USB IO HID G3 Migration

USB HID High Speed Peripheral Controllers

Generation 3

PRELIMINARY

Table of Contents

1 Summary

This document describes the differences between the Delcom USB generation 2 and generation 3 devices. Generation 3, here in referred to as G3, is backwards compatible with G2. This document should be used along with the G2 datasheet USBIOHID.pdf for users migrating from the G2 to the G3 devices. Users new to G3 should use the G3 datasheet USBIOHIDG3.pdf. The G3 devices use the same Vendor and Product ID, the same USB HID drivers and communicates using the same Delcom command set. G3 adds new features such as HID keyboard, mouse and joystick support, configurable I/O, configurable power up state, ADC, stored commands, timer events and many more.

There are some differences between the G3 and G2 devices of which this document describes. The main difference is the hardware, the new devices requires slightly different hardware. See the hardware section for more information. Most of the G2 commands are supported in G3 with a few exceptions. A few commands have been removed or have not yet been implemented, see the unsupported command section below for more information. There are also new commands that have been added to support the new features of G3, see software and command sections below for more information.

1.1 Generation History

2001 – Delcom releases G1 based on the Cypress CY7C63723 and CY7C63743 chips. The G1 chip sets required the Delcom proprietary driver on Windows and Mac OS.

2008 – Delcom released G2 based on the Cypress CY7C63723 and CY7C63743 chips. The G2 chip sets used the preinstalled USB HID driver if winch is preinstalled on most OS.

2015 – Delcom release G3 based on Microchip PIC18F14k50 and PIC18F2550 chips. The G3 chip sets are software compatible with the G2 chip sets. But the circuit is different due to the new chip.

2   Hardware

2.1   Typical Schematic

Below is are typical schematics of the G3 USB devices and required external components. The main differences between G2 and G3 are the new external 12MHz crystal oscillator, extra filter caps and D+ line 150K pull up resistor. Schematic also shows port I/O type and internal pullup options. Note that not all pins have input/output and pullup capabilities.

Figure 1 – 902551 - PIC18F14K50 Minimum Schematic

Figure 2 – 902552 - PIC18F2550 Minimum Schematic
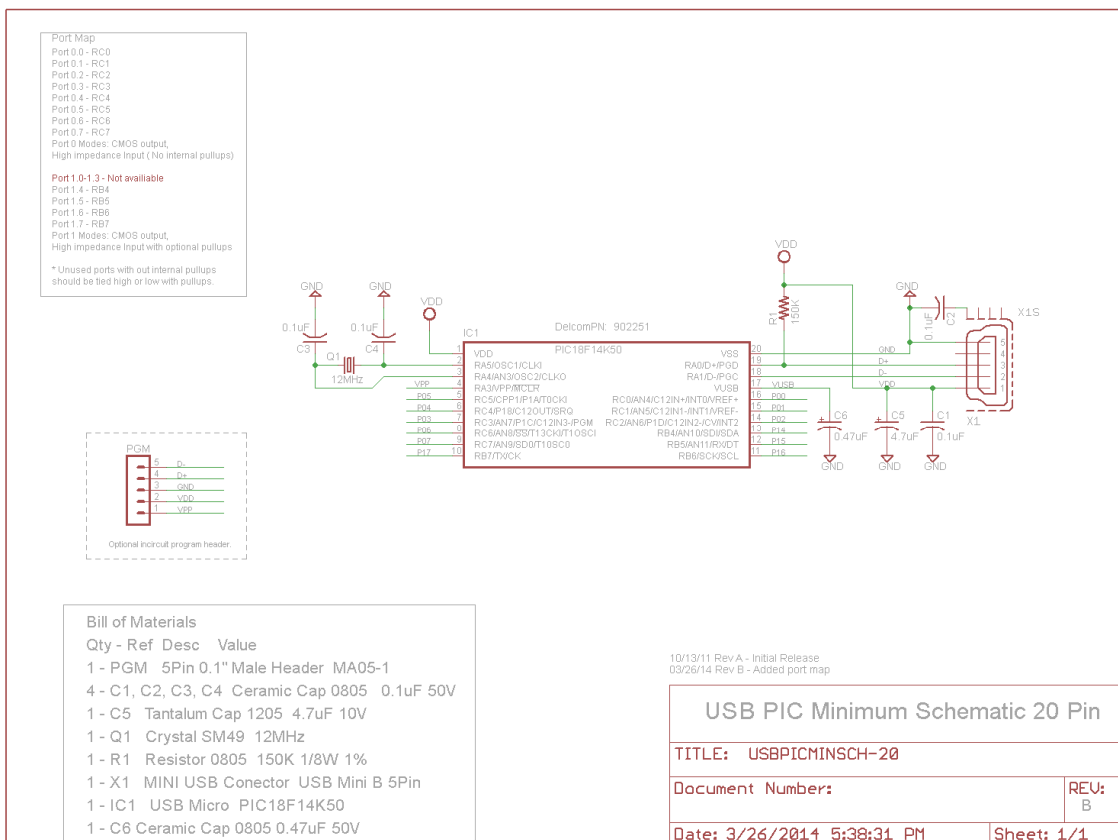
Delcom Products Inc.   USBIO G3 Migration

