No. of Data Bytes**</td><td colspan="8">2</td></tr>
<tr><td>**Description**</td><td colspan="8">This command returns the output current of the module referenced by the current Page.
Direct Data Format: m = 1, b = 0, R = -2 (10mA resolution)
Note: Use command function 0x24 (Read Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the output current module. Data received is 178Bh = 6027d = 60.27A.</td></tr>
<tr><td rowspan="2">**RS485
Command Packet:**</td><td>**Address**</td><td>**Command Index**</td><td>**Command Function**</td><td>**Device Address**</td><td>**PMBus Command**</td><td>**No. of bytes**</td><td>**PEC Enable**</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>8C</td><td>02</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,36,62,140,2,0,</td></tr>
<tr><td rowspan="2">**RS485
Response Packet:**</td><td>Address</td><td>Command Index</td><td>**Command Function**</td><td>**Error Code**</td><td>**Data Byte LSB**</td><td>**Data Byte MSB**</td><td colspan="2"></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>0</td><td>8B</td><td>17</td><td colspan="2"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,36,0,139,23,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="7">READ_TEMPERATURE_1</td></tr>
<tr><td>Command Code</td><td colspan="7">8Dh</td></tr>
<tr><td>Transaction Type</td><td colspan="7">Read Word</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="7">2</td></tr>
<tr><td>Description</td><td colspan="7">This command returns the temperature of the PSU Case.
Data Format: 0.25 degree Celsius resolution, 2’s Complement
Note: Use command function 0x24 (Read Word).</td></tr>
<tr><td>Example:</td><td colspan="7">Read the case temperature of PSU. Data received is 0079h = 121d = 121*0.25 = 30°C.</td></tr>
<tr><td rowspan="3">RS485 Command Packet:
in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>8D</td><td>02</td><td>00</td></tr>
<tr><td colspan="7">62,128,36,62,141,2,0,</td></tr>
<tr><td rowspan="3">RS485 Response Packet:
in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte LSB</td><td>Data Byte MSB</td><td></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>00</td><td>79</td><td>00</td><td></td></tr>
<tr><td colspan="7">1,128,36,0,121,0,</td></tr>
</table>

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">READ_TEMPERATURE_2</td></tr>
<tr><td>Command Code</td><td colspan="8">8Eh</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Read Word</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">2</td></tr>
<tr><td>Description</td><td colspan="8">This command returns the temperature of the PSU primary side.
Direct Data Format: m = 1, b = 0, R = 0 (1 degree Celsius resolution)
Note: Use command function 0x24 (Read Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the case primary heatsink temperature of PSU. Data received is 30h = 48d = 48°C.</td></tr>
<tr><td>RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td></td><td>3E</td><td>80</td><td>24</td><td>3E</td><td>8E</td><td>02</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,36,62,142,2,0,</td></tr>
<tr><td>RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte LSB</td><td>Data Byte MSB</td><td colspan="2"></td></tr>
<tr><td></td><td>1</td><td>80</td><td>24</td><td>00</td><td>30</td><td>00</td><td colspan="2"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,36,0,48,0,</td></tr>
</table>

Supported PMBus Standard Commands

PMBus Command	**READ_TEMPERATURE_3**						
Command Code	**8Fh**						
Transaction Type	Read Word						
No. of Data Bytes	2						
Description	This command returns the temperature of the module referenced by the current Page. Direct Data Format: m = 1, b = 0, R = 0 (1 degree Celsius resolution) Note: Use command function 0x24 (Read Word).						
Example:	Read the module temperature. Data received is 2Ah = 42d = 42°C.						
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	24	3E	8F	02	00
in Decimal:	62,128,36,62,143,2,0,						
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte LSB**	**Data Byte MSB**	
	1	80	24	00	2A	00	
in Decimal:	1,128,36,0,42,0,						

Supported PMBus Standard Commands

PMBus Command	**READ_FAN SPEED_1**						
Command Code	**90h**						
Transaction Type	Read Word						
No. of Data Bytes	2						
Description	This command returns the speed of the PSU Case fan1. Direct Data Format: m = 10, b = 0, R = 0 (10 RPM resolution) Note: Use command function 0x24 (Read Word).						
Example:	Read the speed of Fan 1. Data received is 01C6h = 454d = 454*10 = 4540 RPM.						
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	24	3E	90	02	00
in Decimal:	62,128,36,62,144,2,0,						
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte LSB**	**Data Byte MSB**	
	1	80	24	00	C6	01	
in Decimal:	1,128,36,0,198,1,						

Supported PMBus Standard Commands

PMBus Command	**READ_FAN SPEED_2**						
Command Code	**91h**						
Transaction Type	Read Word						
No. of Data Bytes	2						
Description	This command returns the speed of the PSU Case fan2. Direct Data Format: m = 10, b = 0, R = 0 (10 RPM resolution) Note: Use command function 0x24 (Read Word).						
Example:	Read the speed of Fan 2. Data received is 01C6h = 454d = 454*10 = 4540 RPM.						
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	24	3E	91	02	00
in Decimal:	62,128,36,62,145,2,0,						
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte LSB**	**Data Byte MSB**	
	1	80	24	00	C6	01	
in Decimal:	1,128,36,0,198,1,						

Supported PMBus Standard Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">PMBUS_REVISION</td></tr>
<tr><td>Command Code</td><td colspan="8">98h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Read Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command returns the PMBus revision for which this PSU is compliant.
Data Byte = 00000000 (default value)
<table>
<tr><td>Bits [7:5]</td><td>Part I Revision</td></tr>
<tr><td>0000</td><td>1.0</td></tr>
<tr><td>0001</td><td>1.1</td></tr>
</table>
<table>
<tr><td>Bits [4:0]</td><td>Part II Revision</td></tr>
<tr><td>0000</td><td>1.0</td></tr>
<tr><td>0001</td><td>1.1</td></tr>
</table>
Note: Use command function 0x24 (Read Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the PMBus revision of the PSU. Data received is 00h = 000000000 = Revision 1.0</td></tr>
<tr><td rowspan="2">RS485 Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>98</td><td>01</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,36,62,152,1,0,</td></tr>
<tr><td rowspan="2">RS485 Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte</td><td colspan="3"></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>00</td><td>00</td><td colspan="3"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,36,0,0,</td></tr>
</table>

