USB IO HID Datasheet

USB HID Low Speed Peripheral Controllers

902270 – USB HID Chip 10 I/O SOIC18

902370 – USB HID Chip 10 I/O DIP18

902670 – USB HID Chip 16 I/O SOIC24

902770 – USB HID Chip 16 I/O DIP24

Delcom Products Inc. 200 William St STE302 – Port Chester NY 10573 USA (914)934-5170 www.delcomproducts.com

Table of Contents

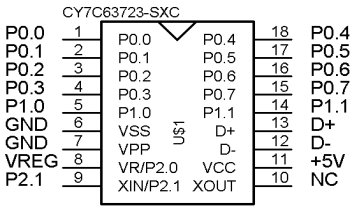
1 Functional Overview 3
2 Pin Definitions 4
3 Pin Descriptions 4
4 Circuit Layout 5
4.1 Typical Schematic 5
4.2 Backward Compatibility 5
5 Programmed Features 6
5.1 Basic I/O 6
5.2 Write data with Strobe 6
5.3 Read Data with Strobe 6
5.4 Clock Generator 6
5.5 PWM 7
5.6 Port Setup 7
5.7 Read Buffer 7
5.8 Scratch Pad 8
5.9 Event Counter 8
5.10 RS232 Serial Port 8
5.11 I^2C Port 8
5.12 64 Bit Read/Write command 9
5.13 SPI Port 9
5.14 Buzzer Command 10
5.15 Auto Clear & Auto Confirm 10
5.16 Pulse Command 11
5.17 H-Bridge Function 11
6 Firmware Communications 12
6.1 Overview 12
6.1.1 Indirect Method – Delcom DLL 12
6.1.2 Direct Method 12
6.2 TX Command Packet Format: 13
6.3 Rx Command Packet Format: 13
6.4 Write Commands 14
6.4.1 Port Write Functions 14
6.4.2 Port Clock Functions 15
6.4.3 Port Setup Functions 16
6.4.4 Feature commands 17
6.4.5 Interrupt and Port Mode 18
6.4.6 Communication Commands 19
6.5 Read Commands 21
7 Specifications 24
7.1 Absolute Maximum Ratings 24
7.2 Electrical Characteristics 24
7.3 Communications 24
8 Package Diagrams 25
9 Ordering Information 26
10 Firmware Release Notes 26
11 Trouble Shooting 27
12 Notes 27
12.1 Power Notes 27
12.2 Interfacing 27
13 Examples 28
13.1 C++ Example Direct Example 28
14 References 31
14.1 Documentation 31
14.2 Examples 31
Optional TID and SID Test 32
Appendix A. Revision History 33
Appendix B. Notices 34

Delcom Products Inc. 200 William St STE302 – Port Chester NY 10573 USA (914)934-5170 www.delcomproducts.com

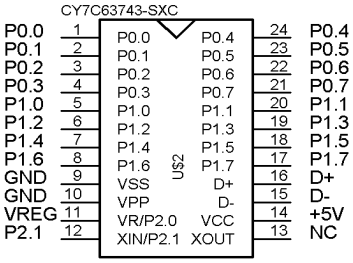
1 Functional Overview

The Delcom USB HID IO chips provide a preprogrammed low cost solution to USB peripherals. These chips are based on the Cypress™ CY7C637xx USB chips. The USB HID IO chip are preprogrammed. The preprogrammed firmware handles all the USB communications and offers the user a rich set of functions. These functions include basic IO to advanced functions such as I2C and SPI.

18-Pin
DIP/SOIC



24-Pin
DIP/SOIC



Delcom Products Inc. 200 William St STE302 – Port Chester NY 10573 USA (914)934-5170 www.delcomproducts.com

2 Pin Definitions

Name	I/O	822270 822370 18-Pin	822670 822770 24-Pin	Description (Alternate function)
P0.0	I/O	1	1	Port 0 bit 0 (I2C SCLK)
P0.1	I/O	2	2	Port 0 bit 1 (I2C SDA)
P0.2	I/O	3	3	Port 0 bit 2
P0.3	I/O	4	4	Port 0 bit 3
P0.4	I/O	18	24	Port 0 bit 4
P0.5	I/O	17	23	Port 0 bit 5 (SPI MISO)
P0.6	I/O	16	22	Port 0 bit 6 (SPI MOSI)
P0.7	I/O	15	21	Port 0 bit 7 (SPI SCLK)
P1.0	I/O	5	5	Port 1 bit 0 (Clock and PWM)
P1.1	I/O	14	20	Port 1 bit 1 (Clock and PWM)
P1.2	I/O	-	6	Port 1 bit 2 (Clock and PWM)
P1.3	I/O	-	19	Port 1 bit 3 (Clock and PWM and Buzzer)
P1.4	I/O	-	7	Port 1 bit 4
P1.5	I/O	-	18	Port 1 bit 5
P1.6	I/O	-	8	Port 1 bit 6
P1.7	I/O	-	17	Port 1 bit 7
P2.1/XIN	I	9	12	Port 2 bit 1 (Input only, weak pull down)
XOUT	O	10	13	Clock Out (Note used)
VREG	O	8	11	3.3 VReg – Use for D- pull up
D+	I/O	13	16	USB Data +
D-	I/O	12	15	USB Data -
VPP	-	7	10	Programming voltage, Connect to VSS
VCC	-	11	14	Voltage Supply +5Volts
VSS	-	6	9	Ground

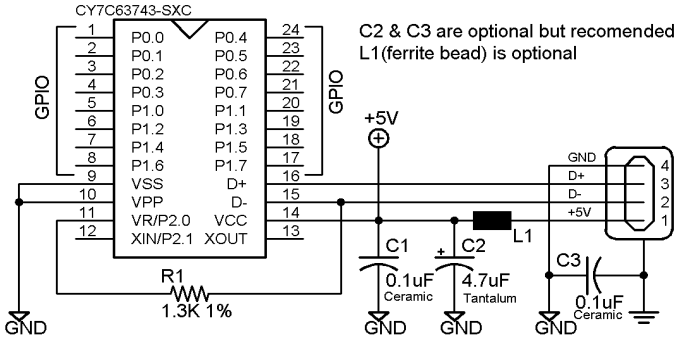
3 Pin Descriptions

Name	**Description**
VCC	Voltage Supply. Nominal 5V, Range 4.0Volts to 5.25Volts
VSS	Ground. Connect to ground
XIN,XOUT	Not used
P0.0-7	Port 0. GPIO. Programmable drive with 14K pullup
P1.0-7	Port 1. GPIO. Programmable drive with 14K pullup
P2.1	Port 2. GPIO. Input Only with weak pull down.
P1.0-7	Port 1. GPIO. Programmable sink current & pullup.
D+,D-	USB data lines. D- requires an external 1.3K resistor to the VReg pin.
VREG	+3.3 Volt Reference
VPP	Programming pin. Must be connected to ground.

4 Circuit Layout

4.1 Typical Schematic

Below is a typical schematic of the USB chip and required external parts. At a minimum you will need to install at least the R1 and C1. C2, C3 and L1 (ferrite bead) are optional. C2 and C3 are recommended. Include C2 when the total current draw is over 25mA. Include C3 for ESD rejection and increase reliability. Include L1 to reduce EMI. Note an external crystal or resonator is not required. The 6Mhz clock is now incorporated internally in the chip. Furthermore pin P2.1 can be used as an extra input.



4.2 Backward Compatibility

The USB HID chips are backwards compatible with the older type circuit layout used in generation I chips. The generation I chips used a 7.5K pull up on D- to +5V, this configuration can still be used. But the recommended circuit above improves reliability. Also the crystal or resonator used in generation I chips is not longer used.

5 Programmed Features

5.1 Basic I/O

The USB HID I/O chips provide general 8 bit input/output commands as well as individual set and reset commands of each pin. See write command 1,2,10,11, 12 and read commands 100.

5.2 Write data with Strobe

See write commands 13, 14, 15 and 16.

The write strobe feature allows the USB I/O chip to interface to another device by using a standard 8-bit data bus with a strobe pin. The data is placed on port 0 and the strobe is selectable on one of the port 1 pins. Theses functions allow one to eight data bytes to be sent on either a positive and negative strobe (pulse). The write strobe functions support an optional acknowledge signal.

Commands 13 and 14 produces the following sequence; 1) Data in is written to Port 0. 2) The strobe pin is toggled active for 1.5us. Optionally, if the acknowledge pin is enabled the strobe pin will wait while the acknowledge pin is held low (See command 10-40 bit 3). 3) Then the strobe pin is toggled non-active. 4) And finally 0xFF is written to Port 0. The strobe pin and the data on port 0 must be initially preset to there no active states before using this function. Port0 should be preset to 0xFF.

