1.0 REVISION LEVEL

1.1 SRS REVISION HISTORY

REV.	ECO# / DATE	REVISION DESCRIPTION	AUTHOR	APPROVER
A	G20092609A	Initial Release	Marco Florcruz / Christine Tan	Richard Mina Vincent Vivar

1.2 PRODUCT REVISION HISTORY

SPEC NO.	REV.	DATE	CUST MODEL NO. OR P/N & REV.	MODEL REV.	REMARKS
			73-544-001/ 73-544-002	0A	Initial Release

1.3 DESIGN AUTHORITY < ASTEC INT'L LIMITED – PHIL BRANCH >

2.0 GENERAL DESCRIPTION & CONTENTS

2.1 TITLE

Software Requirement Specification For

73-544-001/73-544-002 iVS/iMP CAN/RS485 to I2C PROTOCOL

This specification covers the Software requirements
For a Non Isolated Communication Adapter, for iVS and iMP Series cases.

3.0 TABLE OF CONTENTS

4.0 SRS OVERVIEW

4.1 Purpose

This documents defines software requirement specification for product 73-544-001 and 73-544-002 for interfacing all iVS Cases (73-180-001i, 73-190-001i) and all iMP Cases (73-690-0001I, 73-540-0001I and 73-580-0001I) to CAN Bus and RS485 Bus.

4.2 Scope

The microcontroller must have the following functions:

- Convert I2C communication format to CAN Bus and RS485 Bus compatible format.

4.3 Definitions and Acronyms

Term	Definition
PS	Power Supply
PSU	Power Supply Unit
SMPS	Switched Mode Power Supply
UC	Microcontroller
IC	Integrated Circuit
UART	Universal Asynchronous Receiver Transmitter
CAN	Controller Area Network

TABLE 4.3

4.4 References

Description	Revision	Date	Remarks
QP3755 - ASTEC Software Configuration Management (SCM) Quality Procedure	02		
ASTEC AA24150L Schematic Diagram			
Microchip, Microcontroller PIC18F2580 Datasheet	DS41206A	11/17/03	
I2C Bus Specification 2.1			
SMBus Bus Specification 2.0			
Modbus Specification 1.1b			
CAN Specification 2.0			

TABLE 4.4

5.0 SOFTWARE REQUIREMENTS

5.1 Adapter Protocol Overview

This section provides the implementation of the RS485/CAN-to-I2C Adapter of the Adapter Protocol. 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).

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 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.

5.2 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 below illustrates this transaction.

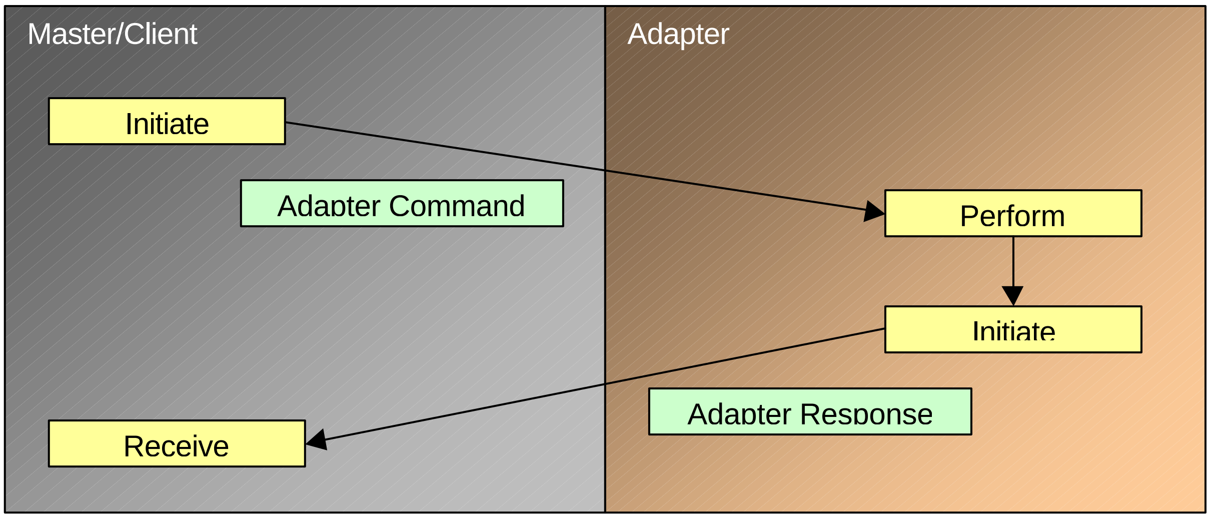

FIGURE 5.2 Adapter Protocol

5.3 Adapter Command and Response Packets

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

FIGURE 5.3-1 Adapter Command Packet

The Adapter Command Packet, as shown in Figure 5.3-1, 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 Error Code	Byte 4 to 67 Command Output

T

FIGURE 5.3-2 Adapter Response Packet

The Adapter Response Packet data frame, as shown in Figure 5.3-2, 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.