2.2   General Purpose IO

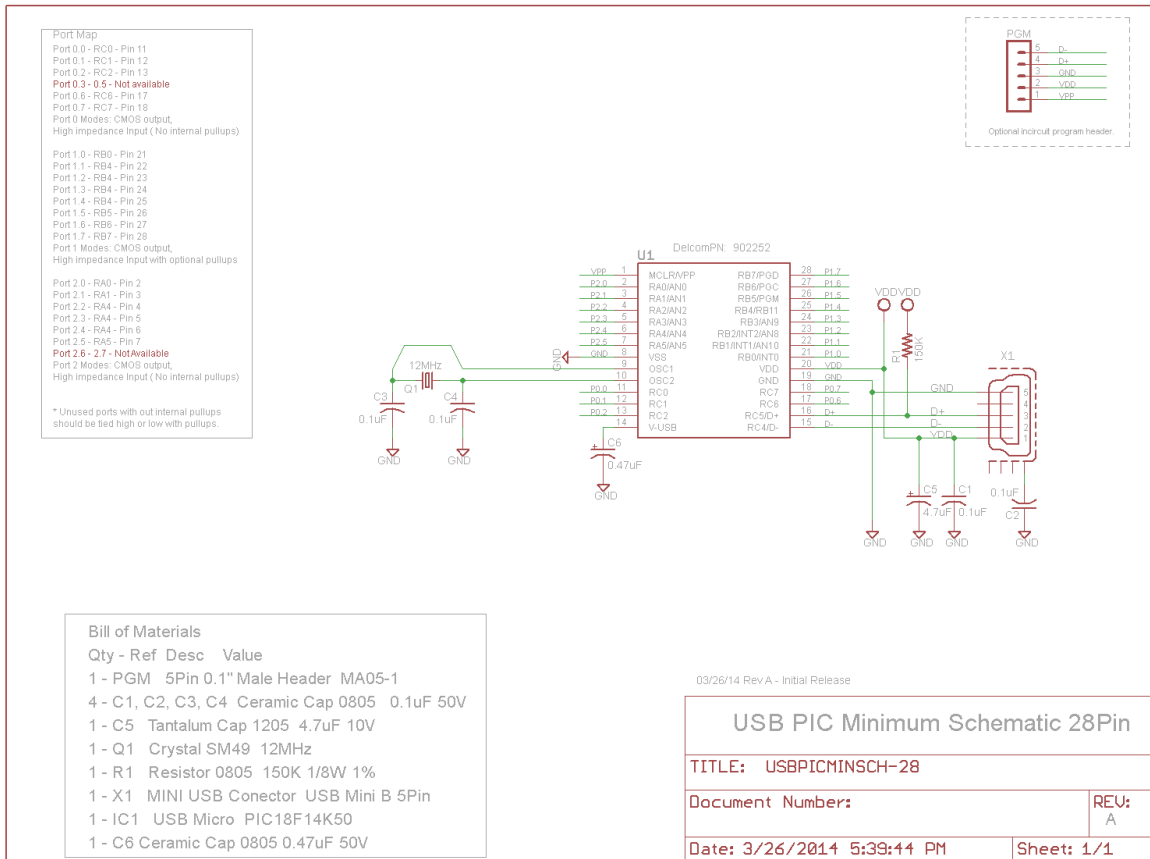The port structure of the G3 is different from the G2. G3 ports must be configured as either inputs or output. In order to be compatible with G2 all G3 ports power up in a Hi-Z input state by default. To change the port configuration see the new port setup commands (130) in the software section. The old G2 port setup commands are still available but are not fully supported. All the pins on the G2 devices 16K pullup. In G3 pull ups are only available on ports 1 and 2. Port 0 does not have internal pull ups, therefore the user must add external pullups if these pins are to be used as inputs.

