Introduction

The Master Interface Module allows users to interface slave cards to a PC using one of three interfaces, USB, Ethernet or RS232. The Module has a high speed 400kHz I²C bus, 23-channels of bidirectional digital I/O and on-board 256KBit EEPROM.

Key Features

Multiple Interfaces (Ethernet, USB, RS232)
One Interface Controls All Desired Slave Boards
Wide Voltage Input
Robust, Flexible Design
Multiple Connection Methods (Piggyback, Ribbon)
Low Cost
Small Size
Multiple Driver Downloads (LabView, Python, .NET)
All Components Fully Traceable

Two different build options are available to reduce cost to users.

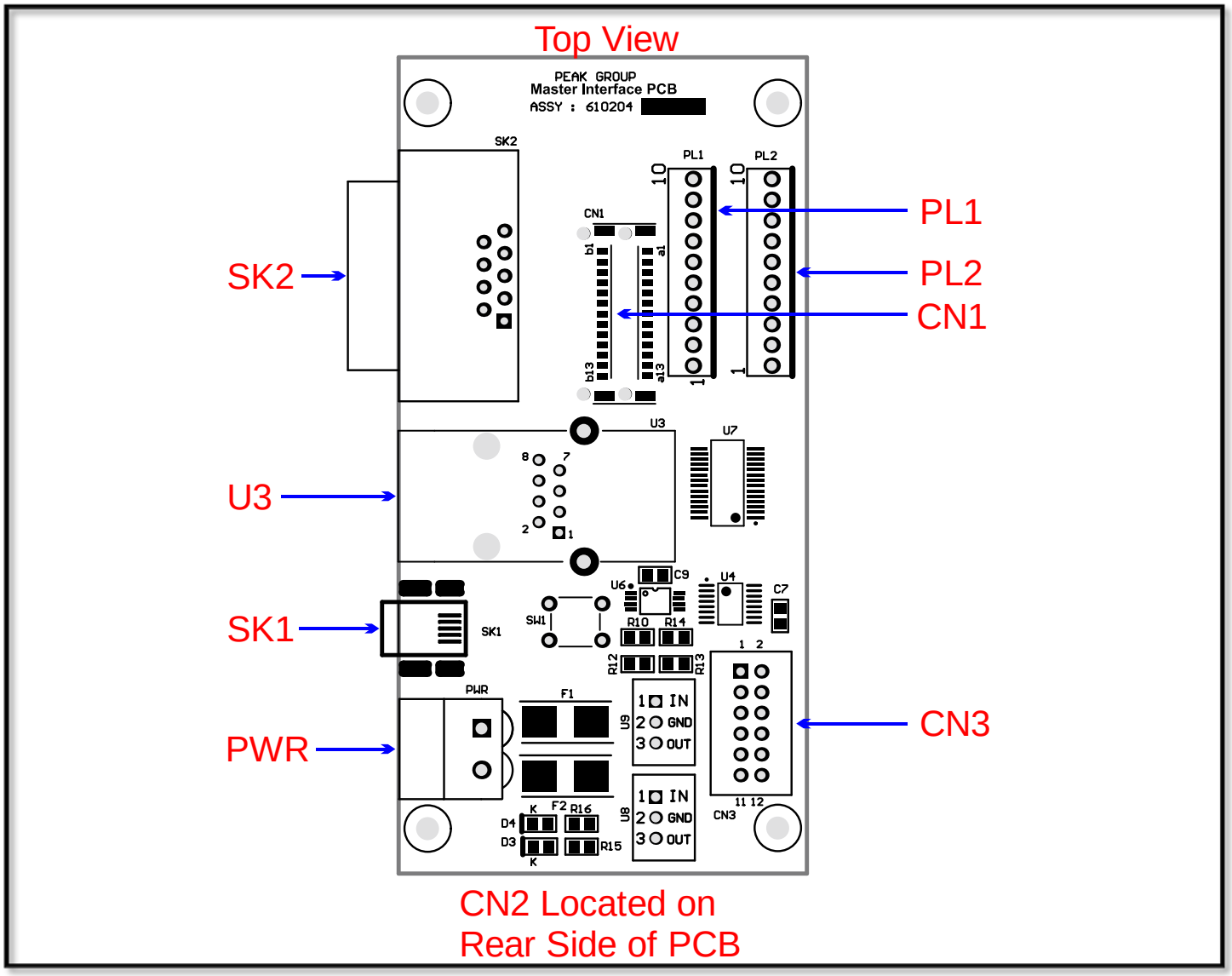
Master Interface - Master Interface Module without Ethernet interface
Master Interface-ETH - Master Interface Module fully populated with all 3 interfaces.

