REV	NOTES	DATE
0	Preliminary Manual	
1	Initial Release: Added PID loop control and the corresponding I2C commands to original PBoard (see-1 manual)	12/04/2014

PID PBoard
Operators Manual

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

Sigenics Inc.
University Technology Park at IIT
3440 S. Dearborn St. – Suite 105N
Chicago, IL 60616

Author:	Date:	Reviewed by:	Date:	Approved by:	Date:
P Chen	11/16/2014	G DeMichele	11/16/2014	S David	11/16/2014

1. Introduction

This manual describes the operation and general specifications for a revision of the UCD Motor Control System. The interface definitions of the modules comprising the system are presented, as well as general specifications for the system modules.

The original PBoard is described in MAN800119-1 UCD Motor Controller Operators Manual. The PID PBoard incorporates the following changes:

1). New PCD for the PID PBoard

The dimensions of the original PBoard were 0.675" x 0.475" and had vias to connect to the motor by flywires and a connector to connect to the power/I2C bus.

The PID PBoard has dimensions of 0.650" x 0.425", but the corners may be ground off to make the board slightly smaller. The bus connector has been replaced by vias for flywires. PIC Programming vias(GND, +5V, VPP, ICPSDAT, ICPSCLK) have been added to the board to allow the PID PennyBoard to be easily reprogrammed after assembly(Figure 3).

2). New firmware for the PID PBoard

The functionality of the original PBoard has been preserved, but new commands have been added to support the added PID control loop functionality of the PID PBoard. The command structure and syntax are identical to that which will be used by the Motor Controller ASIC currently in development, so that any software developed by UCD to control the PID PBoards can be used with little or no modification on the Motor Control ASICs.

2. Definitions

2.1. ***Motor Controller System*** The described system, which is capable of controlling multiple motors. This system comprises several ***PBoards***, a ***Controller Board***, and a PC running ***PC Demo Software***.

2.2. ***PBoard*** A small satellite board which handles the driving and position sensing of one motor. All PBoards in a system are connected across a 4-wire bus. This bus consists of Vbatt, Groud, and a 2-wire serial I2C interface. The PBoard communicates as a slave with the master Controller Board via this I2C interface. The PBoard also contains custom ***PBoard Firmware*** which determines the PBoard behavior. The term "PBoard" grew from the informal name "Penny Board" which was coined because the PBoard is approximately the size of a penny.

2.3. ***Controller Board*** In this system, the Controller Board consists of an off-the-shelf Arduino microcontroller system. The Controller Board talks as a master to all PBoards via the 2-wire I2C interface, and to the PC via a USB port. In this system, the ***Controller Board Firmware*** only translates PC USB commands to I2C. In a future prosthetic system however, a Controller Board could also contain algorithms which control the PBoards based on neural or other sensor input to implement a self-contained prosthesis controller.

2.4. ***PC Demo Software*** Software running on an external PC used to demonstrate the functionality of the Motor control System.

2.5. ***COTS*** A commercial-off-the-shelf item, available for purchase from an existing vendor.