5.3.1 Command Index

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

Command Index	Command Type	Description
0x00	Adapter Control	Controls Adapter specific functions and Active Input Protocol control functions.
0x01–0x7F	Input Protocol Control	Controls Input Protocol specific functions.
0x80–0xFF	Output Protocol Control	Controls Output Protocol specific functions and for sending data through the Output Protocol.

TABLE 5.3.1 Command Index Grouping and Descriptions

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.

5.3.2 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 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 5.3.2 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.

5.4 Adapter Control Supported Commands

Table below 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
0x22–0xEF	Reserved	–	–
0xF0	Go into Bootloader Mode	0	–
0xFF	Adapter Reset	0	See Description

TABLE 5.4 Adapter Control Commands

5.4.1 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.

5.4.2 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 2.3.3.1 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.

5.4.3 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 2.3.3.1 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.

5.4.4 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 2.3.4.1 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.

5.4.5 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 2.3.4.1 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.

5.4.6 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.

*Not yet tested/for future use

5.4.7 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.

5.5 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.

5.6 0x01 Input Protocol–RS485 using Modbus

Input Protocol 0x01 is assigned to the RS485 bus using the Modbus protocol. RTU 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.

5.6.1 Server Address

Adapter Server Address when using this Input Protocol is determined by 3 Dip Switches to set the Server Address. Table 7 below shows the summary of the Dip Switch 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 7 Modbus Server Address Summary

5.6.2 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 6 below shows an illustration of the holding register mapping.

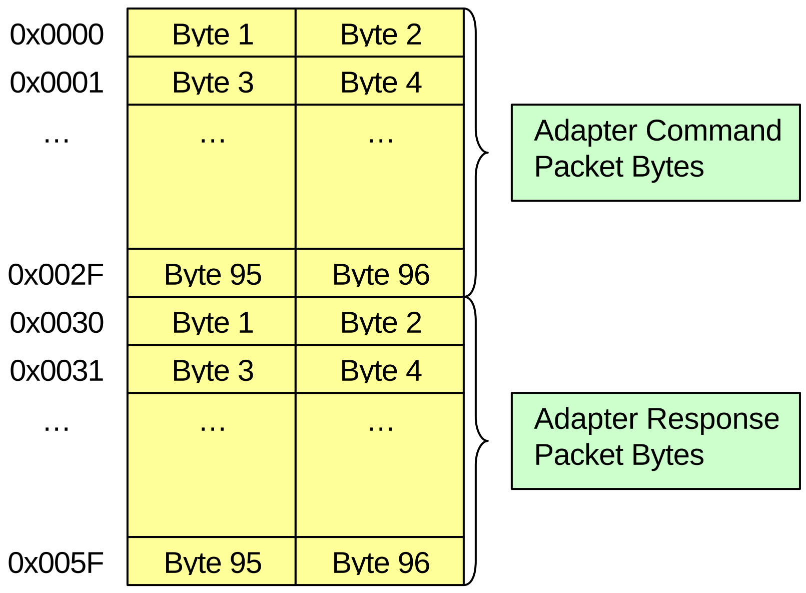

FIGURE 6 Modbus Holding Register Mapping

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.

5.6.2.1 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

*Not yet fully tested/for future use

5.6.3 Input Protocol Control Functions

Table 8 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 8 Supported RS485 using Modbus Functions

5.6.3.1 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	0x05–Not Applicable, None Output Protocol Function.

Error Codes	

5.6.3.2 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.

5.6.3.3 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).

5.6.3.4 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.

5.6.3.5 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).

5.6.3.6 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.

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

TABLE 5.6.4 RS485 Transfer Error Code Summary

5.7 0x02 Input Protocol–CAN using Modified Modbus

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 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.

5.7.1 Server Address

Adapter Server Address when using this Input Protocol is determined by 3 Dip Switches to set the Server Address. The Dip Switches 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 9 below shows the summary of the Dip Switch logic vs. Adapter Server Address. Note the 3-bit 0's pad on bits 8-10, as discussed in Section 2.3.5.3 above.

Server Address	A2	A1	A0

0x30 (0b0000110000)	0	0	0
0x32 (0b0000110010)	0	0	1
0x34 (0b0000110100)	0	1	0
0x36 (0b0000110110)	0	1	1
0x38 (0b0000111000)	1	0	0
0x3A (0b0000111010)	1	0	1
0x3C (0b0000111100)	1	1	0
0x3E (0b0000111110)	1	1	1

TABLE 5.7.1 Modbus
Server Address Summary

5.7.2 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 5.2.2 for Holding Register mapping details.