Specifications

Parameter	Min	Typ	Max	Unit
Supply Voltage	7	-	36	VDC
Supply Current (24VDC Supply Voltage)[1]	32	-	60	mA
Operating Tempreature	-30	-	+75	°C
I²C Bus Speed	37	100	400	kbit/s
I²C HIGH-level Input Voltage	0.7	-	5.5	VDC
I²C LOW-level Input Voltage	-0.5	-	0.3	VDC
GPIO 0-2 HIGH-level Output Current	-	-	8	mA
GPIO 3-7 HIGH-level Output Current	-	-	20	mA
GPIO 0-7 LOW-level Output Current	-	-	20	mA
GPIO 0-7 Maximum Total I/O Current	-	-	120	mA
GPIO 0-7 HIGH-level Output Voltage	-	-	3.3	VDC
GPIO 0-7 Negative-going Threshold Voltage[2]	0.22	-	0.4	VDC
GPIO 0-7 Positive-going Threshold Voltage[2]	-	0.6	0.7	VDC
GPIO Port 0 & 1 HIGH-level Output Current (Sourced)	-	-	25	mA
GPIO Port 0 & 1 LOW-level Output Current (Sunk)	-	-	25	mA
GPIO Port 0 & 1 Maximum Total I/O Current	-	-	200	mA
GPIO Port 0 & 1 HIGH-level Output Voltage	-	4.7	5	VDC
GPIO Port 0 & 1 Negative-going Threshold Voltage[2]	0.15	-	0.8	VDC
GPIO Port 0 & 1 Positive-going Threshold Voltage[2]	0.8	-	5	VDC
Network Interface	10	-	100	Base-T
USB Interface	-	2.0	-	USB
EEPROM Memory	-	-	256	Kbytes
EEPROM Data Erase/Write Cycles	-	-	>1	Million
EEPROM Data Retention	-	-	>200	Years

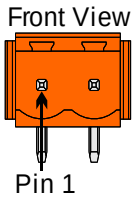
[1] Supply current of Master Interface Module without expansion modules connected.
[2] When GPIO's are configured as inputs.

Connectivity



PWR (POWER)

Pin 1 = +V
Pin 2 = 0V

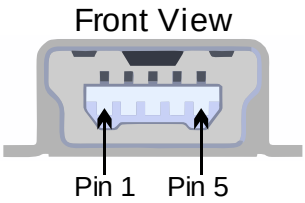


Mating Connector:

Weidmüller – Part No: 1716320000

SK1 (USB)

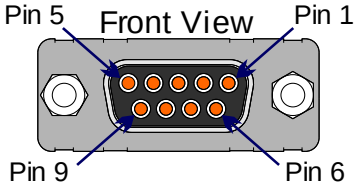
Pin 1 = VBus
Pin 2 = D-
Pin 3 = D+
Pin 4 = N/C
Pin 5 = GND



SK2 (RS232)

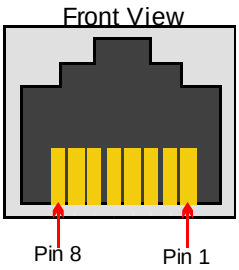
Pin 1 = NC
Pin 2 = Rx
Pin 3 = Tx
Pin 4 = NC
Pin 5 = GND

Pin 6 = NC
Pin 7 = NC
Pin 8 = NC
Pin 9 = NC



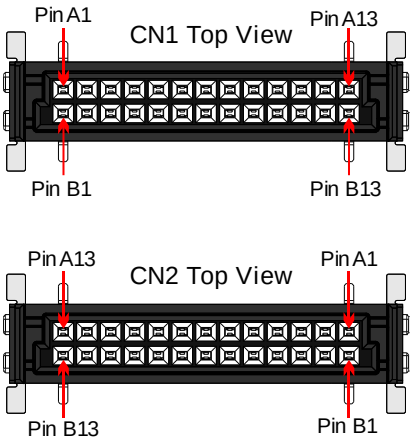
U3 (Ethernet)