Note new read command #104 to read port 2, port 2 value returned in first byte.

Figure 3 – Pin Mapping

Name	I/O	902251 20-Pin PIC18F14K50	902252 28-Pin PIC18F2250	Description (Alternate function)
P0.0	I/O	16, AN4	11	Port 0 bit 0
P0.1	I/O	15, AN5	12	Port 0 bit 1
P0.2	I/O	14, AN6	13	Port 0 bit 2
P0.3	I/O	7, AN7	-	Port 0 bit 3
P0.4	I/O	6	-	Port 0 bit 4
P0.5	I/O	5	-	Port 0 bit 5
P0.6	I/O	8, AN8, SS	17	Port 0 bit 6
P0.7	I/O	9, AN9,SD0	18, SDO	Port 0 bit 7
P1.0	I/O	-	21, PU, SDI	Port 1 bit 0
P1.1	I/O	-	22, PU, AN10, SCK	Port 1 bit 1
P1.2	I/O	-	23, PU, AN8	Port 1 bit 2
P1.3	I/O	-	24, PU, AN9	Port 1 bit 3
P1.4	I/O	13, PU, AN10,SDI	25, PU	Port 1 bit 4
P1.5	I/O	12, PU, AN11	26, PU	Port 1 bit 5
P1.6	I/O	11, PU,SCK	27, PU	Port 1 bit 6
P1.7	I/O	10, PU	28, PU	Port 1 bit 7
P2.0	I/O	-	2, AN0	Port 2 bit 0
P2.1	I/O	-	3, AN1	Port 2 bit 1
P2.2	I/O	-	4, AN2	Port 2 bit 2
P2.3	I/O	-	5, AN3	Port 2 bit 3
P2.4	I/O	-	6, AN4	Port 2 bit 4
P2.5	I/O	-	7, AN5, SS	Port 2 bit 5
P2.6	I/O	-	-	Port 2 bit 6
P2.7	I/O	-	-	Port 2 bit 7

#=Pin Number, PU=Internal Pullup, AN=Analog Pin

3   Software

This sections describes the G3 software differences, new features and commands. For older G2 commands see the USBIOHID.pdf datasheet. The G3 chips now includes a nonvolatile memory where all chip parameters can be saved. Changing any chip parameters and them saving them to the nonvolatile memory allows the chip to boot up with these settings, see section 3.4. For example you can now change a pin to output mode and set the initial logic to low or high using the port configuration commands and save them using the save configuration commands and on next power up the chip will boot up with these settings.

If a version number is indicated then that function was not supported till the firmware version number indicated. User should check if there device has the required firmware version or greater before using the command.

3.1   16 Byte Command

The G3 devices are optimized to use a 16 byte command using major write command 102. The 8 byte major write command 101 is still supported for backwards compatibility, but it is recommended to use the 16 byte, major command 102. The 8 byte write command, major 101 is not efficient and has a known problem on some platforms (see errata section below). This also applies to read commands, 8byte reads are supported but 16 byte reads should be used. To change an 8byte write command to a 16byte write command simply change the major command to 102 and the write length to 16. To change an 8 byte read command to 16bytes, simply just change the read length to 16.

```
// G3 – 16 Byte write and read command example
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

// G3 Write
MyPacket.Tx.MajorCmd = 102;            // 16 byte write command
MyPacket.Tx.MinorCmd = 1;              // Write P0 command
MyPacket.Tx.DataLSB = 0xFF;            // Write 0xFF to P0
HidD_SetFeature(hDevice,MyPacket, 16)  // send the packet

// G3 Read
MyPacket.Rx.Cmd = 100;                 // Read ports command
HidD_GetFeature(hDevice,MyPacket, 16)  // send the packet
BYTE c = MyPacket.Data[0];             // Get the Port0 value
```