9.8 Supported Manufacturer Specific Commands

Command Code	Command Name	SMBus Transaction Type	No. of Data Bytes
D0h	CASE_FIRMWARE_VERSION	Read Block	5
D2h	ACTIVE_SLOTS	R/W Byte	1
D3h	SMART_MODULES	R/W Byte	1
D4h	MODULE_AUTO_DETECT	Send Byte	0
D5h	PSU_CONFIG	R/W Byte	1
D6h	PSU_SETUP	Read Byte	1
D7h	TOTAL_POWER	Read Word	2
D8h	CASE_STATUS_BYTE	Read Byte	1
D9h	CASE_FAULT_BYTE	Read Byte	1
DAh	MODULE_COMMUNICATION_ERROR_BYTE	Read Byte	1
DBh	MODULE_STATUS_FLAGS	Read Byte	1
DEh	EXTRACT_MODULE_VERSION	Send Byte	0
DFh	READ_MODULE_VERSION	Read Block	4
E1h	OVP_LIMIT_PERCENT	Write Byte	1
E2h	UVP_LIMIT_PERCENT	Write Byte	1
E3h	MODULE_OTP_LIMIT	Write Byte	1
E4h	MODULE_CONFIG_FLAGS	Write Byte	1
E5h	LOAD_PREDEFINED_SETTING	Write Byte	1
E7h	MODULE_OPERATIONS	R/W Word	2
E9h	PSU_MONITOR	Read Block	17
EAh	MODULE_MONITOR	Read Block	8
EBh	OVER_POWER_LIMITS	R/W Block	5
ECh	OUTPUT_INDEX	R/W Word	2
EEh	OUTPUT_INDEX_AUTOSWITCHBACK_DELAY	R/W Byte	1

Table 32. Summary of Manufacturer Specific Commands

PMBus Command	**CASE_FIRMWARE_VERSION**
Command Code	**D0h**
Transaction Type	Read Block
No. of Data Bytes	5
Description	This command will return the Case Firmware Version information for both the Primary Microcontroller and Secondary Microcontroller Firmware. Data Format: Byte1 is the Byte Count equal to 4 Byte2 is the Primary Firmware Version Byte3 is the Secondary Firmware Major Version in BCD Byte4 is the Secondary Firmware Minor Version in BCD Byte5 is the Secondary Firmware Version Branch in BCD Note: Use command function 0x26 (Read Block).
Example:	Read the case firmware version of the PSU. Primary FW version = Byte2 = 06 = Version 6.0 Secondary FW version = Byte3, Byte 4, Byte 5 = 020100 = Version 2.01.00

RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	
	3E	80	26	3E	D0	05	00	
in Decimal:	62,128,38,62,208,5,0,							

RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte 1**	**Data Byte 2**	**Data Byte 3**	**Data Byte 4**	**Data Byte 5**
	1	80	26	00	04	06	02	01	00
in Decimal:	1,128,38,0,4,6,2,1,0,								

Manufacturer Specific Commands

PMBus Command	**ACTIVE_SLOTS**
Command Code	**D2h**
Transaction Type	R/W Byte
No. of Data Bytes	1
Description	This command will access the ACTIVE_SLOTS register. This register contains the configuration of the module slots. The eight bits refer to the eight module slots. If a bit is set, then the corresponding slot contains a module (smart or otherwise). This register is stored in a non-volatile memory (User or Default Configuration Memory). Data Byte = 11111111b – all slots are active (default value) Note: Use command function 0x24 (Read Byte/Word).
Example:	Read active slots that have installed modules in the PSU. Data received is 7Fh = 11111111b

Bit	**Module Slot Position**
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7*	---

* Unused slot

RS485 Command Packet:

Address	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
3E	80	24	3E	D2	01	00

in Decimal: 62,128,36,62,210,1,0,

RS485 Response Packet:

Address	**Command Index**	**Command Function**	**Error Code**	**Data Byte**
1	80	24	00	7F

in Decimal: 1,128,36,0,127,

Manufacturer Specific Commands

PMBus Command	**SMART_MODULES**
Command Code	**D3h**
Transaction Type	R/W Byte
No. of Data Bytes	1
Description	This command will access the SMART_MODULES register. This register contains the configuration of the smart modules. The eight bits refer to the eight module slots. If a bit is set, then the corresponding slot contains a smart module. This register is stored in a non-volatile memory (User or Default Configuration Memory).

Bit	**Module Slot Position**
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7*	---

* Unused slot

Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).

Example:	Read the register that smart modules installed. Data received is 24h = 100100b. Smart modules installed at slots 3 and 6.

RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	24	3E	D3	1	0
in Decimal:	62,128,36,62,211,1,0,						

RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte**
	1	80	24	0	24
in Decimal:	1,128,36,0,36,				

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">MODULE_AUTO_DETECT</td></tr>
<tr><td>Command Code</td><td colspan="8">D4h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Send Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">0</td></tr>
<tr><td>Description</td><td colspan="8">This command will initiate auto detection of the Smart Modules installed on the slots. This will automatically update the SMART_MODULES register, and set the appropriate flags in the ACTIVE_SLOTS register. Previously set flags in the ACTIVE_SLOTS register will remain set.
Note: Use command function 0x21 (Send Byte).</td></tr>
<tr><td>Example:</td><td colspan="8">Auto-detection of smart modules that are installed in the PSU.</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte</td></tr>
<tr><td>3E</td><td>80</td><td>21</td><td>3E</td><td>D4</td><td>00</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,33,62,212,0,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td></td><td colspan="3"></td></tr>
<tr><td>1</td><td>80</td><td>21</td><td>00</td><td></td><td colspan="3"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,33,0,</td></tr>
</table>

Manufacturer Specific Commands

PMBus Command	**PSU_CONFIG**						
Command Code	**D5h**						
Transaction Type	R/W Byte						
No. of Data Bytes	1						
Description	This command will access the PSU_CONFIG register. This register is stored in a non-volatile memory (User or Default Configuration Memory), and contains different configuration flags that controls the fan speed, external EEPROM access, and start-up state. Data Format: Refer to PSU_CONFIG register (Page 71). Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).						
Example:	Read the configuration of the PSU. Data received is C0h = 11000000b.						
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	24	3E	D5	01	00
in Decimal:	62,128,36,62,213,1,0,						
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte**		
	1	80	24	00	C0		
in Decimal:	1,128,36,0,192,						

Manufacturer Specific Commands

PMBus Command	**PSU_SETUP**						
Command Code	**D6h**						
Transaction Type	Read Byte						
No. of Data Bytes	1						
Description	This command will return the PSU_SETUP register. This register contains flags regarding the current status of the PSU configuration access. Data format: Refer to PSU_SETUP register (Page 72). Note: Use command function 0x24 (Read Byte/Word).						
Example:	Read the current status of the PSU configuration access. Data received is 03h = 11b.						
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	24	3E	D6	01	00
in Decimal:	62,128,36,62,214,1,0,						
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte**		
	1	80	24	00	03		
in Decimal:	1,128,36,0,3						

PMBus Command	**TOTAL_POWER**							
Command Code	**D7h**							
Transaction Type	Read Word							
No. of Data Bytes	2							
Description	This command will return the total input power of the PSU. Direct Data Format: m = 1, b = 0, R = 0 (1W resolution) Note: Use command function 0x24 (Read Byte/Word).							
Example:	Read the total input power of the PSU. Data received is 0196h = 406d = 406 Watts.							
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	
	3E	80	24	3E	D7	02	00	
in Decimal:	62,128,36,62,215,2,0,0,							
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte LSB**	**Data Byte MSB**		
	1	80	24	00	96	01		
in Decimal:	1,128,36,0,150,1,							

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">CASE_STATUS_BYTE</td></tr>
<tr><td>Command Code</td><td colspan="8">D8h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Read Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command will return the CASE_STATUS_BYTE. This register contains flags about the current status of the PSU. Note that the flags represents the real-time status, and therefore does not require a separate command to be reset.
Data Format: Refer to CASE_STATUS_BYTE register (Page 73).
Note: Use command function 0x24 (Read Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the flags about the current status of the PSU.</td></tr>
<tr><td rowspan="3">RS485 Command Packet:
in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>D8</td><td>01</td><td>00</td><td></td></tr>
<tr><td colspan="8">62,128,36,62,216,1,0,</td></tr>
<tr><td rowspan="3">RS485 Response Packet:
in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte</td><td colspan="3"></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>00</td><td>FD</td><td colspan="3"></td></tr>
<tr><td colspan="8">1,128,36,0,253,</td></tr>
</table>

PMBus Command	**CASE_FAULT_BYTE**						
Command Code	**D9h**						
Transaction Type	Read Byte						
No. of Data Bytes	1						
Description	This command will return the CASE_FAULT_BYTE. This register contains Case fault flags, and must be reset using the CLEAR_FAULTS command. Data Format: Refer to CASE_FAULT_BYTE register (Page 73). Data Byte: 00000000b – no fault (default value) Note: Use command function 0x24 (Read Byte/Word).						
Example:	Read the case fault flags. Data received is 00h = 00000000b. No fault reported.						
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	24	3E	D9	01	00
in Decimal:	62,128,36,62,217,1,0,						
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte**		
	1	80	24	00	00		
in Decimal:	1,128,36,0,0,						

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">MODULE_COMMUNICATION_ERROR_BYTE</td></tr>
<tr><td>Command Code</td><td colspan="8">DAh</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Read Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command will return the MODULE_COMMUNICATION_ERROR_BYTE. This register contains fault flags regarding the internal UART-based module communication bus. The eight bits refer to the eight module slots. If a bit is set, then the attempted communication with the installed module in the corresponding slot failed. These flags must be reset using the CLEAR_FAULTS command.

Data Format: Refer to MODULE_COMMUNICATION_ERROR_BYTE register (Page 74).

Note: Use command function 0x24 (Read Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the module fault flags. Data received is 00h = 00000000b. No fault reported.</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>DA</td><td>01</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,36,62,218,1,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte</td><td colspan="3"></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>00</td><td>00</td><td colspan="3"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,36,0,0,</td></tr>
</table>

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">MODULE_STATUS_FLAGS</td></tr>
<tr><td>Command Code</td><td colspan="8">DBh</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Read Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command will return the MODULE_STATUS_FLAGS referenced by the current Page. This register contains status and fault flags with respect to the module referenced by the current Page. Note that the flags represents the real-time status, and therefore does not require a separate command to be reset.
Data Format: Refer to MODULE_STATUS_FLAGS register (Page 74).
Note: Use command function 0x24 (Read Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the status of the module. Data received is 05h = 00000101b.</td></tr>
<tr><td rowspan="3">RS485 Command Packet:
in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>24</td><td>3E</td><td>DB</td><td>01</td><td>00</td><td></td></tr>
<tr><td colspan="8">62,128,36,62,219,1,0,</td></tr>
<tr><td rowspan="3">RS485 Response Packet:
in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte</td><td colspan="3"></td></tr>
<tr><td>1</td><td>80</td><td>24</td><td>00</td><td>05</td><td colspan="3"></td></tr>
<tr><td colspan="8">1,128,36,0,5,</td></tr>
</table>