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

3. Motor Controller Hardware Architecture

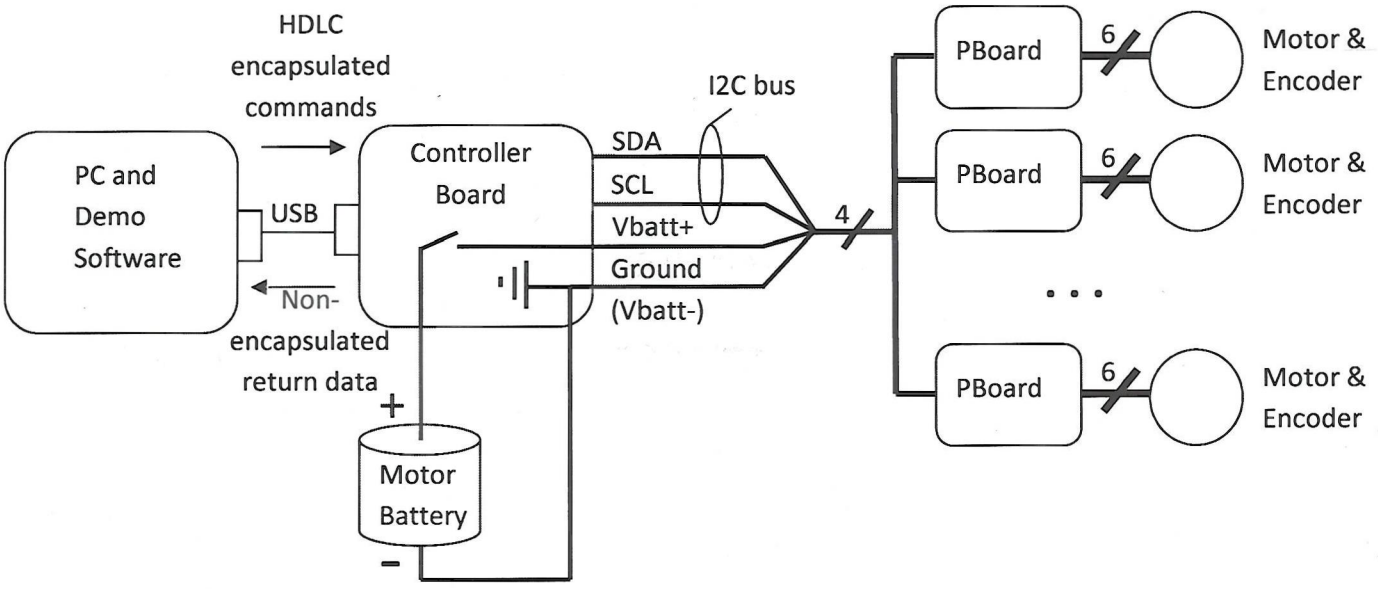

Figure 1). Motor Controller System Block Diagram

The Motor Controller System block diagram is shown above in Figure 1). The processor on the Controller board gets its power and commands from the PC via the USB bus. The motor battery is connected to terminals on the controller board, and this power is applied to only the Vbatt+ and Ground wires in the 4-wire PBoard bus. The Vbatt- terminal is connected to system ground on the Controller Board and PBoards, and is also the ground reference for the I2C bus.

The architecture of the system will support up to 127 motors, however the Vbatt+ and Ground wires in the PBoard bus must be heavy enough to carry the motor currents. In addition, since motor control is done over a serial data bus, system response time must be considered when driving a large array of motors.