5.7.2.1 Supported Modbus Functions

CAN using modified Modbus Input Protocol supports the same Modbus Functions used for RS485 using Modbus Input Protocol. Refer to section 5.2.2.1 for supported Modbus function details.

5.7.3 Input Protocol Control Functions

Table 10 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 5.7.3 Supported CAN
using modified Modbus

5.7.3.1 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.

5.7.3.2 0x01 Get CAN Baud Rate

Parameter	None
Description	Returns the current 1 Byte code for CAN Baud. Possible Outputs will be: 0x01–10 Kbps (not tested) 0x02–20 Kbps (not tested) 0x03–50 Kbps (not tested) 0x04–125 Kbps 0x05–250 Kbps 0x06–500 Kbps 0x07–800 Kbps (not tested)
Output	Byte 1–Code for current CAN Baud Rate.
Additional Error Codes	None.

5.7.3.3 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 (not tested) 0x02–20 Kbps (not tested) 0x03–50 Kbps (not tested) 0x04–125 Kbps 0x05–250 Kbps 0x06–500 Kbps 0x07–800 Kbps (not tested) 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).

5.7.3.4 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.

5.7.3.5 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).

5.7.3.6 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.

5.7.4 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 11 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 5.7.4 CAN Transfer Error Code Summary

5.8 0x80 Output Protocol–I2C with SMBus Support

Table 11 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 5.8 Supported I2C with SMBus Output Protocol Functions

5.8.1 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.

5.8.2 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.

5.8.3 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).

5.8.4 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 5.8.15 for I2C transaction Errors.

5.8.5 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 5.8.15 for I2C transaction Errors.

5.8.6 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 5.8.15 for I2C transaction Errors.

5.8.7 0x21 SMBus Send Byte

Parameter	Byte 1–I2C Address. Byte 2–Data Byte to send. Byte 3–PEC Enable. (currently always disabled - not tested) 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 5.8.15 for I2C transaction Errors.

5.8.8 0x22 SMBus Receive Byte

Parameter	Byte 1–I2C Address. Byte 2–PEC Enable. (currently always disabled - not tested) 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 5.8.15 for I2C transaction Errors.

5.8.9 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. (currently always disabled - not tested) 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 5.8.15 for I2C transaction Errors.

5.8.10 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. (currently always disabled - not tested) 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 5.8.15 for I2C transaction Errors.

5.8.11 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. (currently always disabled - not tested) 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 5.8.15 for I2C transaction Errors.

5.8.12 0x26 SMBus Block Read

Parameter	Byte 1–I2C Address. Byte 2–Command Code. Byte 3–Data bytes to read Byte 4–PEC Enable. (currently always disabled - not tested) 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 5.8.15 for I2C transaction Errors.

5.8.13 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. (currently always disabled - not tested) 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 5.8.15 for I2C transaction Errors.

5.8.14 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.

5.8.15 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 11 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 14 I2C Transfer Error Code Summary

6.0 Hardware Interfaces

See below system overview for 73-544-001/73-544-002.