Manufacturer Specific Commands

PMBus Command	**EXTRACT_MODULE_VERSION**							
Command Code	**DEh**							
Transaction Type	Send Byte							
No. of Data Bytes	0							
Description	This command will initiate reading of the version information from the module referenced by the current Page. Note: Use command function 0x21 (Send Byte).							
Example:	Access the module firmware version and module hardware type.							
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	
	3E	80	21	3E	DE	00	00	
in Decimal:	62,128,33,62,222,0,0,							
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**				
	1	80	21	00				
in Decimal:	1,128,33,0,0,							

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">READ_MODULE_VERSION</td></tr>
<tr><td>Command Code</td><td colspan="8">DFh</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Read Block</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">4 (including the Byte Count)</td></tr>
<tr><td>Description</td><td colspan="8">This command will return the extracted version information of the module referenced by the current Page.

Data Format:
Byte1 is the Byte Count equal to 3
Byte2 is the Module Firmware Major Version
Byte3 is the Module Firmware Minor Version
Byte4 is the MODULE_POWER_VOLTAGE_RANGE_CODE (Page 75)
<table>
<tr><th>Bit</th><th>Flag</th><th>Description</th></tr>
<tr><td>3-0</td><td>Voltage Range</td><td>0000 – 2V to 5.5V
0001 – 6V to 12V
0010 – 14V to 20V
0011 – 24V to 36V
0100 – 42V to 60V
0101 – 5V (Fixed)
0110 – 2V to 6V
0111 – 12V to 15V
1000 – 24V to 28V</td></tr>
<tr><td>7-4</td><td>Power</td><td>0000 – 210W
0001 – 360W
0010 – 144W
0011 – 600W
0100 – 750W
0101 – 1500W</td></tr>
</table>
Note: Use command function 0x26 (Read Block).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the module firmware version and module type. Data received are 03h, 22h, 41h.
Module Firmware Version = 0322h = Version 3.22
Module Hardware Code = 41h = 0100 0001b = 750W 6V-12V</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>26</td><td>3E</td><td>DF</td><td>04</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,38,62,223,4,0,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte 1</td><td>Data Byte 2</td><td>Data Byte 3</td><td>Data Byte 4</td></tr>
<tr><td>1</td><td>80</td><td>26</td><td>00</td><td>03</td><td>03</td><td>22</td><td>41</td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,38,0,3,3,34,65,</td></tr>
</table>

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">OVP_LIMIT_PERCENT</td></tr>
<tr><td>Command Code</td><td colspan="8">E1h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Write Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command will update the Over Voltage Protection Limit of the module referenced by the current Page. The limit is in percentage of the output voltage setting.
Data Format: Parameter Byte is in 1% resolution
Data Range: 101 to 120 %
Note: Use command function 0x23 (Write Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Set the OVP limit to 110% of module’s output voltage. Data format is 110d = 6Eh</td></tr>
<tr><td rowspan="2">RS485 Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>E1</td><td>01</td><td>00</td><td>6E</td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,35,62,225,1,0,110,</td></tr>
<tr><td rowspan="2">RS485 Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>00</td><td colspan="4"></td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,35,0,</td></tr>
</table>

Manufacturer Specific Commands

PMBus Command	**UVP_LIMIT_PERCENT**							
Command Code	**E2h**							
Transaction Type	Write Byte							
No. of Data Bytes	1							
Description	This command will update the Under Voltage Protection Limit of the module referenced by the current Page. The limit is in percentage of the output voltage setting. Data Format: Parameter Byte is in 1% resolution Data Range: 0 to 99 % Note: Use command function 0x23 (Write Byte/Word).							
Example:	Set the UVP limit to 95% of module's output voltage. Data format is 95d = 5Fh.							
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte**
	3E	80	23	3E	E2	01	00	5F
in Decimal:	62,128,35,62,226,1,0,95,							
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**				
	1	80	23	00				
in Decimal:	1,128,35,0,							

Manufacturer Specific Commands

PMBus Command	**MODULE_OTP_LIMIT**							
Command Code	**E3h**							
Transaction Type	Write Byte							
No. of Data Bytes	1							
Description	This command will update the Over Temperature Protection Limit of the module referenced by the current Page. Data Format: Parameter Byte is in 1 degree Celsius resolution Data Range: 20 to 125 degrees Celsius Note: Use command function 0x23 (Write Byte/Word).							
Example:	Set the OTP limit of module to 100°C. Data format is 100d = 64h.							
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte**
	3E	80	23	3E	E3	01	00	64
in Decimal:	62,128,35,62,227,1,0,100,							
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**				
	1	80	23	00				
in Decimal:	1,128,35,0,							

Manufacturer Specific Commands

<table>
<tr><td colspan="1">PMBus Command</td><td colspan="8">MODULE_CONFIG_FLAGS</td></tr>
<tr><td>Command Code</td><td colspan="8">E4h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">Write Byte</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">1</td></tr>
<tr><td>Description</td><td colspan="8">This command will update the MODULE_CONFIG_FLAGS of the module referenced by the current Page. This register is stored in the non-volatile memory of the module and contains different configuration flags that control certain module operation.

Data Format: Refer to MODULE_CONFIG_FLAGS register (Page 76).