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

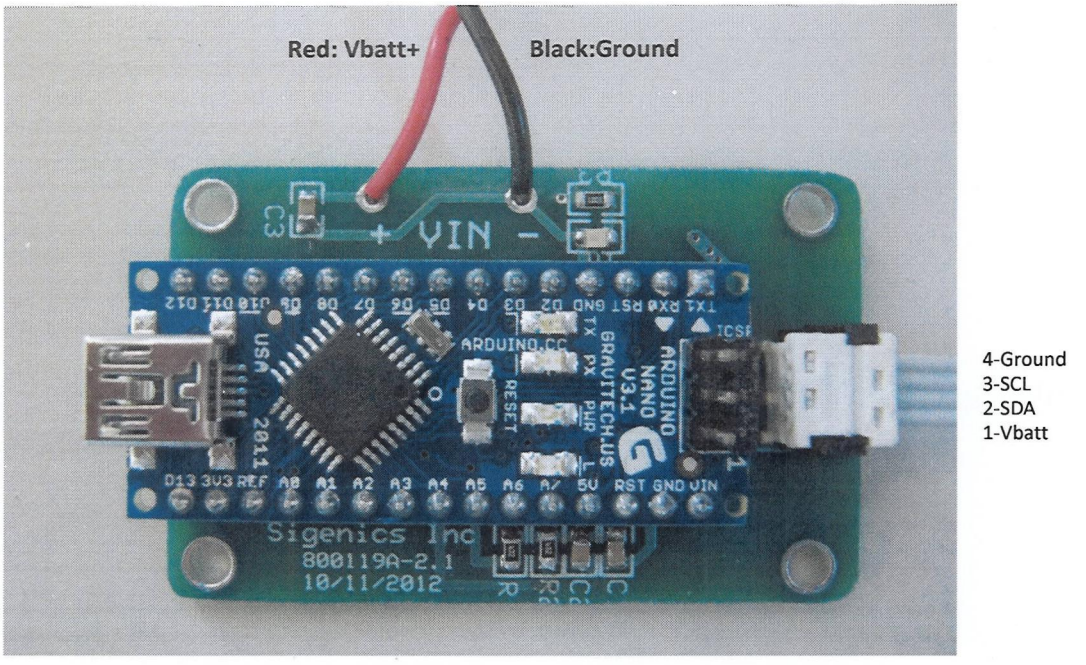

Figure 2). Controller Board

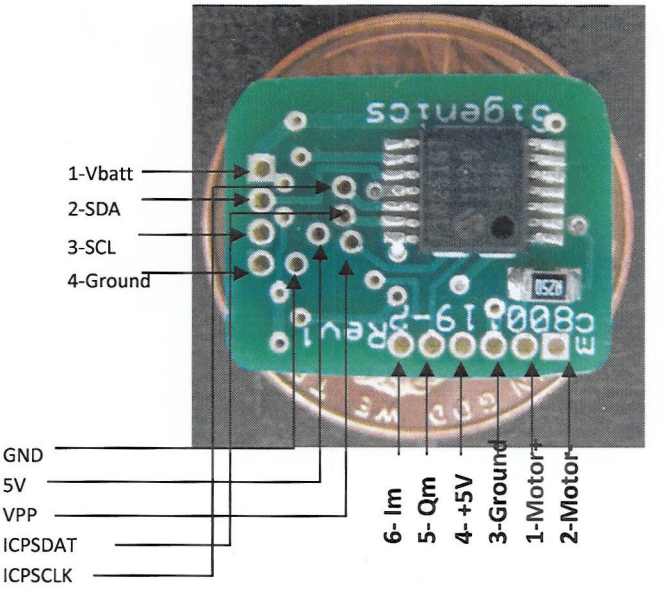

Figure 3). PID PBoard

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

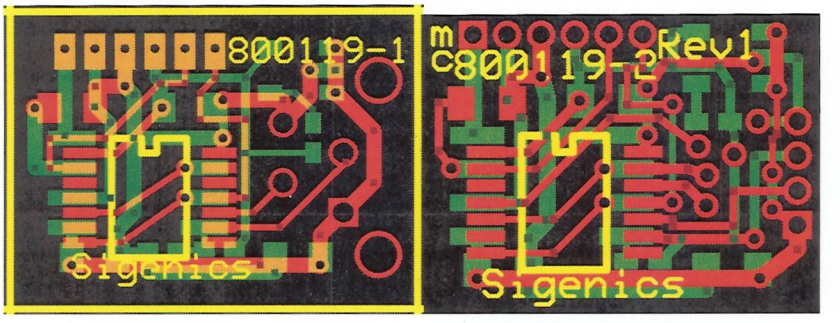

Figure 4). Original (left) and New PID PBoard (right)