Pin 1 = TX+
Pin 2 = TX-
Pin 3 = RX+
Pin 4 = RX-
Pin 5 = NC
Pin 6 = NC
Pin 7 = NC
Pin 8 = NC



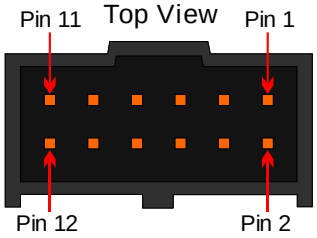
CN1 & CN2 (Slave Link)

Pin a1 = VCC	Pin b1 = VCC
Pin a2 = VCC	Pin b2 = VCC
Pin a3 = VCC	Pin b3 = 0V
Pin a4 = 0V	Pin b4 = 0V
Pin a5 = SDA	Pin b5 = 0V
Pin a6 = SCL	Pin b6 = 0V
Pin a7 = GPIO 1	Pin b7 = GPIO 2
Pin a8 = GPIO 3	Pin b8 = GPIO 4
Pin a9 = GPIO 5	Pin b9 = GPIO 6
Pin a10 = 0V	Pin b10 = 0V
Pin a11 = 0V	Pin b11 = 0V
Pin a12 = VCC	Pin b12 = VCC
Pin a13 = VCC	Pin b13 = VCC



CN3 (Auxiliary)

Pin 1 = VCC	Pin 2 = 3V3
Pin 3 = SDA	Pin 4 = 0V
Pin 5 = SCL	Pin 6 = GPIO 1
Pin 7 = GPIO 2	Pin 8 = GPIO 3
Pin 9 = GPIO 4	Pin 10 = GPIO 5
Pin 11 = GPIO 6	Pin 12 = GPIO 7



Mating Connector:

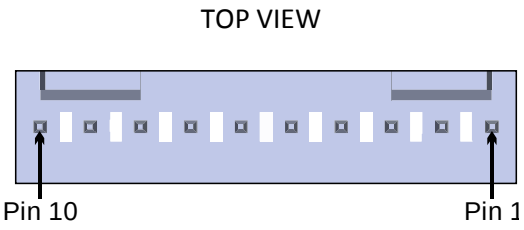
Molex – Part No: 90142-0012

Connector Contacts:

Molex – Part No: 90119-2110

PL1 (GPIO Port 0)

Pin 1 = GPIO 0-0
Pin 2 = GPIO 0-1
Pin 3 = GPIO 0-2
Pin 4 = GPIO 0-3
Pin 5 = GPIO 0-4
Pin 6 = GPIO 0-5
Pin 7 = GPIO 0-6
Pin 8 = GPIO 0-7
Pin 9 = 5V
Pin 10 = 0V



Mating Connector:

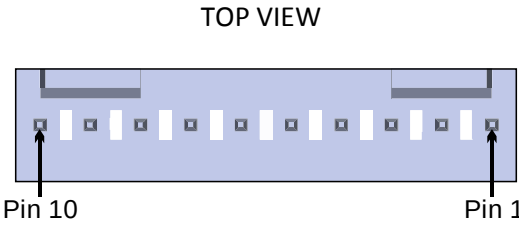
Molex – Part No: 22-01-2105

Connector Contacts:

Molex – Part No: 08-50-0032

PL2 (GPIO Port 1)

Pin 1 = GPIO 1-0
Pin 2 = GPIO 0-1
Pin 3 = GPIO 0-2
Pin 4 = GPIO 0-3
Pin 5 = GPIO 0-4
Pin 6 = GPIO 0-5
Pin 7 = GPIO 0-6
Pin 8 = GPIO 0-7
Pin 9 = 5V
Pin 10 = 0V



Mating Connector:

Molex – Part No: 22-01-2105

Connector Contacts:

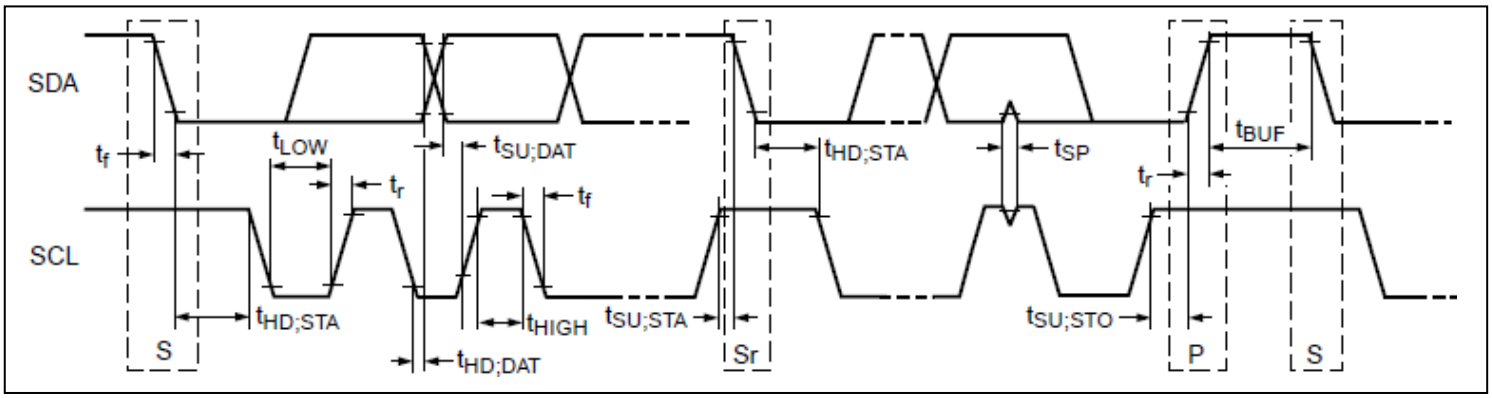
Molex – Part No: 08-50-0032

I²C Bus

The Master Interface Module comes with a master I²C bus for all slave board communications and is also available on CN3 for connection to any other device with a slave I²C interface. As supplied the I²C interface is set in standard mode (100kHz), this can be changed to Fast Mode (400kHz) by sending the commands shown in the communications section.
The Module is fitted with a device that controls all I²C bus specific sequences, protocol, arbitration and timing so control sequences to the Module become very simple.

I²C Bus Timing

			Standard Mode I²C-bus		Fast Mode I²C-bus		
Symbol	**Parameter**	**Conditions**	**Min**	**Max**	**Min**	**Max**	**Unit**
fSCL	SCL clock frequency		0	100	0	400	kHz
tBUF	bus free time between a STOP and START condition		4.7		1.3		µs
tHD;STA	hold time (repeated) START condition		4.0		0.6		µs
tSU;STA	set-up time for a repeated START condition		4.7		0.6		µs
tSU;STO	set-up time for a STOP condition		4.0		0.6		µs
tHD;DAT	data hold time		0		0		µs
tVD;ACK	data valid acknowledge time			0.6		0.6	µs
tVD;DAT	data valid time	LOW-level		0.6		0.6	µs
		HIGH level		0.6		0.6	µs
tSU;DAT	data set-up time		250		100		µs
tLOW	LOW period of the SCL clock		4.7		1.3		µs
tHIGH	HIGH period of the SCL clock		4.0		0.6		µs
tf	fall time of both SDA and SCL signals			0.3		0.3	µs
tr	rise time of both SDA and SCL signals			1		0.3	µs
tSP	pulse width of spikes that must be suppressed by the input filter			50		50	ns



Communications

All communications to the Master Interface Module and Slave modules are made by sending string commands to COM ports or virtual COM ports. **No drivers are required for the RS232 interface,** just simply send and receive data to the COM port the Module is connected to. The USB and Ethernet interface require drivers to function, once the drivers are installed the device will appear as a virtual COM port and data can be sent to and from the Module. It is also possible to send data over Ethernet using TCP. The Ethernet device server IP address can also be changed along with many other settings by installing the Lantronix device installer.