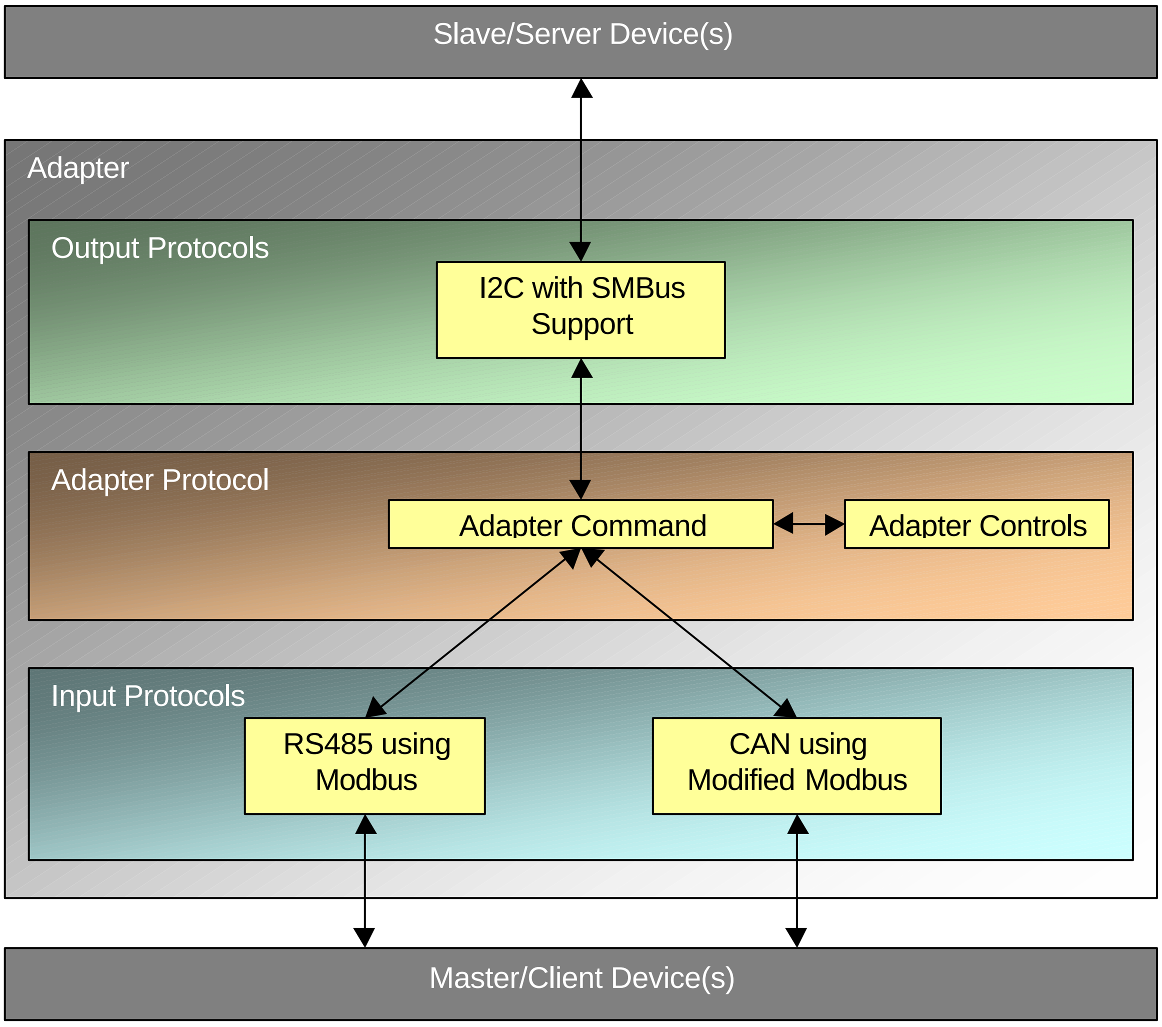

FIGURE 5 RS485/CAN –to –I2C Overview

Primary Micro controller Pin assignments and definitions

PIC18 F2580 Pin #	PIC18 F2580 Port	Signal Name	IO Type	IO Type	Description
1	RA2	A2	Input	Digital	Adapter Address pin 2
11	RC3	SCL	Output	Digital	I2C Clock Pin

12	RC4	SDA	Input/Output	Analog	I2C Data Pin
14	RC6	UART_TX	Output	Digital	UART transmit pin
15	RC7	UART_RX	Input	Digital	UART receive pin
20	RB2	CAN_TX	Output	Digital	CAN transmit pin
21	RB3	CAN_RX	Output	Digital	CAN receive pin
24	RB6	PGC	Input/Output	Digital	Programming pins
23	RB5	LED	Output	Digital	Controls the LED
25	RB7	PGD	Input/Output	Digital	Programming pins
26	RE3	VPP	Input/Output	Digital	Programming pins
27	RA0	A0	Input	Digital	Adapter Address pin 0
28	RA1	A1	Input	Digital	Adapter Address pin 1

TABLE 5.1

Device List

No.	Device Name	ASTEC PN	Description	Remarks
1	PIC18F2580	22166014790		

TABLE 5.2

6.1 Hardware Signal Definition

6.1.1 **Name of Signal**: **A0, A1, A2**

Description of purpose: 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: NA

6.1.2 **Name of Signal**: **SCL, SDA**

Description of purpose: 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 400kHz

6.1.3 **Name of Signal**: **UART_TX, UART_RX**

Description of purpose: 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

6.1.4 **Name of Signal**: **CAN_TX, CAN_RX**

Description of purpose: 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

6.1.3 **Name of Signal**: **LED (for 73-544-001 only)**

Description of purpose: 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: NA

6.2 Function Summary

No.	Function Description	Sub-Function Description	Referenced Signals	Remarks
6.2.1	CAN communication	5.1.1 CAN transmit	CAN_TX	Refer to software requirements
		5.1.2 CAN receive	CAN_RX	Refer to software requirements
6.2.2	UART communication	5.2.1 UART transmit	UART_TX	Refer to software requirements
		5.2.2 UART receive	UART_RX	Refer to software requirements
6.2.3	I2C communication	5.2.1 I2C send	SDA, SCL	Refer to software requirements
		5.2.2 I2C receive	SDA, SCL	Refer to software requirements
6.2.4	Control LED	NA	LED	- Should be ON when the unit is powered on.

				- Blinking fast when an error is received

Table 5.3 Function Summary Table

6.3 Assumptions and Dependencies

N/A

6.4 Design Constraints

Program is patterned to work with inputs and outputs only indicated in section 5.0 software requirements.