<table>
<tr><th>Bit</th><th>Flag</th><th>Description</th></tr>
<tr><td>0</td><td>Inhibit High-Asserted</td><td>If this flag is set, the module output is disabled if the inhibit signal is high.</td></tr>
<tr><td>1</td><td>Foldback OCP Mode</td><td>If this flag is set, the module’s OCP mode is Fold Back, else the OCP mode is Constant Current.</td></tr>
<tr><td>2</td><td>UART Mode</td><td>This is always set.</td></tr>
</table>

Note: Use command function 0x23 (Write Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="8">Set module configuration to Constant Current (CC mode). Data format is 100b = 64h.</td></tr>
<tr><td rowspan="3">RS485
Command Packet:

in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>E4</td><td>01</td><td>00</td><td>64</td></tr>
<tr><td colspan="8">62,128,35,62,228,1,0,100,</td></tr>
<tr><td rowspan="3">RS485
Response Packet:

in Decimal:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="4"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>0</td><td colspan="4"></td></tr>
<tr><td colspan="8">1,128,35,0,</td></tr>
</table>

Manufacturer Specific Commands

PMBus Command	**LOAD_PREDEFINED_SETTING**
Command Code	**E5h**
Transaction Type	Write Byte
No. of Data Bytes	1
Description	This command will load an indexed pre-defined setting stored in the module referenced by the current Page. Data Format: Parameter Byte is the index to the pre-defined setting stored in the module Data Range: Refer to the module documentation for valid stored settings index

Byte Value	**Pre-defined Settings**
00h	Backup Setting
01h	Factory Preset 1
02h	Factory Preset 2
03h	Factory Preset 3
04h	Factory Preset 4
05h	Factory Preset 5
06h	Factory Preset 6
07h	Factory Preset 7

Note: Use command function 0x23 (Write Byte/Word).

Example:	Restore Factory Preset 2 to the module. Data format is 02h.

RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte**
	3E	80	23	3E	E5	01	00	02
in Decimal:	62,128,35,62,229,1,0,2							

RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**
	1	80	23	0
in Decimal:	1,128,35,0,			

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="9">MODULE_OPERATIONS</td></tr>
<tr><td>Command Code</td><td colspan="9">E7h</td></tr>
<tr><td>Transaction Type</td><td colspan="9">R/W Word</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="9">2</td></tr>
<tr><td>Description</td><td colspan="9">This command will forward to the module referenced by the current Page an Operation Request. Note that the case will not perform any verification if the request is valid or not. The details of the supported operations depends on the module. Please review module documentation for this. An I^{2}C Read with the said command code will return two bytes wherein Byte1 is the latest Operation type received, and Byte 2 is reply of the module (a value of FFh means there is no reply yet). Note that there is significant delay from issuing the Module Operations command to the time of the module’s reply. A fixed delay may be used or a repeated poll for the reply may be implemented.
Data Format: Byte1 is the Operation Type
Byte2 is the Operation Parameter (Write) or Operation Reply (Read)
<table>
<tr><th>Type</th><th>Description</th><th>Parameter</th><th>Reply</th><th>Details</th></tr>
<tr><td>00h</td><td>Output Off</td><td>Delay
255 ms</td><td>AAh</td><td>Clears the module software On/Off control flag which disables the output of the module .</td></tr>
<tr><td>01h</td><td>Output On</td><td>Delay
255 ms</td><td>55h</td><td>Sets the module software On/Off control flag which enables the output of the module .</td></tr>
<tr><td>02h</td><td>Read On/Off</td><td>(N/A)</td><td>AAh / 55h</td><td>Extracts the status of the module software On/Off control flag (AAh = Off, 55h = On).</td></tr>
<tr><td>03h</td><td>FW Branch</td><td>(N/A)</td><td>Branch</td><td>Extracts the firmware version branch of the module (00h = main, else it is a Modification branch.</td></tr>
</table>
Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).</td></tr>
<tr><td>Example:</td><td colspan="9">Turn OFF the output of the module with 0 ms delay. Data format is 00h.</td></tr>
<tr><td rowspan="2">RS485 Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td>Data Byte 1</td><td>Data Byte 2</td></tr>
<tr><td>3E</td><td>80</td><td>23</td><td>3E</td><td>E7</td><td>02</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>in Decimal:</td><td colspan="9">62,128,35,62,231,2,0,0,0,</td></tr>
<tr><td rowspan="2">RS485 Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="5"></td></tr>
<tr><td>1</td><td>80</td><td>23</td><td>0</td><td colspan="5"></td></tr>
<tr><td>in Decimal:</td><td colspan="9">1,128,35,0,</td></tr>
</table>

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="8">ELAPSED_TIME</td></tr>
<tr><td>Command Code</td><td colspan="8">E8h</td></tr>
<tr><td>Transaction Type</td><td colspan="8">R/W Block</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="8">4 (including the Byte Count)</td></tr>
<tr><td>Description</td><td colspan="8">This command will access the accumulated Elapsed Time of the PSU device’s operation. Write access to this data requires that the Factory Setup Flag is set.
Data Format: Byte1 is the Byte Count equal to 3 (succeeding three bytes are the 24-bit Elapsed Time data in 1 minute resolution)
Data Range: 0 to 16,777,215 minutes
Note: Use command function 0x25 (Write Block) or 0x26 (Read Block).</td></tr>
<tr><td>Example:</td><td colspan="8">Read the elapsed time of the PSU operation. Data received is 03h, 00h, 07h, A1h.
Elapsed time = 07A1h = 1,953 minutes.</td></tr>
<tr><td rowspan="2">RS485 Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td></td></tr>
<tr><td>3E</td><td>80</td><td>26</td><td>3E</td><td>E8</td><td>04</td><td>00</td><td></td></tr>
<tr><td>in Decimal:</td><td colspan="8">62,128,38,62,232,4,0,</td></tr>
<tr><td rowspan="2">RS485 Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte</td><td>Data Byte</td><td>Data Byte</td><td>Data Byte</td></tr>
<tr><td>1</td><td>80</td><td>26</td><td>0</td><td>03</td><td>00</td><td>07</td><td>A1</td></tr>
<tr><td>in Decimal:</td><td colspan="8">1,128,38,0,3,0,7,161,</td></tr>
</table>