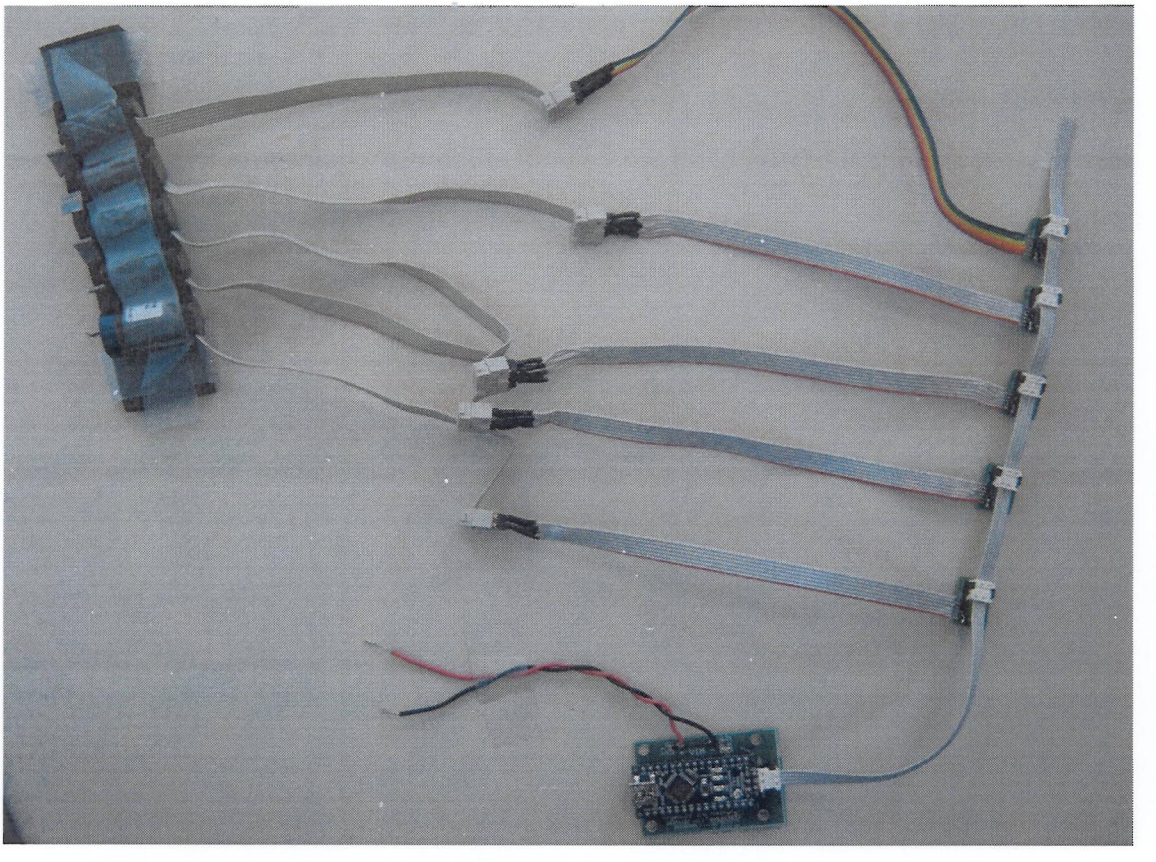

Figure 5). Controller Board, PBoards and motors wired as a system

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

PC and Demo Software
The PC and Demo Software present a control panel to the user to allow the demonstration of motor functionality. The PID control constants may be changed. The PID loop may be disabled by the user, so a forced PWM value can be applied directly to the motor. Braking, or coasting can also be used with the PID loop disabled. These features correspond to the operation of the original PBoard.

Controller Board and Controller Board Firmware
The Controller Board is a board containing an Arduino Nano, which is powered by the PC USB port. This board will run simple Controller Board Firmware which implements a one-for-one bidirectional translation of the PBoard commands and responses between the I2C and USB data buses.

In addition to the USB-to-I2C command translation, an I2C address scan utility is also programmed into the controller board. This allows the user to quickly ascertain the addresses of all PBoards connected to the controller board. This function could also be carried out by code in the PC if desired.

Since the I2C interface to the PBoards is well-defined and documented, the client can later substitute a COTS DSP demo board for the DSP of choice to prove out more sophisticated control code. This flexibility allows for an efficient development path using proven hardware to ultimately design a compact Controller Board using the DSP of choice.

PBoard and PBoard Firmware
The PBoard contains a PIC microprocessor and single motor drive chip. Each PBoard in a system should have a unique 7-bit I2C bus address, 1 to 127, with address 0x00 reserved as a "broadcast" address. Controlled by I2C commands, the PIC applies PWM and direction to the motor driver to drive the motor in either direction at a controlled torque. The optical encoder mounted on each motor present I and Q pulse signals for PIC processing, and the PIC maintains an up/down count representing the motor position. This motor position count can be reset or read by the Controller Board using I2C commands. Full PID loop control also has been added to the PIC microprocessor. The details of the PID mode are described in the following section.
The Motor Battery directly feeds the motor driver chip on the PBoard to drive the motors. This battery power is locally regulated on each PBoard to provide a stable low-voltage supply for the PIC microprocessor on the PBoard.

4. Controller Board to PBoard I2C Protocol

The Controller Board communicates with the PBoard using a 2 wire I2C interface. The PBoards are the slaves, and the Controller Board is the master. The I2C frame structure is shown in Figure 4.