Commands 15 and 16 produces the following sequence; 1) Data in Data Extension is written to Port 0 LSB first. 2) The strobe pin is set active for 1.5us. If the acknowledge pin is enabled the strobe pin will wait while the acknowledge pin is held low (See command 10-40 bit 3). 3) Then the strobe pin is made non-active. 4) And finally 0xFF is written to Port 0. 4) System then delays for the specified time set in Data LSB byte. 5) Then the process is repeated till all data bytes in the Data Extension have been sent. The delay is equal to 8.25us+(0.75us*DelayValue) Example: Command 8,18,10,15,10,1,4,0,0,0,0 will send 4 bytes of data (all zeros here) on a high strobe on pin one of port one with a delay of 15.75us. The strobe pin and the data on port 0 must be initially preset to there no active states before using this function.

5.3 Read Data with Strobe

See read commands 1 and 2.

The read strobe feature allows the USB I/O chip to interface to another device by using a standard 8-bit data bus with a strobe pin. The data is captured on port 0 with a active strobe on port1. The strobe pin can be either active high or low. Note before using this command, users should preset Port1 to 0xFF to place Port1 in input mode. The read data strobe command will produce the following sequence. 1) The selected strobe pin on Port 1 is made active. 2) Delay for 1.5us. 3) Data us latched on Port0 and stored. 4) The strobe pin is released.

5.4 Clock Generator

These functions generate a clock source with variable frequency and duty cycle. Up to four separate clocks can be configured. The clock outputs can be selected on port 1 pins 0

through 3. Clock pins can be preset to a predefined state. Use command 20 to enable this feature.

See write commands 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29.

Frequency and duty commands (21-24). The LSB data value sets the period when the port pin is high and the MSB data value sets the period when the port pin is low. The default resolution of the period is 10ms, but this can be changes with the prescalar command (19). The resolution of the duty cycle is 0.39 percent.

5.5 PWM

This feature allows for a PWM function on ports1 pins P1.0 though P1.3. Use write command 34 to configure the PWM feature. To enabled the PWM feature on a pin you must write a low(zero) to that pin. A PWM value of 100(100%) will keep the pin low infinitely. Any value less than 100 will produce a PWM on the selected port pin. To disable the PWM function you must set the duty to 100 (100%). When the duty is set to 100 (100%) the pin function is it's normal state (high or low depend on how it is set). See write command 34.

5.6 Port Setup

These features allow the user to place the I/O in one of 4 modes. Each pin on ports 0 and 1 can be set independently. The boot up default is mode C. To change the port pin modes use write commands 45-48.

GPIO Mode table

Mode	Mode 1 Value Command 46,48	Mode 0 Value Command 45,47	Port type when data out is low	Port type when data out is high
A	0	0	Hi-Z / CMOS	Hi-Z / TTL
B	0	1	Medium (8mA) Sink / CMOS	High (30mA) Drive / CMOS
C – Default *Boot up State*	1	0	Low (2mA) Sink / CMOS	Pull up (14K) / CMOS
D	1	1	High (50mA) Sink / CMOS	High (30mA) Drive / CMOS

Maximum cumulative source drive current for all GPIO is 30mA.
Maximum cumulative sink drive current for all GPIO is 70mA.
See http://www.delcomproducts.com/downloads/cy7c637xx-B.pdf for more GPIO details.

5.7 Read Buffer

This feature allows the USB I/O chip to interface to a device using a standard 8-bit data bus and a read strobe pin. Data is read on port 0 with a read strobe (pulse) on one of the selectable port 1 pins. The data read buffer is 7 bytes deep. If the read data buffer is full, new data will not be accepted and the over flow flag will be set. Note this function cannot be used while the RS232 functions are in uses. See read command 5.

5.8 Scratch Pad

The scratch pad allows the user to write 8 bytes of user defined information in to the USB I/O device. This area can be used for storing user variables, states or other information. Note this function cannot be used while the RS232 functions or Read Buffer functions are in uses.

5.9 Event Counter

The event counter feature allows the counting of events on one or more of the port 0 pins. The resolution of the counter is 4 bytes. Counting is done on either a rising or falling edge. Active edge is set up with write command 43 and enabled with write command 38. The actual counted value is returned with read command 8.

5.10 RS232 Serial Port

The RS232 functions allow the chip to interface to a RS232 compliant device. Currently the baud rate is fixed at 2400bit/sec with 8 data bit, one stop bit and no parity. To use the RS232 function first enable it with commands 10-40, then use command 10-50 to send data and 11-50 to receive data. You can check the internal buffer count with command 11-9. The RS232 pins are fixed with transmit at port 0 pin 7, receive at port 0 pin 6 and clear to send at port 0 pin 5. This command supports a maximum transfer of 7 bytes per command.

5.11 I^2C Port

The I2C functions allow the chip to interface to an I2C compliant device. The I2C port supports the standard clock rate of 100 KHz. The SCLK signal is on port 0 pin 0 and the SDA signal is on port 0 pin 1. There are two write commands; 60-Write and 63-Selective Read Setup. There are three read commands; 60-Read, 61-Selective 8bit Address Read, and 62-Selective 16bit Address Read. This command supports a maximum transfer of eight bytes per command. If any error occur during the I2C communications bit 4 of byte 7 is set (see read command 9).

In firmware version 23 and above we added the ability to change the output drive mode of the I2C pins. The default power up mode is CMOS. This can be changed to open drain using the 110 write command, bit 0. The open drain mode is useful when connected to 3.3volt I2C devices.

Generic I2C Write. – Writes up to 8bytes of data.
Usage: Write Cmd 60: LSBData=Add/Cmd(B0), MSBData=Length(bytes to send), ExtData[0..7]=Data to write.
I2C output:
[START] [Add/Cmd(B0)] [Write ExtData0..7] [STOP]

Generic I2C Read. - Reads up to 8bytes of data. This action requires two commands, first setup the command with write command 63, then call the read 60 command.
Usage: Write Cmd 63: LSBData=0, MSBData=Length(bytes to read), HidData(0)=Add/Cmd(B0), HidData(1)=0, HidData(2)=0, HidData(3)=0
Usage: Read Cmd 61: Read the data.
I2C output:
[START] [Add/Cmd(B0)] [Read Data0..7] [STOP]

Generic I2C 8 Bit Address Selective Read. - Reads up to 8bytes of data at a specific 8bit address or register. This action requires two commands, first setup the command with write command 63, then call the read 61 command.
Usage: Write Cmd 63: LSBData=0, MSBData=Length(bytes to read), HidData(0)=Add/Cmd(B0), HidData(1)=AddressLSB(B1), HidData(2)=0, HidData(3)=RdSelCmd(B2)
Usage: Read Cmd 61: Read the data.
I2C output:
[START] [Add/Cmd(B0)] [AddressLSB(B1)] [START] [RdSelCmd(B2)] [Read Data0..7] [STOP]

Generic I2C 16 Bit Address Selective Read. - Reads up to 8bytes of data at a specific 16bit address or register. This action requires two commands, first setup the command with write command 63, then call the read 62 command.
Usage: Write Cmd 63: LSBData=0, MSBData=Length(bytes to read), HidData(0)=Add/Cmd(B0), HidData(1)=AddressLSB(B1), HidData(2)= AddressLSB(B2), HidData(3)=RdSelCmd(B3)
Usage: Read Cmd 62: Read the data.
I2C output:
[START] [Add/Cmd(B0)] [AddressLSB(B1)] [AddressMSB(B2)] [START] [RdSelCmd(B3)] [Read Data0..7] [STOP]

5.12 64 Bit Read/Write command

The 64 bit read/write commands allows the user to read or write 64 bits (8 Bytes) of data with one command to eight 8-bit hardware latches. Theses commands require extra hardware. See the USB64BIO-Sch.pdf schematic on our website. The command writes a 3bit address on port1 then writes or reads a byte value on Port 0 with a wr/rd strobe on Port 1.

5.13 SPI Port

The SPI functions allow the chip to interface to an SPI compliant device. The I2C port supports a variable clock period from 20ns to 5.1ms. The default clock is 200ns and can be changed with write command 91. The SCLK signal is on port 0 pin 7, the MOSI signal is on port 0 pin 6 and the MISO signal is on port 0 pin 5. This command supports a maximum transfer of eight bytes per command. There are two SPI commands they are 90-Write SPI Data and 90-Read SPI Data.

The write 90 SPI command will send up to 64 bit of data over the SPI bus. At the same time this command is writing the data out on the MOSI pin, the input values on MISO is captured and can be later read with read command 90.