Manufacturer Specific Commands

PMBus Command	**PSU_MONITOR**
Command Code	**E9h**
Transaction Type	Read Block
No. of Data Bytes	17 (including the Byte Count)
Description	This command will return all the commonly monitored Case data. All the individual data here can be read using separate commands, and has the same formats. Data Format: Byte1 is the Byte Count equal to 16 For the succeeding bytes, refer to the previous discussions on the formats of the individual reply data as referenced by the associated commands. Refer to below reply format:

Reply Byte	Description	Reference Command	Example Data Bytes	Example Values
1	Byte Count = 16		*10h*	*10*
2	Status Byte	STATUS_BYTE	*00h*	*00000000b*
3	Case Status Byte	CASE_STATUS_BYTE	*FDh*	*11111101b*
4-5	Input Voltage	READ_VIN	*2E0Ah*	*117.86 V*
6-7	Input Current	READ_IIN	*0002h*	*0.02 A*
8-9	Total Power	TOTAL_POWER	*0002h*	*2 .00 Watts*
10-11	PSU Case Temperature	READ_TEMPERATURE_1	*006Fh*	*27 °C*
12-13	PSU Case Primary Temp	READ_TEMPERATURE_2	*0026h*	*38 °C*
14-15	Fan 1 Speed	READ_FAN_SPEED_1	*0217h*	*53500 RPM*
16-17	Fan 2 Speed	READ_FAN_SPEED_2	*0217h*	*53500 RPM*

Note: Use command function 0x26 (Read Block).

Example:	Read the case status of the PSU. Data received are as follows: = 10h, 00h, FDh, 0Ah, 2Eh, 02h, 00h, 02h, 00h, 6Fh, 00h, 26h, 00h, 17h, 02h,17h, 02h.

RS485 Command Packet:

Address	Command Index	Command Function	Device Address	PMBus Command	No. of bytes	PEC Enable	
3E	80	26	3E	E9	11	0	

in Decimal: 62,128,38,62,233,17,0,

RS485 Response Packet:

Address	Command Index	Command Function	Error Code	Data Bytes
1	80	26	0	*Refer to above data*

in Decimal: 1,128,38,0,16,0,253,10,46,2,0,2,0,111,0,38,0,23,2,23,2,2

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="7">MODULE_MONITOR</td></tr>
<tr><td>Command Code</td><td colspan="7">EAh</td></tr>
<tr><td>Transaction Type</td><td colspan="7">Read Block</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="7">8 (including the Byte Count)</td></tr>
<tr><td>Description</td><td colspan="7">This command will return all the commonly monitored data of the module referenced by the Page. All the individual data here can be read using separate commands, and has the same formats.

Data Format: Byte1 is the Byte Count equal to 7
(for the succeeding bytes, refer to the previous discussions on the formats of the individual reply data as referenced by the associated commands)

<table>
<tr><th>Reply Byte</th><th>Description</th><th>Reference Command</th><th>Example Data Bytes</th><th>Example Values</th></tr>
<tr><td>1</td><td>Byte Count = 7</td><td></td><td><i>07h</i></td><td><i>7</i></td></tr>
<tr><td>2-3</td><td>Output Voltage</td><td>READ_VOUT</td><td><i>0254h</i></td><td><i>5.96 V</i></td></tr>
<tr><td>4-5</td><td>Output Current</td><td>READ_IOUT</td><td><i>0000h</i></td><td><i>0.00 A</i></td></tr>
<tr><td>6-7</td><td>Module Temperature</td><td>READ_TEMPERATURE</td><td><i>0022h</i></td><td><i>34 °C</i></td></tr>
<tr><td>8</td><td>Module Status Flags</td><td>MODULE_STATUS_FLAGS</td><td><i>05h</i></td><td><i>00000101b</i></td></tr>
</table>
Note: Use command function 0x26 (Read Block).</td></tr>
<tr><td>Example:</td><td colspan="7">Read the status of individual module. Data received are as follows:
= 07h, 54h, 02h, 00h, 00h, 22h, 00h, 05h</td></tr>
<tr><td rowspan="2">RS485
Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td></tr>
<tr><td>3E</td><td>80</td><td>26</td><td>3E</td><td>EA</td><td>8</td><td>0</td></tr>
<tr><td>in Decimal:</td><td colspan="7">62,128,38,62,234,8,0,</td></tr>
<tr><td rowspan="2">RS485
Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td colspan="3">Data Byte</td></tr>
<tr><td>1</td><td>80</td><td>26</td><td>0</td><td colspan="3"><i>Refer to above data</i></td></tr>
<tr><td>in Decimal:</td><td colspan="7">1,128,38,0,7,84,2,0,0,34,0,5,</td></tr>
</table>

Manufacturer Specific Commands

<table>
<tr><td>PMBus Command</td><td colspan="9">OVER_POWER_LIMITS</td></tr>
<tr><td>Command Code</td><td colspan="9">EBh</td></tr>
<tr><td>Transaction Type</td><td colspan="9">R/W Block</td></tr>
<tr><td>No. of Data Bytes</td><td colspan="9">5 (including the Byte Count)</td></tr>
<tr><td>Description</td><td colspan="9">This command will access the power limit settings for both low-line and high-line input voltage operations.