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

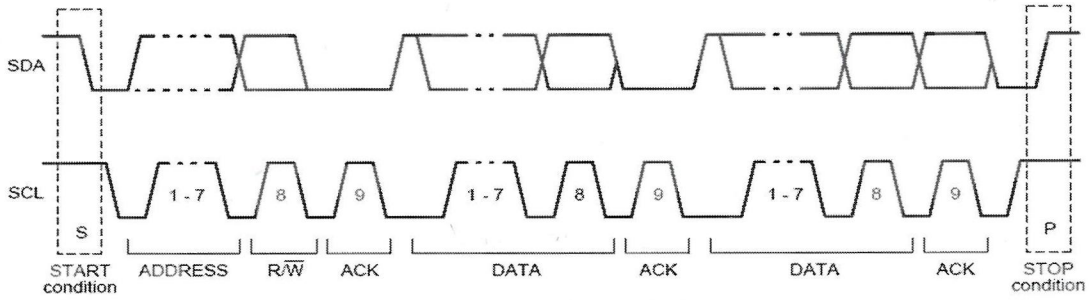

Figure 6). Frame structure of I2C protocol

The master first asserts the START condition, then sends 7 bits of slave device address. The master then asserts R/W_ to indicate whether the command is a Read (1) or Write (0). The master then releases the bus during the ACK bit, where the slave pulls ACK low to indicate proper receipt of the byte. For a write command, the master continues to assert data bits 1-8, and listen for each ACK bit from the slave. For a read command, the slave asserts data bits 1-8, and the master must ACK low during the ACK bit. Data is sent MSB first.

5. PBoard I2C Command Set

All PBoards have a 7-bit I2C address, 1-127. Address 0x00 is reserved as the broadcast address, which allows one write command to be received by multiple PBoards, for example all motors in a system can be turned off simultaneously by sending a broadcast PWM=0 command.
The I2C protocol however does not allow the broadcast address to be used for read commands. Read commands with an address of 0x00 will be ignored.

Note that all write commands start with an {Address} followed by a {CommandCode}. If the {CommandCode} does not match one of the codes defined in the table below, no command will be executed.

Read commands only address a PBoard, and I2C does not allow a command code to be sent with a Read command.

In the original PBoard, the Read command returned only the value of the Position Counter. In the PID PBoard, eight different variables may be read using the Read command. In order to do this, the ***SetReadTarget*** write command tells the PID PBoard what parameter to return when a Read command is issued.

Write Commands
{START Condition} {Address} {R/W_=0} {ACK} {CommandCode[7:0]} {ACK}{payload} {STOP Condition}

Note: Command names preceded by "" are commands added to support PIC control in the PID PBoard.*

Command Name	CommandCode	Payload
SetPositionCount	0000 1000	{MSbyte}{LSbyte}

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

A 16-bit counter in the PBoard counts the pulses generated by the optical encoder on the motor. This command sets this position counter to a known value, and is used to set a zero reference position for the motor. The counter is set to the value {MSbyte,LSByte}

SetPWM 0000 1100 {MSbyte}{LSbyte}
This command implements control of motor drive as follows:

Coast and Brake:
MSByte= 0000 0000 LSByte= XXXX XXXX => Bridge off (Z,Z Coast)
MSByte= 0000 0011 LSByte= XXXX XXXX => Bridge on (L,L Brake)
Forward(R=0) or Reverse (R=1) with PWM
MSByte= 1RXX XXXX LSByte= {8 bit PWM value}

A PWM value of 0 will result in no motor drive (Bridge=L,L, Duty Cycle=0%), and a PWM value of 100 or greater will result in Bridge=H,L (R=0) or Bridge=L,H (R=1) for a 100% duty cycle

****SetEnablePID*** 1000 0000 {Byte}
This command implements control of PID mode as follows:
Byte = 0XXX XXXX Disable the PID function
Byte = 1XXX XXXX Enable the PID function

****SetKpParameter*** 1000 0001 {MSbyte}{byte}{byte}{LSbyte}
This command will set the Kp parameter in the PID formula. The Kp register is set to the value {MSbyte,LSbyte}.

****SetKiParameter*** 1000 0010 {MSbyte}{byte}{byte}{LSbyte}
This command will set the Ki parameter in the PID formula. The Ki register is set to the value {MSbyte,LSbyte}.

****SetKdParameter*** 1000 0011 {MSbyte}{byte}{byte}{LSbyte}
This command will set the Kd parameter in the PID formula. The Kd register is set to the value {MSbyte,LSbyte}.