In firmware version 23 and above we added the ability to change the output drive mode of the SPI pins. The default power up mode is CMOS. This can be changed to open drain using the 110 write command, bit 1. The open drain mode is useful when connected to 3.3volt I2C devices.

5.14 Buzzer Command

This feature is intended to be used to drive a buzzer or other auditory device. The feature is fixed on port 1 pin 3 (also H-Bridge Option below). The command number for this feature is 70. The frequency, duty cycle and repeat value are all programmable. The frequency is programmed by setting the buzzer's frequency time variable, the units are in 256us. For example a desired buzzer frequency of 1KHz would yield a frequency value of around 4. The buzzer's on time and off time variables are used to program the duty cycle of the buzzer. These units are in 50ms. If you want the buzzer to turn on and off every second you would program 10 for the on time and off time. The repeat value dictates what mode the buzzer will be in. If a value of zero is used for the repeat value then the buzzer will sound continuously at the frequency specified until the user turns it off. If a value of 255 is used then the buzzer will sound at the frequency and duty cycle specified until the user turns it off. If any other value is used the buzzer will sound at the frequency and duty cycle specified and repeat for that many times. The DataLSB turns this feature on (1) or off (0). The DataMSB sets the frequency. The DataExt[0] sets the repeat value. The Data Ext[1] sets the on time. And the Data Ext[2] sets the off time.

Freq Value	Freq(Hz)	Freq Value	Freq(Hz)	Freq Value	Freq(Hz)
1	3906	5	781	9	434
2	1953	6	651	10	390
3	1302	7	558	11	355
4	976	8	488	12	325

```
// Buzzer Example (Freq=3906Hz DutyOn=200ms DutyOff=100ms Repeat=3)
MajorCmd =102;               // note this is a 16byte command
MinorCmd =70;                // Buzzer Command
LSBData = 0x01;              // Turn buzzer on
MSBData = 0x01;              // Set the frequency
DataExt0 = 3;                // Repeat 3 times
DataExt1 = 4;                // On Duty 200ms
DataExt2 = 2;                // Off Duty 100ms
```

H-Bridge Option – In version 25 and above the buzzer function can optionally drive an H-Bridge circuit. See H-Bridge function below. When the H-Bridge controller has been set to mode 1 or mode 2, this function will driver the H-Bridge pins (P0.0-P0.3) instead of P1.3. Also while the buzzer function is running P1.1 is automatically brought low and returned high when the buzzer function terminates. The P1.1 is intend to be connected to the H-Bridge power enable circuit.

5.15 Auto Clear & Auto Confirm

The Auto Clear and Auto Confirm feature are intended for the Delcom USB indicator light. They use the built in button of the indicator (on port 0 pin 0) to either on off the led lights (AutoClear) and/or to sound the buzzer in the indictor light when the button is pressed (Auto Confirm). In order to use this feature you must first enable the event

counter (see write command 38). The write command number 72 controls these feature. DataLSB bit 6 enables or disables the Auto Clear feature. And DataLSB bit 7 enables or disables the Auto Confirm feature

5.16 Pulse Command

This command allows the user to send a custom pulse stream on port 0 or port 1. The command number is 76. All 8bits on either port0 or port1 can be changed. The LSBData parameter contains the delay prescaler and the port select bit. Bit 7 of the LSBData selects the port, a low selects port 0 and high selects port 1. The remaining bits 6 through 0 hold the prescaler value. The prescaler range is 0 to 127. The delay between the states is equal to (DelayValue+1) x Prescalar x ~2us. There are 5 port pin state change parameters and 4 delay parameters. The change the port data parameters change the port value by executing a XOR with the current port value and the StateXPortXORData value. So to toggle a pin set the StateXPortXORData bit value high. You can toggle as many pins as you like. Up to 5 states can be set, for less than 5 states set the remaining data to all zeros. The initial port value should be preset with the write port command. Note this command processes inline and therefore no other command will be processed till this command terminates.

```
// Pulse Command example
MajorCmd =102;              // note this is a 16byte command
MinorCmd =76;               // Pulse Command
LSBData = 0x01;             // PortSelect=Port0 and Prescaler=1
MSBData = 0x01;             // S0PortXORData - Toggle P0.0
DataExt0 = 10;              // S0Delay – delay for 10 x 1 x 2us = 20us
DataExt1 = 0x02;            // S1PortXORData - Toggle P0.1
DataExt2 = 5;               // S1Delay – delay for 10 x 1 x 2us = 20us
DataExt3 = 0x02;            // S2PortXORData - Toggle P0.1
DataExt4 = 5;               // S2Delay – delay for 10 x 1 x 2us = 20us
DataExt5 = 0x02;            // S3PortXORData - Toggle P0.1
DataExt6 = 10;              // S3Delay – delay for 10 x 1 x 2us = 20us
DataExt7 = 0x03;            // S4PortXORData - Toggle P0.0 & P0.1

States    0    1    2    3    4
Port0.0 __--------------------____
Port0.1 _______-----_____-----____
```

5.17 H-Bridge Function

The H-Bridge control functions is intended to drive an H-Bridge circuit. The H-Bridge function can be used to drive DC motor or a buzzer. Pins P0.0 through P0.3 are used to control the H-Bridge. Pin P0.0 drives the bottom right transistor, P0.1 drives the top right transistor, P0.2 drives the top left transistor and P0.3 drives the bottom left transistor. The active level is low. The H-bridge mode is set with the write command 71. The LSB Data parameter is used to set the mode. There are 4 modes; 0=off (All pins high), 1=State 1 (forward), 2=State 2(reverse), 0xFF=Brake (both bottom drivers low). Default boot up pins values are all high.

6 Firmware Communications

6.1 Overview

There two ways to communicate with the USB HID device. They are the direct and indirect methods. The direct method communicates directly to the USB driver via the OS API functions. The indirect method communicates to the USB HID device via the Delcom DLL. Using the Delcom DLL is the easiest way of communicating with the USB HID device and is backwards compatible with the older generation I USB Chips Sets. You can also mix indirect and direct functions. For example you can use the DLL to get the device name and then you the direct method to open and write/read to the device.

6.1.1 Indirect Method – Delcom DLL

Using the Delcom DLL is the simplest method to communicate with the USB device. Simply copy the DLL in your program executing directory (e.g. bin/debug) or to a searchable system path (e.g. windows/system32). Then including the prototyping or declaration file in your project and start calling the DLL functions. The DLL also offers enhanced functions that simplify the communication with the USB device. See the Delcom DLL for more information.

```
// Simply C# Delcom DLL Example
public class Main
    {
    unsigned int hDevice = 0;
    int Result;
    byte Port0, Port1;
    StringBuilder DeviceName = new tringBuilder("",Delcom.MAXDEVICENAMELEN);

    cout << "Welcom to the Delcom CS DLL example.\r\n";
    cout << "DelcomDLL Ver="+ Delcom.DelcomGetDLLVersion().ToString()+"\r\n";

    Result = Delcom.DelcomGetNthDevice(Delcom.USBIODS, 0, DeviceName);
    if (Result == 0) cout << "DelcomGetNthDevice - Failed to find device!\r\n";
    else { // ok, device found, now try to open it.
      cout << "Device Found: " + DeviceName.ToString() + "\r\n";
      hDevice = Delcom.DelcomOpenDevice(DeviceName, 0); // Open the device
      // Read ports 0 and 1
      Delcom.DelcomReadPorts(hDevice, out Port0, out Port1);
      cout << "Port0 = " + Port0.ToString();
      cout << "Port1 = " + Port1.ToString();
      // Now set Port0 to 0xFF and Port1 to zero
      Delcom.DelcomWritePorts(hDevice, 0xFF, 0x00);
      Delcom.DelcomCloseDevice(hDevice);
      }
  }
```

6.1.2 Direct Method

In order to communicate to the USB device you must first finds its device name in the system. To find the device name we use the system API functions to request all USB device witch have the unique USB HID GUID. There are numerous USB HID devices installed in most machines, so you will have to loop through each device found and check it for the correct Vendor ID (VID) and Product ID (PID). Also at this time we can further reduce our search by

check the Family ID (FID) and/or the unique serial number (SID) in each Delcom USB HID Chip. The device name can change each time the USB device is plugged in, so you must search for the device name each time.

```
USB HID GUID:   {4D1E55B2-F16F-11CF-88CB-001111000030}
Delcom USB VID: 0x0FC5
Delcom USB PID: 0xB080
Delcom USB FID: 1=IOChips, 2=VisualIndicators, 3=NumericDisplays,
                4=PanelLigths, 5=Buzzers,
Delcom USB SID: Unique Serial Number
```

Once the device name has been found, we use it to create or open the device. The open function returns a handle to the device. This handle is then used to communicate with the device and to finally close the device when we are done with it. In Windows use the CreateFile() functions to open the device. Note some OS required USB HID device to always be opened in a shared mode.

```
//In MS Windows we would use the following functions:
GetHidGuid()                          // Returns the unique USB HID GUID
SetupDiGetClassDevs()                 // Returns the device details
SetupDiEnumDeviceInterfaces ()        // Returns the device interface
SetupDiGetDeviceInterfaceDetail()     // Returns the interface details
CreateFile()                          // Opens the device
HidD_GetAttributes()                  // Returns the device attributtes
HidD_SetFeature                       // Writes to the device
HidD_GetFeature                       // Read from the device
CloseFile() or CloseHandle()          // Closes the device
```

Writing and reading to the device is done by passing a small buffer to the write and read system API functions. The buffer we call a data packet. This data packet is preset by the user to the command numbers and parameters we want and then sent down to the USB device. The USB device acts on the data packet and in a read commands returns data in the same data packet that we passed. All data packets are at least 8 byte long. The transmit(TX) and receive(RX) data/command packets are defined below.