Data Format: Byte1 is the Byte Count equal to 4
Byte2 is the least significant byte of the Low Line Power Limit
Byte3 is the most significant byte of the Low Line Power Limit
Byte4 is the least significant byte of the High Line Power Limit
Byte5 is the most significant byte of the High Line Power Limit
Data Range: 0 to 65536 Watts

Note: Use command function 0x25 (Write Block) or 0x26 (Read Block).</td></tr>
<tr><td>Example:</td><td colspan="9">Read overpower limits range of the PSU.
Data received are 04h, 80h, 07h, 60h, 09h.
Low Line Power Limit = 0780h = 1920d = 1,920 Watts
High Line Power Limit = 0960h = 2400d = 2,400 Watts</td></tr>
<tr><td rowspan="2">RS485 Command Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Device Address</td><td>PMBus Command</td><td>No. of bytes</td><td>PEC Enable</td><td colspan="2"></td></tr>
<tr><td>3E</td><td>80</td><td>26</td><td>3E</td><td>EB</td><td>5</td><td>0</td><td colspan="2"></td></tr>
<tr><td>in Decimal:</td><td colspan="9">62,128,38,62,235,5,0,</td></tr>
<tr><td rowspan="2">RS485 Response Packet:</td><td>Address</td><td>Command Index</td><td>Command Function</td><td>Error Code</td><td>Data Byte 1</td><td>Data Byte 2</td><td>Data Byte 3</td><td>Data Byte 4</td><td>Data Byte 5</td></tr>
<tr><td>1</td><td>80</td><td>26</td><td>0</td><td>04</td><td>80</td><td>07</td><td>60</td><td>09</td></tr>
<tr><td>in Decimal:</td><td colspan="9">1,128,38,0,4,128,7,96,9</td></tr>
</table>

Manufacturer Specific Commands

PMBus Command	**OUTPUT_INDEX**						
Command Code	**ECh**						
Transaction Type	R/W Word						
No. of Data Bytes	2						
Description	This command will access the Output Index register. This register will determine the output of the modules in which the case will communicate with, and is zero (referring to the first output) at start-up. The second byte refers to the new smart modules flags (see SMART_MODULES, page 108) applicable to this output index value, since the new output index may not be applicable to some of the installed modules (a value of 2 is not applicable to dual-output modules). If the output index is set to zero, the second byte is disregarded since the SMART_MODULES register will be used instead. Data Format: Byte1 is the Output Index Byte2 is the Smart Modules flags Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).						
Example:	Read the index register of the module. Data received is 00h. Index is at the main output.						
RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**
	3E	80	24	3E	EC	02	00
in Decimal:	62,128,36,62,236,2,0,						
RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**	**Data Byte 1**	**Data Byte 2**	
	1	80	24	00	00	00	
in Decimal:	1,128,36,0,0,0,						

Manufacturer Specific Commands

PMBus Command	**OUTPUT_INDEX_AUTOSWITCHBACK_DELAY**
Command Code	**EEh**
Transaction Type	R/W Byte
No. of Data Bytes	1
Description	This command will access the Output Index Autoswitchback Delay Register. The output index autoswitchback delay register controls the amount of delay before switching back to output index 0 after it has been changed to a different value. Entering a value of 0 will make the autoswitchback delay equal to the default delay that is 30 Seconds. Setting bit 7 of the byte disables the autoswitchback. Direct Data Format: m = 1, b = 0, R = 0 (1 Second Resolution) Data Range: 0 to 128 Seconds Note: Use command function 0x23 (Write Byte/Word) or 0x24 (Read Byte/Word).
Example:	Set the index auto switchback delay to 60 seconds. Data format is 60d = 3Ch.

RS485 Command Packet:	**Address**	**Command Index**	**Command Function**	**Device Address**	**PMBus Command**	**No. of bytes**	**PEC Enable**	**Data Byte**
	3E	80	24	3E	**EE**	01	00	3C
in Decimal:	62,128,36,62,238,1,0,60,							

RS485 Response Packet:	**Address**	**Command Index**	**Command Function**	**Error Code**
	1	80	24	0
in Decimal:	1,128,36,0,			

Appendix A – List of Supported Standard PMBus Commands

Command Code	Command Name	SMBus Transaction Type	No. of Data Bytes
00h	PAGE	R/W Byte	1
01h	OPERATION	R/W Byte	1
02h	ON_OFF_CONFIG	R/W Byte	1
03h	CLEAR_FAULTS	Send Byte	0
10h	WRITE_PROTECT	R/W Byte	1
11h	STORE_DEFAULT_ALL	Send Byte	0
12h	RESTORE_DEFAULT_ALL	Write Byte	1
15h	STORE_USER_ALL	Send Byte	0
16h	RESTORE_USER_ALL	Write Byte	1
20h	VOUT_MODE	R/W Byte	1
21h	VOUT_COMMAND	R/W Word	2
3Ah	VFAN_1	R/W Word	2
4Fh	OT_FAULT_LIMIT	R/W Word	2
51h	OT_WARN_LIMIT	R/W Word	2
60h	TON_DELAY	R/W Word	2
78h	STATUS_BYTE	Read Byte	1
88h	READ_VIN	Read Word	2
89h	READ_IIN	Read Word	2
8Bh	READ_VOUT	Read Word	2
8Ch	READ_IOUT	Read Word	2
8Dh	READ_TEMPERATURE1	Read Word	2
8Eh	READ_TEMPERATURE2	Read Word	2
8Fh	READ_TEMPERATURE3	Read Word	2
90h	READ_FAN_SPEED_1	Read Word	2
91h	READ_FAN_SPEED_2	Read Word	2
98h	PMBUS_REVISION	Read Byte	1