****SetTargetPosition*** 1000 0100 {MSbyte}{LSbyte}
This command will set the target position. The target position register is set to the value {MSbyte,LSbyte}, and when active, the PID loop strives to make PositionCounter=TargetPosition

****SetReadTarget*** 0000 0001 {Byte}
This command determines which parameter is returned when a Read command is issued to the PBoard.
Byte = 0000 0000 Read position counter.
Byte = 0000 0001 Read Target Position
Byte = 0000 0010 Read Kp
Byte = 0000 0011 Read Ki
Byte = 0000 0100 Read Kd
Byte = 0000 0101 Read 32-bit u(t)

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

Byte = 0000 0110	Read PIDEn
Byte = 0000 0111	Read 32-bit E(t)
Byte = XXXX 1000	Reserved
Byte = XXXX 1001	Reserved
Byte = XXXX 1010	Reserved
Byte = XXXX 1011	Reserved
Byte = XXXX 1100	Reserved
Byte = XXXX 1101	Reserved
Byte = XXXX 1110	Reserved
Byte = XXXX 1111	Reserved

SetAddress 1010 1101 {current address}{new address}

This command assigns a new I2C address to the PBoard. The I2C {Address} could be either the PBoard address or the broadcast address of 0x00, but this command is executed only if the current address of the PBoard is equal to {current address} and the {new address}<>0. In the event of a command programming error, this prevents all PBoard I2C addresses from being inadvertently set to the same {new address}.

Read Command (note:Broadcast Read operations are not allowed)

{START Condition} {Address} {R/W_=1} {ACK} {PayloadMSByte} {ACK} ... {PayloadByte} {ACK} ... {PayloadLSByte} {ACK} {PBoard Address} {ACK} {STOP Condition}

The Read command returns bytes of payload data (MSByte, ... , LSbyte) followed by the PBoard address. The number of bytes returned depends on which parameter is selected using the ***SetReadTarget*** write command.

The number of bytes is shown below:

Read Parameter	**Number of Payload bytes**
Read position counter.	2
Read Target Position	2
Read Kp	4
Read Ki	4
Read Kd	4
Read 32-bit u(t)	4
Read PIDEn	1 (MSB bit =1 if PID is enabled)
Read 32-bit E(t)	4

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

6. PID Control Algorithm

The PID computation used in the microprocessor is known as the "velocity form", and yields the following time domain expression:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$$

e(t) is the error term, it comes from

e = Target_Position – Current_Position

Both *Target_Position* and *Current_Position* are 16-bit unsigned integers from the user point of view (i.e. the user sets only 16-bit registers). 32 bit floating point math is used in the microprocessor, resulting in a 32 bit value for u(t).

6.1. Representation of Kp, Ki and Kd

Kp, Ki and Kd are represented as single-precision floating point numbers represented in 4 bytes. This standard format is described in this link:
http://en.wikipedia.org/wiki/Single-precision_floating-point_format

6.2. Demo GUI description

The demo GUI was written in LabView (code available upon request).