6.2 TX Command Packet Format:

Major Command	1 Byte
Minor Command	1 Byte
Data LSB	1 Byte
Data MSB	1 Byte
DataHID[0..3]	4 Bytes
DataExt[0..7]	8 Bytes(Optional)

6.3 Rx Command Packet Format:

ReadData[0..15]	8-16 Bytes

6.4 Write Commands

This section describes the USBIOHID write commands. There are currently two major commands (101 and102) for the write command/data functions. A major command of 101 will send an 8 byte write command. A major command of 102 will send a 16 byte write command. In the future we will add two more commands to send 32 and 64 byte write commands. Note you can always send a higher number of bytes of a write command, but not the opposite.

```
MajorCmd = 101 – Send an 8 Byte write command.
MajorCmd = 102 – Send an 16 Byte write command.
```

6.4.1 Port Write Functions

MinorCmd = 0 Length = 8 or 16 Bytes	***0 - Test function***. Writes the command/data packet to the device command/data buffer. Uses for testing purposes, command does nothing. The written command/data packet can be read with read commands 101 or 102.
MinorCmd = 1 LSBData = Port0Value Length = 8 Bytes	***1 – Write Port 0.*** Writes the LSB data to Port0
MinorCmd = 2 LSBData = Port1 Length = 8 Bytes	***2 – Write Port 1.*** Writes the LSB data to Port1
MinorCmd = 3 LSBData =ReadStrobe MSBData = Port1Value Length = 8 Bytes	***3 – Setup Read Strobe.*** The LSB data value sets the read strobe pin to be used in read commands 1 and 2. See read commands 1 and 2.
MinorCmd = 8 LSBData = LSBWrite MSBData = MSBWrite Length = 8 Bytes	***8 – Setup 64bit Write 2Bytes Read 8bytes..*** The LSB data value LSB write data and the MSB data value sets the MSB write data. See read command 18
MinorCmd = 10 LSBData = Port0Value MSBData = Port1Value Length = 8 Bytes	***10 – Write both the port 0 and port 1 values***. The LSB data is written to Port0 and the MSB data is written to Port1.
MinorCmd = 11 LSBData = Port0Reset MSBData = Port0Set Length = 8 Bytes	***11 - Sets or resets the port 0 pins individually***. The LSB resets the corresponding port pin(s) and the MSB sets the corresponding port pin(s) on port 0. Resetting the port pin(s) takes precedence over setting the bits.
MinorCmd = 12 LSBData = Port1Reset MSBData = Port1Set Length = 8 Bytes	***12 - Sets or resets the port 1 pins individually*.** The LSB resets the corresponding port pin(s) and the MSB sets the corresponding port pin(s) on port 1. Resetting the port pin(s) takes precedence over setting the bits.
MinorCmd = 13 LSBData = Port0Data MSBData = Port1Strobe	***13 - Write strobe high function.*** This command writes the LSB to port 0 and then toggles the corresponding pin marked in the MSB byte high then low.

Length = 8 Bytes	
MinorCmd = 14 LSBData = Port0Date MSBData = Port1Strobe Length = 8 Bytes	***14 - Write strobe low function***. This command writes the LSB to port 0 and then toggles the corresponding pin marked in the MSB byte low then high.
MinorCmd = 15 LSBData = Delay MSBData = Port1Strobe ExtData0..7 = DataBuffer Length = 16 Bytes	***15 - Write 8-byte strobe high function.*** This command writes the Data Extension data to port 0 and then toggles the corresponding pin marked in the MSB byte high then low and then delays for the specified time set in the LSB byte.
MinorCmd = 16 LSBData = Delay MSBData = Port1Strobe ExtData0..7 = DataBuffer Length = 16 Bytes	***16 - Write8-byte strobe low function***. This command writes the Data Extension data to port 0 and then toggles the corresponding pin marked in the MSB byte low then high and then delays for the specified time set in the LSB byte.
MinorCmd = 17 ExtData0..7 = DataBuffer Length = 16 Bytes	***17 - Write 64 Bit Command***. This command writes 8 bytes of data to the external hardware latches. The data is passed in the data extension registers. The LSB of the data extension is written to address zero. This command requires external hardware. See USB64BIO-Sch.pdf on our website.

6.4.2 Port Clock Functions

MinorCmd = 19 LSBData = Prescalar Length = 8 Bytes	***19 - Loads the Clock Generator Global Prescalar value.*** This value is passed in the LSB register. Increasing this number decreases all the clock function frequencies. Prescalar range is 1 to 255 and the boot up default value is 10.
MinorCmd = 20 LSBData = Disable MSBData = Enable Length = 8 Bytes	***20 - Enables or disables the clock generator on port 1.*** The lower nibble of the LSB disables the corresponding port pin(s) and the lower nibble of the MSB enables the corresponding port pin(s). Disabling the port pin(s) takes precedence over enabling.
MinorCmd = 21 LSBData = HighDuty MSBData = LowDuty Length = 8 Bytes	***21 - Loads the frequency and duty cycle for port 1 pin 0.*** The LSB data sets the high duty cycle and the MSB data sets the low duty cycle on port 1 pin 0.
MinorCmd = 22 LSBData = HighDuty MSBData = LowDuty Length = 8 Bytes	***22 - Loads the frequency and duty cycle for port 1 pin 1.*** The LSB data sets the high duty cycle and the MSB data sets the low duty cycle on port 1 pin 1.
MinorCmd = 22 LSBData = HighDuty MSBData = LowDuty Length = 8 Bytes	***22 - Loads the frequency and duty cycle for port 1 pin 2.*** The LSB data sets the high duty cycle and the MSB data sets the low duty cycle on port 1 pin 2.
MinorCmd = 23 LSBData = HighDuty MSBData = LowDuty Length = 8 Bytes	***23 - Loads the frequency and duty cycle for port 1 pin 3.*** The LSB data sets the high duty cycle and the MSB data sets the low duty cycle on port 1 pin 3.
MinorCmd = 25	***25 - Synchronizes the clock generation***. This command synchronizes

LSBData = Enable MSBData = Preset Length = 8 Bytes	all the clock generators to start now plus an initial phase delay, see below. The lower nibble of the LSB enables this function on the corresponding pins P1.0 to P1.3. The lower nibble of the MSB presets the initial value on the corresponding pins P1.0 to P1.3. Initial phase delay resolution is in 10ms and is passed in the LSB register. Initial phase delay registers are cleared after this command is sent. Therefore the initial phase delay registers must be set each time this command is called.
MinorCmd = 26 LSBData = DelayValue Length = 8 Bytes	***26 - Load initial phase delay on port 1 pin 0.*** Sets the LSB data as the port1 pin0 initial delay value. To be used with command 25.
MinorCmd = 27 LSBData = DelayValue Length = 8 Bytes	***27 - Load initial phase delay on port 1 pin 1.*** Sets the LSB data as the port1 pin0 initial delay value. To be used with command 25.
MinorCmd = 28 LSBData = DelayValue Length = 8 Bytes	***28 - Load initial phase delay on port 1 pin 2.*** Sets the LSB data as the port1 pin0 initial delay value. To be used with command 25.
MinorCmd = 29 LSBData = DelayValue Length = 8 Bytes	***29 - Load initial phase delay on port 1 pin 3.*** Sets the LSB data as the port1 pin0 initial delay value. To be used with command 25.