Table 33. Summary of Supported PMBus Commands

Appendix B - Supported Manufacturer Specific Commands

Command Code	Command Name	SMBus Transaction Type	No. of Data Bytes
D0h	CASE_FIRMWARE_VERSION	Read Block	5
D2h	ACTIVE_SLOTS	R/W Byte	1
D3h	SMART_MODULES	R/W Byte	1
D4h	MODULE_AUTO_DETECT	Send Byte	0
D5h	PSU_CONFIG	R/W Byte	1
D6h	PSU_SETUP	Read Byte	1
D7h	TOTAL_POWER	Read Word	2
D8h	CASE_STATUS_BYTE	Read Byte	1
D9h	CASE_FAULT_BYTE	Read Byte	1
DAh	MODULE_COMMUNICATION_ERROR_BYTE	Read Byte	1
DBh	MODULE_STATUS_FLAGS	Read Byte	1
DEh	EXTRACT_MODULE_VERSION	Send Byte	0
DFh	READ_MODULE_VERSION	Read Block	4
E1h	OVP_LIMIT_PERCENT	Write Byte	1
E2h	UVP_LIMIT_PERCENT	Write Byte	1
E3h	MODULE_OTP_LIMIT	Write Byte	1
E4h	MODULE_CONFIG_FLAGS	Write Byte	1
E5h	LOAD_PREDEFINED_SETTING	Write Byte	1
E7h	MODULE_OPERATIONS	R/W Word	2
E9h	PSU_MONITOR	Read Block	17
EAh	MODULE_MONITOR	Read Block	8
EBh	OVER_POWER_LIMITS	R/W Block	5
ECh	OUTPUT_INDEX	R/W Word	2
EEh	OUTPUT_INDEX_AUTOSWITCHBACK_DELAY	R/W Byte	1

Table 34. Summary of Manufacturer Specific Commands

Appendix C - Output Voltage Adjustability Range

Voltage	Voltage Code	Single Output Module Code				Dual Output***		Triple Output			I^2C Adjustment Ranges****
		1	2	3	5^+	4	4	-	-	-	
2 V	A	35 A	60 A	150 A	—	10 A	10 A	—	—	2 A	**1.8 - 2.2**
2.2 V	B	35 A	60 A	150 A	—	10 A	10 A	—	—	2 A	**2.0 - 2.4**
3 V	C	35 A	60 A	150 A	—	10 A	10 A	—	—	2 A	**2.7 - 3.3**
3.3 V	D	35 A	60 A	150 A	—	10 A	10 A	—	—	2 A	**3.0 - 3.6**
5 V	E	35 A	60 A	150 A	—	10 A	10 A	—	—	2 A	**4.5 - 5.5**
5.2 V	F	35 A	60 A	144 A	—	10 A	10 A	—	—	2 A	**4.7 - 5.7**
5.5 V	G	34 A	58 A	136 A	—	10 A	10 A	—	—	2 A	**5.0 - 6.1**
6.0 V	H	23 A	42 A	97.5 A	140 A	10 A*	10 A*	—	—	2 A	**5.4 - 6.6**
8.0 V	I	20 A	36 A	84.4 A	140 A	10 A	4 A	1 A	1 A	1 A	**7.2 - 8.8**
10 V	J	18 A	32 A	75 A	140 A	10 A	4 A	1 A	1 A	1 A	**9.0 - 11.0**
11 V	K	17 A	31 A	68 A	136.3 A	10 A	4 A	1 A	1 A	1 A	**9.9 - 12.1**
12 V	L	17 A	30 A	62.5 A	125 A	10 A	4 A	1 A	1 A	1 A	**10.8 - 13.2**
14 V	M	14 A	21 A	53.5 A	107 A	9 A	4 A	1 A	1 A	1 A	**12.6 - 15.4**
15 V	N	14 A	20 A	50 A	100 A	8 A	4 A	1 A	1 A	1 A	**13.5 - 16.5**
18 V	O	11 A	19 A	41.6 A	83.3 A	—	—	—	0.5 A	0.5 A	**16.2 - 19.8**
20 V	P	10.5 A	18 A	37.5 A	75 A	—	—	—	0.5 A	0.5 A	**18.0 - 22.0**
24 V	Q	8.5 A	15 A	30 A	62.5 A	4 A	2 A	—	0.5 A	0.5 A	**21.6 - 26.4**
28 V	R	6.7 A	11 A	26.8 A	53.5 A	3 A	2 A	—	0.5 A	0.5 A	**25.2 - 30.8**
30 V	S	6.5 A	11 A	25 A	50 A	—	—	—	—	—	**27.0 - 33.0**
33 V	T	6.2 A	10.9 A	22.7 A	35.8 A	—	—	—	—	—	**29.7 - 36.3**
36 V	U	5.8 A	10 A	20.8 A	35.8 A	—	—	—	—	—	**32.4 - 39.6**
42 V	V	4.2 A	7.5 A	16 A	35.7 A	—	—	—	—	—	**37.8 - 46.2**
48 V	W	4.0 A	7.5 A	15.6 A	31.2 A	—	—	—	—	—	**43.2 - 52.8**
54 V	X	3.7 A	6.0 A	13.9 A	27.7 A	—	—	—	—	—	**48.6 - 59.4**
60 V	Y	3.5 A	6.0 A	12.5 A	25 A	—	—	—	—	—	**54.0 - 66.0**
Non-std**	Z	Special Voltage - Consult Factory for specifications									

*Note: Contact Factory for extended range down to 6 V.
**Increments of current not shown can be achieved by paralleling modules (add currents of each module selected).
***Total output power on dual module must not exceed 144 W.
**** For single output modules only.
+ Applicable for iMP1 only.

Table 35. Voltage Adjustability Range of iMP Module Series

WORLDWIDE OFFICES

Americas
2900 S.Diablo Way
Tempe, AZ 85282
USA
+1 888 412 7832

Europe (UK)
Waterfront Business Park
Merry Hill, Dudley
West Midlands, DY5 1LX
United Kingdom
+44 (0) 1384 842 211

Asia (HK)
14/F, Lu Plaza
2 Wing Yip Street
Kwun Tong, Kowloon
Hong Kong
+852 2176 3333

ARTESYN™
EMBEDDED TECHNOLOGIES

www.artesyn.com

For more information: **www.artesyn.com/power**
For support: **productsupport.ep@artesyn.com**