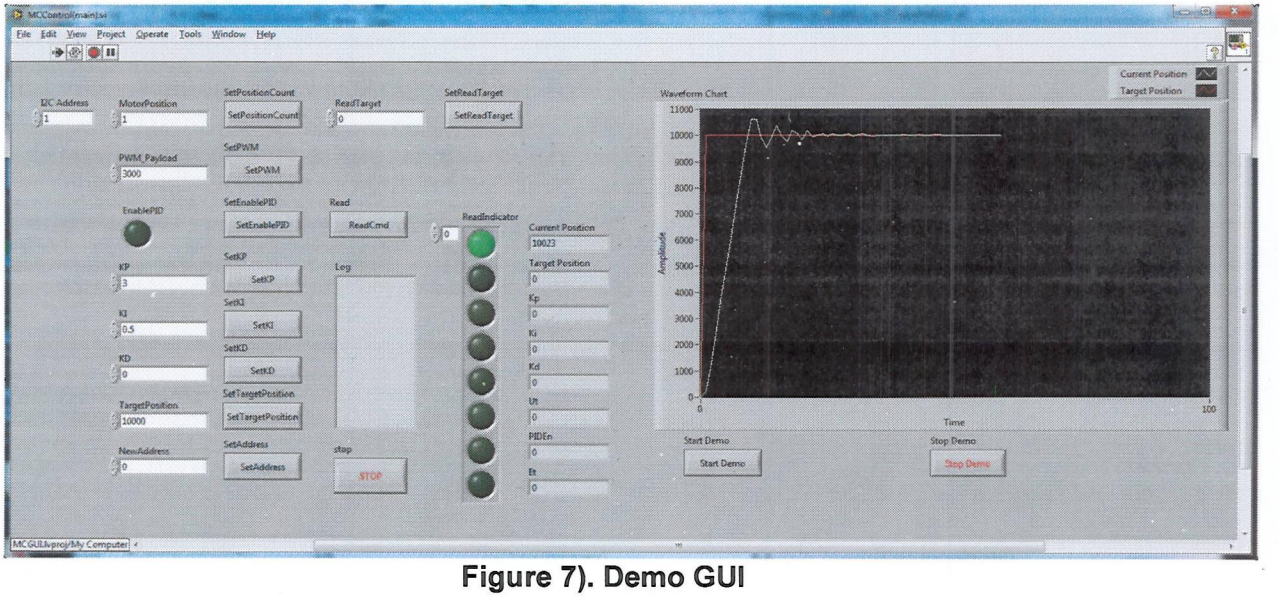

Figure 7). Demo GUI

Instructions:
1). Connect motor to PBoard as shown in Figure 3.
2). Connect the PBoard to the I2C bus (this includes the Battery connections).
3). Connect the I2C bus to the Arduino Control Board.

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

4). Connect the Battery wires in the I2C bus to a 9V-12V DC power supply. OBSERVE CORRECT POLARITY.
5). Connect USB between Arduino Control Board and PC.
6). Launch the LabView demo control panel.
7). Input the I2C address and all other desired parameters into the white text boxes. Do not click the "Set" buttons, as this is done automatically when you start the demo.
Note: Initially use 0 for PWM, since large PWM's may cause safety current limit trip out on PBoard. To reset this trip out, set PWM to 0 and click the SetPWM button.
8). When StartDemo is clicked, the demo software will automatically load the contents of all of the white boxes to the PBoard with the exception of NewAddress and PWM_Payload. It will also SetEnablePID to turn on the control loop. On the plot, the target position is shown in red, and the current position is shown in white.
10). The loop will continue to run indefinitely until the Stop Demo button is pushed. This allows the user to let the position reach the target, then perturb the motor to see how the loop responds to the perturbations.

6.3. Parameter values used for the "Eraser load" demonstration

Hardware:
1. Sigenics USBtoI2C V1.0 20120904 (Controller board)
2. Sigenics PennyBoard Tester 20120904 (Penny board)
3. Motor (Model: FAULHABER 1724T4.5 SR IE2-16 105)
4. Motor Load: PaperMate pink pearleraser with motor shaft at eraser's center of mass.
5. 12V power supply and PC with MotorController GUI

PID PBoard Parameter Settings for demo:

I2C Addr:	1
MotorPosition:	1
PWM_Payload:	0
KP:	3
KI:	0.5
KD:	0
TargetPosition:	10000

7. PC to Controller Board HDLC protocol

The Controller Board is a simple USB to I2C converter. Because USB packets are sent and received as a bit stream (packetizing of USB data happens at a lower level), there needs to be a way to delimit the PBoard I2C commands in the USB stream. The HDLC protocol is used for PC to Controller Board "Forward" communications to delimit these I2C commands. Since the Controller Board never sends data to the PC without first receiving a command, no method is necessary to delimit "Reverse" data from the Controller Board to the PC.

In HDLC, there are two special characters defined, a "FLAG" and an "ESC" character. A simple "ByteStuffing" algorithm ensuresthat FLAG or ESC will never occur in the transmitted data stream, so FLAG and ESC are sufficient to divide the data stream into logical packets (the PBoard I2C commands).

In the USB data stream, the first HDLC FLAG indicates that a I2C start condition is to be asserted by the Controller Board. USB bytes that follow are sent directly to the I2C after

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

HDLC byte stuffing and unstuffing. The second HDLC FLAG indicates that an I2C STOP condition is to be asserted by the Controller Board.