6.4.3 Port Setup Functions

MinorCmd = 30 Length = 8 Bytes	***30 - Enable or disable port 0 pull up resistors***. This command is not supported; it has been left for backwards compatibility. See new port setup commands 44-48.
MinorCmd = 31 Length = 8 Bytes	***31 - Enable or disable port 1 pull up resistors.*** This command is not supported; it has been left for backwards compatibility. See new port setup commands 44-48.
MinorCmd = 33 Length = 8 Bytes	***32 - Setup port 0 pins sink current level.*** This command is not supported; it has been left for backwards compatibility. See new port setup commands 44-48.
MinorCmd = 33 Length = 8 Bytes	***33- Setup port 1 pins sink current level***. This command is not supported; it has been left for backwards compatibility. See new port setup commands 44-48.
MinorCmd = 34 LSBData = Port1PinValue MSBData = PWMValue Length = 8 Bytes	***34 - Load the PWM values.*** Port pins P1.0 through P1.3 can be placed is PWM mode by writing the PWM value with this command. The LSB Data parameter is the port pin number, range is 0-3. The MSB Data parameter is the PWM value, range is 0-100.

6.4.4 Feature commands

MinorCmd = 35 LSBData = Storbe Pin Length = 8 Bytes	***35 - Setup read buffer function.*** This command sets up the micro to read the current values on port 0 when a read strobe in presented on the configured strobe pin on port 1. The LSB will enable the correspond pin on port 1 to latch data on port 0 on the active edge. The active edge is set up the pull ups command 30 and 31. If the pull-ups are enabled then the active transition is from high to low. Otherwise the active transition is from low to high. The read buffer is only 7 bytes deep. Default is 0x00, read buffer disabled. See read buffer command 5. Note this function cannot be used while the RS232 functions are in uses.
MinorCmd = 37 LSBData = Data MSBData = Address[0-7] Length = 8 Bytes	***37 - Write scratch pad area.*** Writes the LSB to the scratch pad. The MSB contains the pointer to the scratch pad. Pointer values can range from 0 to 7. The scratch pad area is 8 bytes deep. This area can be used for storing user variables, states or information. Defaulted to all 0x00 on boot up. Note this function cannot be used while the RS232 functions are in uses.
MinorCmd = 38 LSBData = EnablePin MSBData = DisablePin Length = 8 Bytes	***38 - Enable/Disable Events Counter.*** This command sets up the event counter. LSB data byte enables this function on the corresponding pin on port 0. The MSB data byte disabled this function on the corresponding pin on port 0. Once enabled the system will count events on the enabled pin on the active edge. The active edge is configured by the pull ups command 10-30 and 10-31. If the pull-ups are enabled then the active transition is from high to low. Otherwise the active transition is from low to high. The event counter value is read with command 11-8. This feature is off by default.
MinorCmd = 40 LSBData = CtrlReg Value Length = 8 Bytes	***40 - Enable/Disable Control Register.*** This function sets the control register value. Each bit in this register controls different options. The LSB data byte sets the control register. Bit 0: Status LED. When set Port1 pin 3 (P1.3) will toggle low when USB communications are present. Only available on this pin. Bit 1: Enables the RS232 Serial port with fixed 2400 baud rate. Version 5. Bit 3: Enables the acknowledge pin in the write strobe functions 13,14,15 &16. The acknowledge pin is only available on pin P1.2 and is active low. The write strobe will be extended while the acknowledge pin is held low. Version 8. Bits7-4,2: Future Implementation. These bits are reserved for future implementation and should be set to zero for future compatibility.

6.4.5 Interrupt and Port Mode

MinorCmd = 43 LSBData = Port0IntEdge Length = 8 Bytes	***43 - Set Port 0 Interrupt Edge.*** The LSB Data parameter sets the Port 0 Interrupt Edge. 1= Rising edge, 0=Falling edge.
MinorCmd = 44 LSBData = Port1IntEdge Length = 8 Bytes	***44 - Set Port 1 Interrupt Edge.*** The LSB Data parameter sets the Port 1 Interrupt Edge. 1= Rising edge, 0=Falling edge.
MinorCmd = 45 LSBData = Port0Mode Length = 8 Bytes	***45 - Configures Port 0 GPIO – Mode 0 Register*** The LSB data parameter is the value passed. Each bit represents a port pin. See the GPIO Mode table.
MinorCmd = 46 LSBData = Port0Mode1 Length = 8 Bytes	***46 - Configures Port 0 GPIO – Mode 1 Register*** The LSB data parameter is the value passed. Each bit represents a port pin. See the GPIO Mode table.
MinorCmd = 47 LSBData = Port1Mode0 Length = 8 Bytes	***47 - Configures Port 1 GPIO – Mode 0 Register*** The LSB data parameter is the value passed. Each bit represents a port pin. See the GPIO Mode table.
MinorCmd = 48 LSBData = Port1Mode1 Length = 8 Bytes	***48 - Configures Port 1 GPIO – Mode 1 Register*** The LSB data parameter is the value passed. Each bit represents a port pin. See the GPIO Mode table.

6.4.6 Communication Commands

MinorCmd = 50 ExtData[0] = Length ExtData[1..7] = Data Length = 16 Bytes	***50 - Writes to the RS232 Serial Port.*** This command sends data to the serial port. Both the data count and data are passed in the Data Extension. The MSB and LSB bytes should be zero. The data count is in the LSB byte (first byte of the DataExt) and the data is in the remaining 7 bytes. Issuing this command clears the TX Status register (see 11-9). Example command 8,18,0,0,6,5,1,2,3,4,5 will send 5 bytes of data (1,2,3,4,5) to the serial port.
MinorCmd = 60 LSBData = Add/Cmd(B0) MSBData = Length ExtData[0..7] = Data Length = 16 Bytes	***60 - Write to the I2C Port.*** This command writes the data found in the data extension to the I2C device. The device address/command is set in the Data LSB byte and the number of bytes to send is set in the Data MSB byte. If an error occurs bit 4/7 of byte 7 is set in the system parameters bytes, else reset. See read command 9.
MinorCmd = 61 LSBData = Add/Cmd(B0) MSBData = Address(B1) Length = 8 Bytes	***61 -I2C Selective Read Setup 1.*** Note this is an old command left for backwards compatibility with generation I chip sets. It is recommended that you use command 63 instead of this command. This commands setups the selective read command 61. You will also need to call command number 62 to setup the selective read. The Data LSB should be set to the device address/command (byte 0) and the Data MSB should be set to the selective read address (byte 1). Also see read command 61.
MinorCmd = 62 LSBData = RdSelCmd(B2) MSBData = Length Length = 8 Bytes	***62 -I2C Selective Read Setup 2.*** Note this is an old command left for backwards compatibility with generation I chip sets. It is recommended that you use command 63 instead of this command. This commands setups the selective read command 61. You will also need to call command number 61 to setup the selective read. The Data LSB should be set to the device read command (byte 2) and the Data MSB should be set to the read length. Also see read command 61.
MinorCmd = 63 LSBData = 0 MSBData = Length HidData0 = Add/Cmd(B0) HidData1 = AddLSB(B1) HidData2 = AddMSB(B2) HidData3 =RdSelCmd(B2/3) Length = 8 Bytes	***63 -I2C Selective Read Setup.*** This commands setups the selective read command 61. The Data LSB is not used and should be set to zero for future compatibility. The Data MSB is set to the read length in bytes. The HidData0 is set to the I2C device address/command (byte 0). The HidData1 is set to the LSB address of the I2C device. The HidData2 is set to the MSB address of the I2C device (byte 2), this byte is only sent in 16 bit mode. The HidData3 is set to the I2C read command this will be byte 2 or 3 depending on the mode used. Once this command has been sent the user should call the read 61 or 62 command.
MinorCmd = 70 LSBData = On/Off MSBData = Length DataExt[0] = Repeat Value DataExt[1] = OnTime DataExt[2] = Off Time Length = 16 Bytes	***70 – Buzzer Control.*** This command is used to drive a buzzer on port 0 pin3. See the buzzer control description for more information.
MinorCmd = 71 LSBData = Mode Length = 8 Bytes	***71 – H-Bridge Control.*** This command is used to control the H-Bridge functions. The LSBData parameter configures the H-Bridge mode. Values: 0=Off, 1=State 1 (forward), 2=State 2(reverse),0xFF= Brake (both lower drivers are on). See H-Bridge description for more info. *Included in firmware version 25 and above.*