3.2   Port Setup

The following commands setup the port pins initial power up value, port direction (input/output), enable/disable pull ups and enable/disable interrupts. The DataLSB parameter selects the port number to configure; 0=port0, 1=port1, 2=port2 and 3=port3. The DataHid0 sets the initial power up value. The DataHid1 sets the port direction. The DataHid2 enables the port pin pull ups. Note not all ports have internal pull ups. The DataHid3 enables port pin interrupts. The interrupts are used for the event counter feature (see write command 38).

Sending this command configures the port immediately. To make these changes permanent so that they are used on sequential power up you must call the save configuration command (see write command 186).

Note the older G2 setup port commands are still support but have limited functionality. The recommended method is to use the new G3 commands.

Syntax: WR 130 – Setup ports
DataMSB = 130
DataLSB = Port number. Range 0-3
DataHid0 = Initial port value on power up. 0=Low, 1=High. Default is High.
DataHid1 = Tristate value. Configures the port direction. 0=Output, 1=Input. Default is Input.
DataHid2 = Enable pullups. 0=Disabled, 1=Enabled. Default is enabled.
DataHid3 = Enable interrupt. 0=Disabled, 1=Enabled. Default is disabled.

Read Command 130 – Read port setup status
D00=P0.Value, D01=P0.Tris, D02=P0.Pullups, d03=P0.Interrupts,
D04=P1.Value, D05=P1.Tris, D06=P1.Pullups, D07=P1.Interrupts,
D08=P2.Value, D09=P2.Tris, D10=P2.Pullups, D11=P2.Interrupts,
D12=P3.Value, D13=P3.Tris, D14=P3.Pullups, D15=P3.Interrupts

```
// G3Port Setup example
// Set port 0 to be all inputs, with pull ups on.
MyPacket.Tx.MajorCmd = 102;            // 16 byte write command
MyPacket.Tx.MinorCmd = 130;            // Setup ports command
MyPacket.Tx.DataLSB  = 0x00;           // Select port 0
MyPacket.Tx.DataHid0 = 0xFF;           // Initail port value all high
MyPacket.Tx.DataHid1 = 0xFF;           // Port direction input
MyPacket.Tx.DataHid2 = 0xFF;           // Pullups on
MyPacket.Tx.DataHid3 = 0x00;           // Interrupts off
HidD_SetFeature(hDevice,MyPacket, 16)  // send the packet

// Set port 1 to be all Outputs, Initail value of 0x0F & pull ups off.
MyPacket.Tx.MajorCmd = 102;            // 16 byte write command
MyPacket.Tx.MinorCmd = 130;            // Setup ports command
MyPacket.Tx.DataLSB  = 0x10;           // Select port 1
MyPacket.Tx.DataHid0 = 0x0F;           // Initail port value all high
MyPacket.Tx.DataHid1 = 0x00;           // Port direction input
MyPacket.Tx.DataHid2 = 0x00;           // Pullups on
MyPacket.Tx.DataHid3 = 0x00;           // Interrupts off
HidD_SetFeature(hDevice,MyPacket, 16)  // send the packet

// G3 Read ports
MyPacket.Rx.Cmd = 100;                 // Fill the packet
HidD_GetFeature(hDevice,MyPacket, 16)  // send the packet

// G3 Read ports configuration
MyPacket.Rx.Cmd = 130;                 // Read Port Parameters
HidD_GetFeature(hDevice,MyPacket, 16)  // send the packet
```

3.3   Event Counter

With the G2 devices the event counter was available on all I/O pins. With G3 devices the event counter is only supported on P0.0 (INT0), P0.1 (INT1), and P0.2 (INT2) pins. The G2 write command 38 is still supported to enable/disable this feature. But you will need to use the new G3 write command 130 to set the active level. The default active level is high to low. See the 130 write command in the USBHIDG3 datasheet for more information. Also the reading of the event capture feature has changed. Before only one event capture value was available, now each pin P0.0, P0.1, P0.2 (INT0, INT1, INT2) has its own event counter register. To read the event counter use the same read command as in G2 (read command 8). The first 4 bytes will return P0.0 (INT0) counter value, the next 4 bytes will return P0.1 (INT1) counter value, and the next 4 bytes will return P0.2 (INT2) counter value.