7.1. Simplified HDLC frame structure

The forward commands are send to the Controller Board through a USB interface. The commands are encapsulated into HDLC frames by simplified asynchronous HDLC protocol. It uses "byte stuffing" or "octet stuffing". The frame boundary octet, also called "FLAG", is 01111110 (7E in hexadecimal notation). A "control escape octet", also called "ESC", has the bit sequence 01111101 (7D hexadecimal). If either of these two octets appears in the transmitted data, an escape octet is sent, followed by the original data octet with bit 5 inverted (where bit 0 is the LSB).

For example, the data sequence "01111110" (7E hex) would be transmitted as "01111101 01011110" ("7D 5E" hex). The data sequence "01111101" (7D hex) would be transmitted as "01111101 01011101" ("7D 5D" hex). The simplified HDLC frame structure is shown below in Figure 1. Note that the Forward Commands shown would be byte stuffed to prevent the FLAG from appearing inside the Forward Command stream of data bytes.

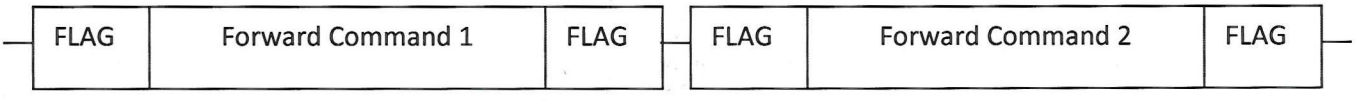

Figure 8). Simplified HDLC protocol

This protocol is called "Simplified HDLC" because the full HDLC specification also describes error correction/detection methods. In this system however, error detection/correction is not used. *This HDLC transport mechanism is used only to send data from the PC to the Controller Board and not used when data is sent from the Controller Board to the PC.*

Note that the HDLC protocol is not used on the I2C bus, but only used in the PC to USB data path.

8. Controller Board Command Set

In the ultimate application of the PBoards in a prosthesis control system, the user is expected to replace the controller board with a DSP processor and code which controls the PBoards using I2C commands as previously described.

The controller board supplied with this system directly translates PC-originated HDLC USB commands into I2C commands. In the "reverse" direction, I2C payloads are passed directly to the USB without HDLC encoding for interpretation by the PC software.

There is only one utility command present in the Controller Board, an I2C address scan utility. This allows the user to ascertain the addresses of all PBoards connected to the

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

controller board. This function could also be carried out by the issuance of multiple I2C commands by the PC if desired.

The I2C address scan utility is accessed by sending data that is not encapsulated by HDLC start flags. Specifically, sending a 0x2C byte which is outside of any HDLC START/STOP Flag boundaries will cause the controller to scan the bus and issue the list of present devices. Currently only device addresses up to 0x47 (decimal 71) are supported by this utility.

The following table shows the format of the list which is reported back to the PC via USB as a result of this I2C address scan function:

Byte 0 (MSB)	Byte 1	Byte 2	Byte 3	...	Byte 71 (Max Address)
# of Devices Detected	Device Address	Device Address	Device Address	...	Device Address

Only detected addresses are included in the report. Therefore if 3 devices are detected, 4 bytes are reported to the PC Host: The first byte contains the number 3, which is the number of devices detected. This alerts Host software of the number of bytes to expect in the rest of the report. Next, the Arduino will report the addresses of the 3 devices detected on the bus, in the order they were detected (lowest address first).

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.

9. System Specifications

9.1. Controller Board

Parameter	Min	Max	Units	Comment
Vdd	4	5.5	Volts	Supplied via USB connector

9.2. PBoard

Parameter	Min	Max	Units	Comment
Vbatt	8	16	Volts	PBoard operating voltage supplied by battery. Motor driver chip has an undervoltage lock-out at 8V.
Idrive	--	2	Amps	Motor overcurrent will result in trip-out of driver. This condition may occur when rapidly changing motor direction. Resetting PWM to 0% will reset the trip-out.
Thermal Shutdown		160	C	Trip point with increasing temperature
Thermal Hysteresis		20	C	
Iqq		TBD	uA	Quiescient current (no motor current)

This document contains information that is the property of Sigenics Inc. This document may not, in whole or in part, be duplicated, disclosed, or used for design or manufacturing purposes without the prior written permission of Sigenics Inc.