MinorCmd = 72 LSBData = ControlByte Length = 8 Bytes	***72 – Auto Clear & Auto Confirm Control.*** This command enables or disables the Auto Clear and Auto Confirm feature. Bit 6 of the DataLSB controls the Auto Clear feature and bit 7 controls the Auto Confirm feature. A high value enables the feature and low disables it.
MinorCmd = 76 LSBData = Port & Prescalar MSBData = S0PortXORData DataExt0= S0Delay DataExt1 = S1PortXORData DataExt2= S1Delay DataExt1 = S2PortXORData DataExt4= S2Delay DataExt1 = S3PortXORData DataExt6= S3Delay DataExt7 = S4ortXORData Length = 16 Bytes	***76 – Pulse Control.*** This command allows the user to send a custom pulse stream on port 0 or port 1. See the Pulse Command describtion for more information.
MinorCmd = 90 LSBData = Bit Length ExtData[0..7] = Data Length = 16 Bytes	***90 - Write to the SPI port.*** This commands writes up to 8 bytes (64bits) of data (passed in the DataExt) to the SPI port. The number of bit to write is passed in the LSB Byte, range is 1-64. Also see command 11-90 to read the MISO data.
MinorCmd = 110 LSBData = Disbale Bits LSBData = Enable Bits Length = 8 Bytes	***110 – Enable/Disable Option0 Bits..*** This command enables or disables the option0 bits. There are 8 options bits. Set the corresponding bit in LSBData high to disable the option. And set the corresponding bit in MSBData high to enable the option. The option0 default value is zero. Bit 0 – I2C Output Mode. 0=CMOS, 1=Open drain. Bit 1 – SPI Output Mode. 0=CMOS, 1=Open drain. Bit 3-7 Reversed for future use. *Included in firmware version 23 and above.*

6.5 Read Commands

This section describes the USBIOHID read commands. To send a read command send a user defined buffer to the USB HID read function. The first byte passed in the buffer must be preset to the command number. On a successfully return of the USB HID read function, the buffer passed will return the requested read data. Read command can return 8 to 16 bytes of requested data, see individual commands for command read length.

Command = 0	***0 – Not supported.*** See read command 100 for read ports command. .
Command = 1 **Length = 8 Bytes**	***1- Reads port 0 with High strobe.*** Reads the current data on port 0 with a high strobe on pin X on port 1. The LSB sets up which pin is to be used for the high strobe. See Read port 0 with strobe sequence below.
Command = 2 **Length = 8 Bytes**	***2 - Reads port 0 with Low strobe.*** Reads the current data on port 0 with a low strobe on pin X on port 1. The LSB sets up which pin is to be used for the low strobe. See Read port 0 with strobe sequence below.
Command = 5 **Length = 8 Bytes**	***5 - Reads the Read Buffer.*** This command is setup with the read Buffer Setup Command 35. The LSB byte returned is the read buffer status byte, it will contain the number of bytes available in the read buffer. The next 7 bytes contain the data. The read data buffer is only 7 bytes deep. Data is filled from byte 1 to byte 7. If the read data buffer is full and another read strobe occurs before this command is read. Then the read buffer status byte will be set to 0xFF and the new data byte would be lost. The user must check the read status byte to see if; new data is present, not present or present with data over flow. This command resets the read status byte to zero. Note this function cannot be used when the RS232 function is in use.
Command = 7 **Length = 8 Bytes**	***7 - Reads the 8 bytes in the scratch pad area.*** Default values are zero.
Command = 8 **Length = 8 Bytes**	***8 - Reads the event counter value.*** This command returns the 4 byte event counter value and then resets the counter. If the counter over flows then the over flow status byte will be set to 0xFF otherwise it will be 0x0. The event counter is returned in the first 4 bytes and the over flow byte is in the 5 byte.
Command = 9 **Length = 8 or 16 Bytes**	***9 - Reads system variables.*** This function returns the following system variables. This command can be read as an 8 or 16 byte command. Byte0: Control Register. Byte1: Clock Generator Pre-Scalar. Byte2: Port 0 Pull Up Register. Byte3: Port 1 Pull Up Register. Byte4: USB Port Address. Byte5: RS232 Rx Status. Returns the available data count in the lower nibble. Bit 7of 7 is set on Rx Buffer overflow and bit 6/7 is set on Rx framing error. Byte6: RS232 Tx Status. The lower nibble returns the number of data bytes still pending in the Tx buffer. Bit 7of 7 is set on a Tx buffer overflow. Byte7: Bit 4/7 is set if an I2C error is detected. This bit is update each time an I2C function is called. Byte8: Option0 – See write command 110 for more information. Byte9-15: Reversed for future use.

Command = 10 **Length = 8 Bytes**	***10 - Reads the firmware information.*** Byte 0-3: Unique Device Serial Number. DWORD Little Endian. Byte 4: Firmware Version. Byte 5: Firmware Date. Byte 6: Firmware Month. Byte 7: Firmware Year.
Command = 12 **Length = 8 Bytes**	***12 - Reads 8 bytes of memory data.*** This is peek functions used only for firmware debugging. Address was be preset with write command 3.
Command = 17 **Length = 8 Bytes**	***17 - Read 64 Bit Command.*** This command reads 8 bytes of data from the external hardware. The LSB of the returned data is address zero. This command requires external hardware. See USB64BIO-Sch.pdf on our website.
Command = 18 **Length = 8 Bytes**	***18 - Write 2 bytes, Read 8 byte Command.*** This command reads 8 bytes of data from the external hardware, similar to the above command. But the data setup in write command 8, is write to address 0 and 1. See write setup command 8. This commands requires external hardware. See USB64BIO-Sch.pdf on our website.
Command = 50 **Length = 8 Bytes**	***50 - Reads the RS232 Rx Buffer.*** This byte returns 8 bytes, the first byte is the Rx Buffer Status and data count and the remaining bytes are the RS232 data bytes. The Rx buffer is 7 bytes deep and is in LSB first order. The Rx Status and data count byte are cleared when this command is issued. The lower nibble of the status byte contains the Rx buffer data length count, pin 7of 7 of the rx status byte is set on an Rx overflow and pin 6 of 7 is set on a Rx framing error. Note you can read both the Rx Status and Tx Status bytes with command 11-9 without clearing there content.
Command = 60 **Length = 8 Bytes**	***60 -Reads from the I2C Port.*** Reads 1 to 8 bytes of data from the I2C port. This command must be pre setup with write commands 63. If an error occurs bit 4/7 of byte 7 in read command 9 is set, else reset. Also see read command 9.
Command = 61 **Length = 8 Bytes**	***61 - Selective Reads from the I2C Port (8-Bit Address)*** This command is typically used in I2C memory devices. This command first writes a 8bit selective address and then reads 1 to 8 bytes of data from the I2C port. This command must be pre setup with write command 63 (or commands 61 and 62). If an error occurs bit 4/7 of byte 7 in read command 9 is set, else reset. Also see read command 9.
Command = 62 **Length = 8 Bytes**	***62 - Selective Reads from the I2C Port (16-Bit address)*** This command is typically used in I2C memory devices. This command first writes 16bit selective address and then reads 1 to 8 bytes of data from the I2C port. This command must be pre setup with write command 63. If an error occurs bit 4/7 of byte 7 in read command 9 is set, else reset. Also see read command 9.
Command = 90 **Length = 8 Bytes**	***90 - Read SPI Data.*** Returns 8 bytes (64bits) of data captured from the last SPI write command (90). To read data from the SPI port first send Write SPI data command 90 and then send this command.
Command = 100 **Length = 8 Bytes**	***100 - Read ports 0 and port 1.*** This command will read the currently port values. The first byte (LSB) will contain the current value on port 0 and the second byte (MSB) will contain the current value on port 1. The third byte returns the clock enable status on port 1. And the fourth byte returns the Port2 values.
Command = 101 **Length = 8 Bytes**	***101 – Read 8 Byte Command Buffer.*** This command returns the values of the last 8 byte value in the command buffer. This command does not alter the command buffer. The command can be used to double check that your last write or read command was sent or received correctly. This command can be used for testing or for mission critical designs.

Command = 102 **Length = 16 Bytes**	***102 – Read 16 Byte Command Buffer.*** This command returns the values of the last 16 byte value in the command buffer. This command does not alter the command buffer. The command can be used to double check that your last write or read command was sent or received correctly. This command can be used for testing or for mission critical designs.
Command = 103 **Length = 8 Bytes**	***103 – Read Errors. .*** This command will read the currently error value. After this command has been sent the current error value is reset to zero. Return values: Bytes0-1: FamilyCode (1=USB Chips, 2=Visual Indicators, 3=Numeric Display, 4=TLights) Bytes2: Firmware version Byte3: Always zero Byte4: System Error Bit0 – Set on power up. Bit1 – Watch Dog Reset Bit2 – Low Voltage Reset Bit3 – Invalid ISR Reset Bit4 – USB No Data Error Byte5: Command Error Bit0 – Write Command failed or not supported. Bit1 – Read Command failed or not supported. Byte6: Generic Error
Command = 104 **Length = 16 Bytes**	***104 – Read Family.*** This command will return the current family number. The return syntax is as follows. Bytes0-1: FamilyCode (1=USB Chips, 2=Visual Indicators, 3=Numeric Display, 4=TLights, 5=Buzzer) Bytes2-3: Security Code Byte4: Firmware version Byte5: Firmware Date Byte6: Firmware Month Byte7: Firmware Year Bytes8-10: Serial Number

7 Specifications

For more specifications all so see the Cypress™ data sheet CY7C637XX.