Syntax: WR 38 – Setup Event Counters
MajorCmd: 102 – 16 byte command
MinorCmd: 38 – Setup Event Counters
DataLSB = Enable Port 0 bitwise
DataLSB = Enable Port 0 bitwise
DataMSB = Disable Port 0 bitwise

```
// G3 Event Counter Setup example
// Enable event counter on P0.0, P0.1 and P0.2
MyPacket.Tx.MajorCmd = 102;          // 16 byte write command
MyPacket.Tx.MinorCmd = 38;           // Setup ports command
MyPacket.Tx.DataLSB  = 0x07;         // Enable P0.0, p0.1, p0.2
MyPacket.Tx.DataMSB  = 0x00;         // Disable none high
HidD_SetFeature(hDevice,MyPacket, 16)   // send the packet


// G3 Read ports
MyPacket.Rx.Cmd = 8;                 // Read Event Counter command
HidD_GetFeature(hDevice,MyPacket, 16)   // send the packet
// P0.0 Counter = MyPacket[0-3]
// P0.1 Counter = MyPacket[4-7]
// P0.2 Counter = MyPacket[8-11]

// Event Counter Disable
MyPacket.Tx.MajorCmd = 102;          // 16 byte write command
MyPacket.Tx.MinorCmd = 38;           // Setup ports command
MyPacket.Tx.DataLSB  = 0x00;         // Enable none
MyPacket.Tx.DataMSB  = 0x07;         // Disable ALL
HidD_SetFeature(hDevice,MyPacket, 16)   // send the packet
```

3.4   Save Configuration Command

This command saves all the current device configuration to memory. There by allowing the device to power up in a user configurable state. This same command can also be used to restore a device to know factory settings. Note command will store all current settings so care should be used when using this command. This includes port initial values, port direction, port pull ups, port interrupts, port PWM, Port clock feature, events stored functions, timers and many more. For example if you have configured Port0 Pin0 to be a output pin with a initial power up value of low. With a PWM value of 60% and 20% duty value and P0.7 is current sounding a buzzer with a clock function and you call this command. Then these settings will be loaded every time the device powers up! Also note that this function writes to a non-volatile memory and write are limited to 100,000 write. Therefore this command should not be called on repeatedly. This command takes one parameter (DataHid3). When DataHid3 is set to zero all current configuration data is saved. If DataHid3 is other than zero then factory default configuration values are loaded (see table below). The nonzero value represents the Delcom family type number. Each family type configures (port settings and features) of the device to behavior is a certain way. Below is a partial list of Delcom family types. Note the family type number only defaults the chips setting, the user can overwrite these default settings at any time.

Table 1 – Family Type Table

#	Delcom Family Type
1	IO Controller, All ports inputs, Default for all standalone chips.
2	LED controller with step up regulators
3	Numeric Displays
4	LED controller without step up regulators
5	Buzzers
6	Microphones
7	Input devices with HID events
9	RF input devices with HID events
200+	User defined family types

Syntax: WR 186 – Save current configuration data
MajorCmd: 102 – 16 byte command
MinorCmd: 186 – Enhanced clock command
DataLSB = 0xAA
DataMSB = 0x55
DataHid0 = 0x5
DataHid1 = 0x1
DataHid2 = 0x7
DataHid3 = Mode, 0=Save current configuration, else factory reset to family type #

```
// G3 Save/restore configuration command
// Save current configuration
MyPacket.Tx.MajorCmd = 102;              // 16 byte write command
MyPacket.Tx.MinorCmd = 186;              // save config command
MyPacket.Tx.DataLSB  = 0xAA;             // 0xAA
MyPacket.Tx.DataMSB  = 0x55;             // 0x55
MyPacket.Tx.DataHid0 = 0x5;              // 5
MyPacket.Tx.DataHid1 = 0x1;              // 1
MyPacket.Tx.DataHid2 = 0x7;              // 7
MyPacket.Tx.DataHid3 = 0x0;              // 0 - Save current config data
HidD_SetFeature(hDevice,MyPacket, 16)    // send the packet

// Restore configuration data to family type 2 (LED controller)
MyPacket.Tx.MajorCmd = 102;              // 16 byte write command
MyPacket.Tx.MinorCmd = 186;              // save config command
MyPacket.Tx.DataLSB  = 0xAA;             // 0xAA
MyPacket.Tx.DataMSB  = 0x55;             // 0x55
MyPacket.Tx.DataHid0 = 0x5;              // 5
MyPacket.Tx.DataHid1 = 0x1;              // 1
MyPacket.Tx.DataHid2 = 0x7;              // 7
MyPacket.Tx.DataHid3 = 0x2;              // 2 - Save current config data
HidD_SetFeature(hDevice,MyPacket, 16)    // send the packet
```