USB Drivers

http://www.ftdichip.com/Drivers/VCP.htm

Ethernet Drivers

https://www.lantronix.com/products/com-port-redirector/

http://www.lantronix.com/products/deviceinstaller/

Commands

1.0.0 – Send data over I²C Bus

To send data to a slave I²C device the following commands need to be sent:

The 1st byte (01010011, 0x53, S)	Start the I2C bus.
The 2nd byte (???????0, ?, ?)	Address the I²C slave device plus 0 (first 7 bits = slave address, last bit (0) = write)
The 3rd byte (????????, ?, ?)	Number of data bytes to send (between 1 and 256 bytes)
Data Bytes	Send data bytes.
Last byte (01010000, 0x50, P)	Stop the I2C bus.

1.1.0 – Read data over I²C Bus

To read data to a slave I²C device the following commands need to be sent:

The 1st byte (01010011, 0x53, S)	Start the I2C bus.
The 2nd byte (???????1, ?, ?)	Address the I²C slave device plus 1 (first 7 bits = slave address, last bit (1) = read)
The 3rd byte (????????, ?, ?)	Number of data bytes to read (between 1 and 256 bytes)
The 4rd byte (01010000, 0x50, P)	Stop the I2C bus.

*After this command you must read the number of data bytes from the seral bus.

2.0.0 - Using Master Module GPIO Ports 0 and 1 as Outputs

First configure the digital pins to be outputs:

Binary = 01010011-01001110-00000010-00000110-00000000-01010000
Hex = 0x53-0x4E-0x02-0x06-0x00-0x50
ASCII = S-N-STX-ACK-NUL-P

The 1st byte (01010011, 0x53, S)	Start the I2C bus.
The 2nd byte (01001110, 0x4E, N)	Address the GPIO function.
The 3rd byte (00000010, 0x02, STX)	We need to send 2 bytes of data. (4th & 5 th byte)
The 4th byte (00000110, 0x06, ACK)	Address Port 0 I/O direction register.
The 5th byte (00000000, 0x00, NUL)	Set all pins to be outputs.
The 6th byte (01010000, 0x50, P)	Stop the I2C bus.

This has set port 0 to be outputs. The same command is required to set port 1 to outputs but the 4th byte will need to be 00000111 (0x07(BEL)) to address Port 1 IO direction register.

We can now write to the GPIO 0 & GPIO 1 registers to change the state of the IO pins.

Binary = 01010011-01001110-00000010-00000001-????????-01010000
Hex = 0x53-0x4E-0x02-0x01-0x??-0x50
ASCII = S-N-STX-SOH-?-P

The 1st byte (01010011, 0x53, S)	Start the I2C bus.
The 2nd byte (01001110, 0x4E, N)	Address the GPIO function.
The 3rd byte (00000010, 0x02, STX)	We need to send 2 bytes of data. (4th & 5 th byte)
The 4th byte (00000000, 0x00, NUL)	Address Port 0 register.
The 5th byte (?)	Set the state of Port 0, LSB=P0.0, MSB=P0.7, 0=LOW, 1=HIGH
The 6th byte (01010000, 0x50, P)	Stop the I2C bus.

The same command is required to change the state of port 1 but the 4th byte will need to be 00000001 (0x01(SOH)) to address the GPIO 1 register.

2.1.0 - Reading the Master Module GPIO Ports 0 and 1 State

Binary = 01010011-01001110-00000001-00000000-01010000-01010011-01001111-00000001-01010000
Hex = 0x53-0x4E-0x01-0x00-0x50-0x53-0x4F-0x01-0x50
ASCII = S-N-SOH-NUL-P-S-O-SOH-P

The 1st byte (01010011, 0x53, S)	Start the I2C bus.
The 2nd byte (01001110, 0x4E, N)	Address the GPIO function.
The 3rd byte (00000001, 0x01, SOH)	We need to send 1 byte of data. (4th byte)
The 4th byte (00000010, 0x00, NUL)	Address Port 0 register.
The 5th byte (01010000, 0x50, P)	Stop the I2C bus.
The 6th byte (01010011, 0x53, S)	Start the I2C bus.
The 7th byte (01001111, 0x4F, O)	Address the GPIO function asking to send data.
The 8th byte (00000001, 0x01, SOH)	Send 1 byte of data.
The 9th byte (01010000, 0x50, P)	Stop the I2C bus.