7.1 Absolute Maximum Ratings

Storage Temperature	-65C to +150C
Operating Temperature	-0C to +70C
Vss relative to Vcc	-0.5V to +7.0V
DC Input Voltage	-0.5V to Vcc+0.5V
DC voltage on HiZ pins	-0.5V to Vcc+0.5V
Max Current Summed on Port1 pins 60ma	
Max Current Summed on Port0 pins 10ma	
Power Dissipation	300mW
Static Discharge Voltage	>2000V
Latch Up Current	200mA

7.2 Electrical Characteristics

VCC Operating Current	20mA
VCC Limits	4 to 5.25V
Max GPIO Sink Current	70mA*
Max GPIO Source Current	30mA*
Pull Up Resistor	24Kohms

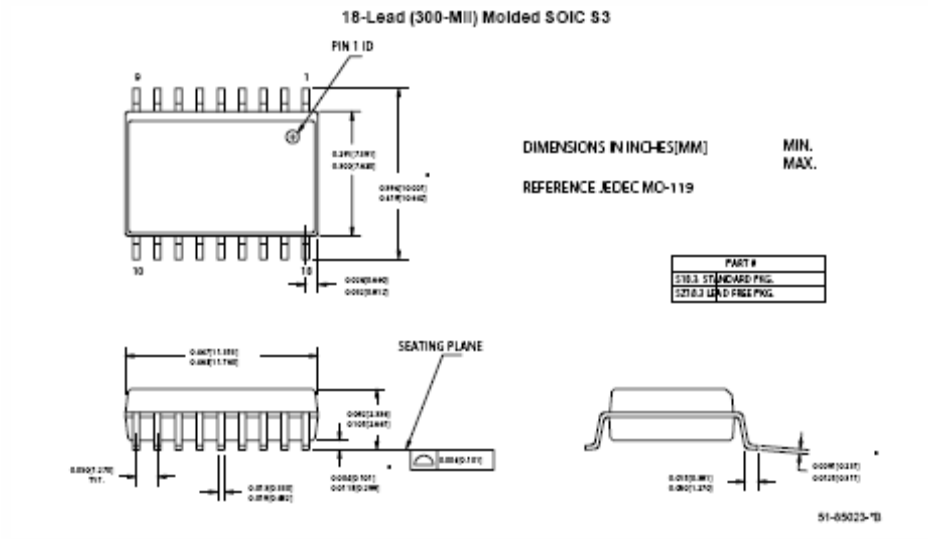
* Cumulative across all ports.

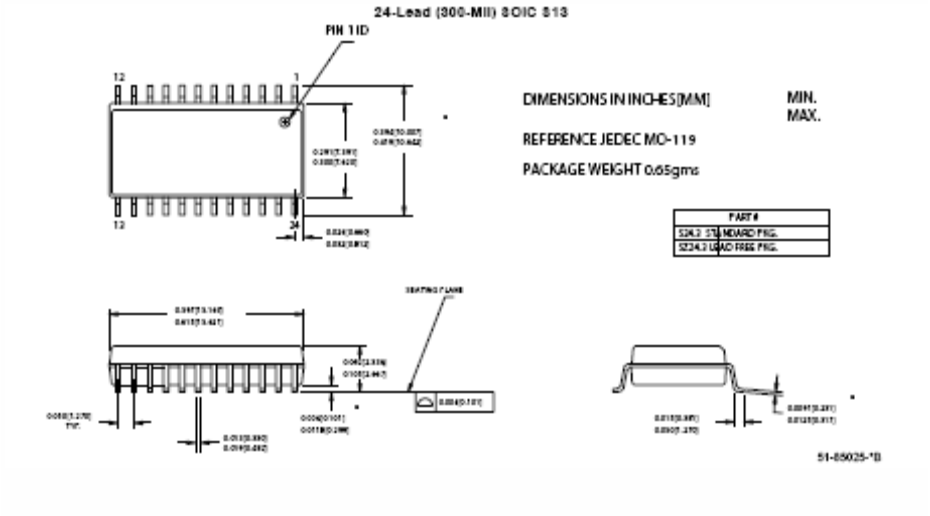
7.3 Communications

Packet Bandwidth*	100 Packet/sec

* The USB 1.1 Low Speed specification defines the maximum packet period to be 10ms. Therefore you can send a read or write command every 10ms or 100 packets per second. Depending on the command you can send 8 or 16 byte per packet.

8 Package Diagrams





9 Ordering Information

Order Number	Number of GPIO	Package Type
902270	10	18 Pin (300Mil) SOIC
902370	10	18 Pin (0.300") DIP
902670	16	24 Pin (300Mil) SOIC
902770	16	24 Pin (0.300") DIP

10 Firmware Release Notes

Version 20 - Nov 1 2008 - Initial Release
Version 21 - Dec 2 2008 – Added support for Delcom Indicator products.
Version 22 - Jan 9 2009 – Fixed auto clear, auto confirm and I2C 16bit read command.
- Added more error bits to the error command. Fixed event counter.
Version 23 - Jan 16 2009 – Added open drain options for I2C and SPI. See write cmd 110.
Version 24 - Jan 27 2009 – Fixed error in 64Bit read command 17.
Version 25 - Mar 26 2009 – Added H-Bridge command. See write cmd 71.

11 Trouble Shooting

If Windows does not see the USB device in the Windows Device Manager or it is listed as an 'Unknown device' then you have a hardware problem. Most common errors are; Reserved D+/D- (green/white) wires, incorrect pull up resistor on the D- pin (1.3K resistor to the VREG pin), missing +5volts or ground, and VPP pin not tied to ground. Make sure your circuit matches the circuit in the Typical Schematic section above.

12 Notes

12.1 Power Notes

When the device boots up the total current consumed by the device should be at a minimum to comply with the USB standard (less than 25mA).

Cable length and cable size should be selected in order to maintain an operating voltage at the USB I/O chip of at least 4 volts. Failure to maintain 4 volts at the USB chip will cause it to reset.

This device can be used in a self-powered mode or with an external power supply if more than 450mA is required by the user's circuit. When using external power supplies, connect the USB I/O chip VCC to the USB supplied power and run the user added circuitry off the external power supply. Do not connect the USB VCC and external power supplies together, only connect the grounds.

12.2 Interfacing

When interfacing the USB I/O chip to your circuit, one must be careful not to over load the current on the individually pins or the total maximum current for the chip. Also one most not exceed the voltage maximums on the pins. If the voltage or current exceeds the limits of the chip you will have to add buffering to your design. For example most relays require more than 25mA to actuate the relay, and the USB I/O device can only sink 25mA. Therefore a current amplifier is required, such as a transistor or opto-coupler. When working with excessive currents, voltages or with high EMI circuits it is recommended that you use relays and/or opto-couplers to isolate the circuits.

13 Examples

13.1 C++ Example Direct Example

The following code snippet is not complete, shown here for example purposes only. For the complete code listing see the link in the references section. The following C function scans for the USB HID device and optionally tests for the family type number and serial number. If the device is found it copies the device handle to the global variable hDevice, saves the device name, and leaves the file open.