*After this command you must read 1 byte of data from the seral bus. This byte of data will show the state of the port, LSB=P0.0, MSB=P0.7, 0=LOW, 1=HIGH.

The same command is required to read the state of port 1 but the 4th byte will need to be 00000001 (0x01(SOH)) to address the GPIO 1 register.

2.2.0 - Using Master Module GPIO Ports 0 and 1 as Inputs

First configure the digital pins to be inputs:

Binary = 01010011-01001110-00000010-00000110-11111111-01010000
Hex = 0x53-0x4E-0x02-0x06-0xFF-0x50
ASCII = S-N-STX-ACK-ÿ-P

The 1st byte (01010011, 0x53, S)	Start the I2C bus.
The 2nd byte (01001110, 0x4E, N)	Address the GPIO function.
The 3rd byte (00000010, 0x02, STX)	We need to send 2 bytes of data. (4th & 5th byte)
The 4th byte (00000110, 0x06, ACK)	Address Port 0 I/O direction register.
The 5th byte (11111111, 0xFF, ÿ)	Set all pins to be inputs.
The 6th byte (01010000, 0x50, P)	Stop the I2C bus.

This has set port 0 to be inputs. The same command is required to set port 1 to inputs but the 4th byte will need to be 00000111 (0x07(BEL)) to address the Port 1 IO direction register.

We can now read the state of the input ports as shown in section 1.1.0.

3.0.0 - Using Master Module GPIO 0 to 7 as Outputs (on Auxiliary Connector)

First set the GPIO 0 to 7 as outputs.

Binary = 01010111-00000010-10101010-01010000-01010111-00000011-10101010-01010000
Hex = 0x57-0x02-0xAA-0x50-0x57-0x03-0xAA-0x50
ASCII = W-STX-a-P-W-ETX-a-P

The 1st byte (01010111, 0x57, W)	Address the GPIO 0-7 function.
The 2nd byte (00000010, 0x02. STX)	Select GPIO 0 to 3.
The 3rd byte (10101010, 0xAA, a)	Set GPIO0-3 as push-pull outputs.
The 4th byte (01010000, 0x50, P)	Terminate this frame.
The 5th byte (01010111, 0x57, W)	Address the GPIO 0-7 function.
The 6th byte (00000011, 0x03, ETX)	Select GPIO 4 to 7.
The 7th byte (10101010, 0xAA, a)	Set GPIO4-7 as push-pull outputs.
The 8th byte (01010000, 0x50, P)	Terminate this frame.

We can now write to the GPIO 0 to7 register to change the state of the IO pins.

Binary = 01001111-????????-01010000
Hex = 0x4F-0x??-0x50
ASCII = O-?-P

The 1st byte (01001111, 0x4F, O)	Address the GPIO 0-7 register.
The 2nd byte (?)	Set the state of GPIO 0-7, LSB=GPIO 0, MSB=GPIO 7, 0=LOW, 1=HIGH
The 3rd byte (01010000, 0x50, P)	Terminate this frame.

3.1.0 - Reading the Master Module GPIO Port 2 State

Binary = 01001001-01010000
Hex = 0x49-0x50
ASCII = I-P

The 1st byte (0101001, 0x49, I)	Ask for status of GPIO 0-7.
The 2nd byte (01010000, 0x50, P)	Terminate this frame.

*After this command you must read 1 byte of data from the seral bus. This byte of data will show the state of the port, LSB=P0.0, MSB=P0.7, 0=LOW, 1=HIGH.

3.2.0 - Using Master Module GPIO Port 2 as Inputs

First set the GPIO 0 to 7 as inputs.

Binary = 01010111-00000010-01010101-01010000-01010111-00000011-01010101-01010000
Hex = 0x57-0x02-0x55-0x50-0x57-0x03-0x55-0x50
ASCII = W-STX-U-P-W-ETX-U-P

The 1st byte (01010111, 0x57, W)	Address the GPIO 0-7 function.
The 2nd byte (00000010, 0x02, STX)	Select GPIO 0 to 3.
The 3rd byte (01010101, 0x55, U)	Set GPIO0-3 as inputs.
The 4th byte (01010000, 0x50, P)	Terminate this frame.
The 5th byte (01010111, 0x57, W)	Address the GPIO 0-7 function.
The 6th byte (00000011, 0x03, ETX)	Select GPIO 4 to 7.
The 7th byte (01010101, 0x55, U)	Set GPIO4-7 as inputs.
The 8th byte (01010000, 0x50, P)	Terminate this frame.