Note that the device can be opened in a shared or non-shared mode. Most OS required HID device to be opened in a shared mode.

```
#define USB_VID 0x0FC5    // USB Vendor ID (Always 0x0FC5 for Delcom products)
#define USB_PID 0xB080    // USB Product ID (Always 0xB080 for Delcom HID device)
#define USB_TID 0x0001    // USB Type ID (0=all, 1=USBHIDIO, 2=USBHIDVI,…)
#define USB_SID 0x0000    // USB Serial ID (zero=scan for all)
HANDLE  hDevice;          // Handle to the device
char   DeviceName[512];  // Devicename string

// ------------------------------------------------------ //
// ScanForHidDevice(VID,PID,TID,SID) - Scan thru all the HID device lookking
// for a match on the VID, PID and optional TID (Type ID) and SID(SerialNum)
// Sets the hDevice ghandle varible if found and opens the device
// Return zero if found, else non-zero error code.
// 0 = Success
// 1 = No matching HID devices
// ------------------------------------------------------ //
unsigned int ScanForHIDDevice(unsigned int VID, unsigned int PID, unsigned int
TID, unsigned int SID )
{
   //Use a series of API calls to find a HID with a matching Vendor,Product,
Type and Serial ID.
   DelcomDeviceInfoStruct        DelcomInfo;
   PSP_DEVICE_INTERFACE_DETAIL_DATA     detailData;
   GUID                                        HidGuid;
   HANDLE                                      hDevInfo;
   ULONG                                       Required;
   HIDD_ATTRIBUTES                             Attributes;
   SP_DEVICE_INTERFACE_DATA                    devInfoData;
   bool                                        LastDevice = FALSE;
   int                                         MemberIndex = 0;
   bool                                        MyDeviceDetected = FALSE;
   LONG                                        Result;
   ULONG                                       Length;

   // Variable init
   Length = 0;
   detailData = NULL;
   hDevice=NULL;
   MemberIndex = 0;
   LastDevice = FALSE;

   HidD_GetHidGuid(&HidGuid);           // Get the GUID for all system HIDs.

   hDevInfo=SetupDiGetClassDevs(&HidGuid, NULL, NULL, DIGCF_PRESENT |
DIGCF_INTERFACEDEVICE);
   devInfoData.cbSize = sizeof(devInfoData);
   do      {
   MyDeviceDetected=FALSE;
   Result=SetupDiEnumDeviceInterfaces(hDevInfo, 0, &HidGuid, MemberIndex,
&devInfoData);
   if (Result != 0)
```



```
   {      //A device has been detected, so get more info

   Result = SetupDiGetDeviceInterfaceDetail(hDevInfo, &devInfoData, NULL, 0,
&Length, NULL);  // zero length call to get size
   detailData = (PSP_DEVICE_INTERFACE_DETAIL_DATA)malloc(Length);
   detailData -> cbSize = sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA);

   Result = SetupDiGetDeviceInterfaceDetail(hDevInfo, &devInfoData, detailData,
Length, &Required, NULL);

   // To open device in a non-share mode, change the 3 parameter to NULL
   hDevice=CreateFile(detailData->DevicePath, GENERIC_READ|GENERIC_WRITE,
FILE_SHARE_READ|FILE_SHARE_WRITE,(LPSECURITY_ATTRIBUTES)NULL, OPEN_EXISTING, 0,
NULL);

   Attributes.Size = sizeof(Attributes);
   Result = HidD_GetAttributes(hDevice, &Attributes);
   MyDeviceDetected = FALSE;
   if( (Attributes.VendorID == VID) && (Attributes.ProductID == PID))
   {
   MyDeviceDetected = TRUE;
strncpy(DeviceName, detailData->DevicePath, 512);  // save the devicename
/* optional check for Family id and serial number, See Appendix A
   if(TID || SID) {       // Now check for TID and SID if non-zero
   if(GetDeviceInfo(&DelcomInfo)) MyDeviceDetected = FALSE;
   else {
          if(TID && (DelcomInfo.Family != TID)) MyDeviceDetected = FALSE;
          if(SID && (DelcomInfo.Serial != SID)) MyDeviceDetected = FALSE;
          }
     }    // end of TID or SID
*/
   }
Else {//The PID and/or VID doesn't match. Close the device try the next one
   CloseHandle(hDevice);
   hDevice = 0;
   }
free(detailData);
}  //if (Result != 0)
else      {       // End of List - No HID devices detected!
   LastDevice=TRUE;       // returned 0, so there are no more devices to check.
}
//If we haven't found the device yet,  then try the next one.
MemberIndex++;
} // loop till either end of deivce list or we find our device
while ((LastDevice == FALSE) && (MyDeviceDetected == FALSE));

SetupDiDestroyDeviceInfoList(hDevInfo);         //Free the memory
if (MyDeviceDetected == FALSE) {
   // Device not found
   hDevice = 0;
   return(1);
   }
else      {
   // Device Found
   HidD_SetNumInputBuffers(hDevice,1);
   return(0); // Success
 }
}
```


Write example.

```
// GEN2 Example to set port0 & port1 to 0xFF.
typedef union HIDPacketStruct {
   unsigned char Data[256];
   struct {
          unsigned char MajorCmd;
          unsigned char MinorCmd;
          unsigned char DataLSB;
          unsigned char DataMSB;
          unsigned char DataHID[4];
          unsigned char DataExt[8];
          } Tx;
   struct {
          unsigned char Cmd;
          } Rx;
   } HIDPacketStruct, *pHIDPacketStruct;

HIDPacketstruct MyPacket;        // Declare the packet
MyPacket.Tx.MajorCmd = 101;      // Fill the packet
MyPacket.Tx.MinorCmd = 1;
MyPacket.Tx.DataLSB = 0xFF;
HidD_SetFeature(hDevice,MyPacket, 8) // lastly send the packet
```

Read example.

```
// GEN2 USB Write/Read Command/Data Packet
typedef union HIDPacketStruct {
   unsigned char Data[256];
   struct {
          unsigned char MajorCmd;
          unsigned char MinorCmd;
          unsigned char DataLSB;
          unsigned char DataMSB;
          unsigned char DataHID[4];
          unsigned char DataExt[8];
          } Tx;
   struct {
          unsigned char Cmd;
          } Rx;
   } HIDPacketStruct, *pHIDPacketStruct;

HIDPacketstruct MyPacket;        // Declare the packet
MyPacket.Rx.Cmd = 100;                  // Fill the packet – read ports cmd
HidD_GetFeature(hDevice,MyPacket, 8)    // lastly send the packet

unsigned char Port0 = MyPacket Data[0];
unsigned char Port2 = MyPacket Data[1];
```


14 References

14.1 Documentation

Delcom USB HID Documentation and Examples
http://www.delcomproducts.com/productdetails.asp?productnum=900000

Delcom DLL
http://www.delcomproducts.com/productdetails.asp?productnum=890510

Delcom USB HID Chips
http://www.delcomproducts.com/products_USBIO.asp

Delcom Distribution Disk
http://www.delcomproducts.com/delcomcd/Start.htm

Cypress CY7C637XX Data Sheet.
http://www.delcomproducts.com/downloads/cy7c637xx-B.pdf

14.2 Examples

Delcom USB HID C# Net Direct Example
http://www.delcom-eng.com/productdetails.asp?ProductNum=890630

Delcom USB HID C++ Direct Example
http://www.delcom-eng.com/productdetails.asp?ProductNum=890607

Optional TID and SID Test

If you plan on also testing for the family type and/or serial number add the following code to your project, and uncomment the 'optional check for Family id and serial number' code in the ScanForHIDDevice function.

```
// Family Type Values
enum FamilyType{
   ALL,                 // all Delcom USB device
   USBIO,               // all Delcom USB IO Chips & foot switch
   USBVI,               // all Delcom USB Visual Indicators
   USBND                // all Delcom USB Numeric Displays
};

// DataStruct used by the GetDeviceInfo functions
typedef struct DelcomDeviceInfoStruct_ {
          unsigned short int Family;
          unsigned short int Security;
          unsigned char Version;
          unsigned char Day;
          unsigned char Month;
          unsigned char Year;
          unsigned int Serial;
          unsigned int Spare;
          } DelcomDeviceInfoStruct, *pDelcomDeviceInfoStruct;

// ------------------------------------------------------------------ //
// Reads device info
// Returns zero on success, else non-zero
// Return data in a 16byte data buffer. Buffer must be predeclared
// ------------------------------------------------------------------ //
int GetDeviceInfo(pDelcomDeviceInfoStruct pInfo)
{
   myPacket.Rx.Cmd = 104;

   if(!HidD_GetFeature(hDevice,&myPacket, 16)) {

          return(1);     // command failed
          }

   // now get the data if the variable has been passed
   if(!pInfo) return(1);
   memcpy(pInfo,&myPacket,16);
   return 0;
}
```


Appendix A. Revision History

Rev	Date	Author	Description
0	12/15/2008	DL	Initial Release
1	01/08/2009	DL	Corrected minimum schematic. Added pulse, auto clear, auto confirm and buzzer commands. Removed unsupported SPI commands. Fixed misc typos. Added bit details to read error command.
2	01/16/2009	DL	Added option command #110.
3	04/01/2009	DL	Added H-Bridge command #71. Added shared access note.
4	04/07/2009	DL	Correct Buzzer Example in 5.14.

Appendix B. Notices

DELCOM PRODUCTS INC. takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on DELCOM PRODUCTS INC procedures with respect to rights in DELCOM PRODUCTS INC. specifications can be found at the DELCOM PRODUCTS INC. website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers' or users of this specification, can be obtained from the DELCOM PRODUCTS INC. Executive Director.

DELCOM PRODUCTS INC. invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the DELCOM PRODUCTS INC. Executive Director.

This document and the information contained herein is provided on an "AS IS" basis and DELCOM PRODUCTS INC. DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

LIFE SUPPORT POLICY - Delcom Products are not authorized for use in life support devices and/or systems without the express written approval of Delcom.