We can now read the state of the input ports as shown in section 2.1.0.

4.0.0 – Writing Data to EEPROM

The EEPROM memory has 500 pages with 64 bytes of data on each, as shown below.

Memory Map

Page \ Byte	1	2	3	4	5	6	-	64
0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
1	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
2	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
3	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
4	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
5	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
6	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
7	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
8	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
9	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
10	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
11	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
12	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
-	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
500	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

4.0.1 – Byte Write

To write bytes of data to the EEPROM the following commands need to be sent:

The 1st byte (10101110, 0xAE)	Address the EEPROM plus 0 (first 7 bits = slave address, last bit (0) = write)
The 2nd byte (????????, ?)	Memory address high byte
The 3rd byte (????????, ?)	Memory address low byte
Data Bytes	Data
Stop	

Example:

0xA0, 0x00, 0x00, Data = Addressing byte 1 of page 1

If more than 64 bytes of data are sent the device will automatically roll over to the next page e.g. If 100 bytes of data are sent to page 1, page 1 will contain the first 64 bytes and page 2 will contain the last 36 bytes.

4.0.2 – Page Write

To write a complete page of data to the EEPROM the following commands need to be sent:
Note: A complete page consists of 64 bytes, if less than 64 bytes are sent, the remaining bytes will be refreshed.

The 1st byte (10101110, 0xAE)	Address the EEPROM plus 0 (first 7 bits = slave address, last bit (0) = write)
The 2nd byte (????????, ?)	Memory address high byte
The 3rd byte (????????, ?)	Memory address low byte
Data Bytes (64 maximum)	Data
Stop	

Example:

0xA0, 0x00, 0x00, Data = Addressing Page 0
0xA0, 0x00, 0x01, Data = Addressing Page 1

0xA0, 0x01, 0xF4, Data = Addressing Page 500

4.1.0 – Reading Data from EEPROM

4.1.1 – Current Address Read

This function can be used to read the next byte from a previous read. E.g. if page 1 byte 3 was the last byte read from the device the following command will read page 1 byte 4. After this the address is internally incremented by 1 so if the command is sent again it will read from the next byte in sequence.

The 1st byte (10101110, 0xAE)	Address the EEPROM plus 0 (first 7 bits = slave address, last bit (1) = read)
Read Data Byte	Read the data back
Stop	

4.1.2 – Random Read

If you need to read data from a random location the following commands must be sent.

First the location address will need to be set:

The 1st byte (10101110, 0xAE)	Address the EEPROM plus 0 (first 7 bits = slave address, last bit (0) = write)
The 2nd byte (????????, ?)	Memory address high byte
The 3rd byte (????????, ?)	Memory address low byte
Stop	

Then the data can be read by the following command:

The 4th byte (10101111, 0xAF)	Address the EEPROM plus 0 (first 7 bits = slave address, last bit (1) = read)
Read Data Byte	Read the data back
Stop	

After a random read command, the internal address counter will point to the address location following the one that was just read.

4.1.3 – Sequential Read

A sequential read allows users to define a start address as a random read does but allows multiple bytes to be read back with a single command by sending acknowledgements between the bytes received.

First the location address will need to be set:

The 1st byte (10101110, 0xAE)	Address the EEPROM plus 0 (first 7 bits = slave address, last bit (0) = write)
The 2nd byte (????????, ?)	Memory address high byte
The 3rd byte (????????, ?)	Memory address low byte
Stop	

Then the data can be read by the following command:

The 4th byte (10101111, 0xAF)	Address the EEPROM plus 0 (first 7 bits = slave address, last bit (1) = read)
Read Data Byte	Read the data back
Acknowledge	
Read Data Byte	Read the data back
Etc.	
Stop	

After each acknowledge, the internal address counter will point to the address location following the one that was just read.

5.0.0 – Connection to Slave Boards

TBC