3.5   Enhanced Clock Command

This command is similar to the G2 clock command (#70) but allows the clock function to be placed on any pin and has a finer control on the frequency. This command is a 16 byte command, therefore you must use the major cmd of 102. Note output pin must be in output mode for the pin to work.

$$FreqValue = 2/(10.666us*FreqHz)$$

Do to integer math the actual frequency would be:
$$ActualFreq = 2/(10.666us*FreqValue)$$

For example is you wanted 4KHz the program value would be 47 (round up from 46.8). And the actual value would be 3989Hz

Syntax: WR 170 – Enhanced clock command
MajorCmd: 102 – 16 byte command
MinorCmd: 170 – Enhanced clock command
DataLSB: EnableP0 – Set the bitwise pin the clock is to be enabled on port0.
DataMSB: EnableP1 – Set the bitwise pin the clock is to be enabled on port1.
HidData0:   FreqValueLSB – The LSB value of the frequency.
HidData1:   FreqValueMSB – The LSB value of the frequency.
HidData2-3:N/A – Not used set as zero.
DataExt0:   Repeat – 0=Continuous, else number of times to repeat.
DataExt1:   TimeOn – On Duty cycle in 50ms units. Used with repeat function.
DataExt2:   TimeOff – Off Duty cycle in 50ms units. Used with repeat function.
DataExt3-7: N/A – Not used set as zero.

```
// Configure P1.7 has an output (if not already done)
MyPacket.Tx.MajorCmd = 102;              // 16 byte write command
MyPacket.Tx.MinorCmd = 130;              // setup ports command
MyPacket.Tx.DataLSB  = 0x0;              // Select port 0
```

```
MyPacket.Tx.DataHid0 = 0xFF;              // Init Value (all high)
MyPacket.Tx.DataHid1 = 0x7F;              // 0:Output, 1:Input, Only P0.7=OUT!
MyPacket.Tx.DataHid2 = 0xFF;              // All Pullups On
MyPacket.Tx.DataHid3 = 0x00;              // All interrupts off
HidD_SetFeature(hDevice,MyPacket, 16)     // send the packet

// Output clock on P0.7 at 4000Hz repeat 2 times at a duty of 200ms/100ms
MyPacket.Tx.MajorCmd = 102;               // 16 byte write command
MyPacket.Tx.MinorCmd = 170;               // Enhanced clock command
MyPacket.Tx.DataLSB  = 0x80;              // Select port P0.7
MyPacket.Tx.DataMSB  = 0x00;              // not using port 1
MyPacket.Tx.DataHid0 = 47;                // FreqValueLSB
MyPacket.Tx.DataHid1 = 0;                 // FreqValueMSB
MyPacket.Tx.DataExt0 = 2;                 // Repeat twice, note 0=continuous
MyPacket.Tx.DataExt1 = 4;                 // Duty on 4*50ms = 200ms
MyPacket.Tx.DataExt2 = 2;                 // Duty on 2*50ms = 100ms
HidD_SetFeature(hDevice,MyPacket, 16)     // send the packet
```

3.6   Enhanced Capture/Compare/PWM (ECCP) Command

This command produces a single, half bridge or full bridge PWM waveform. The frequency range is from 500KHz (at 96 resolution) to 2930Hz at 1020 resolutions). The PWM output is on pins P1A,P1B,P1C,P1D. Single PWM can be on placed on any of the P1# pins. Half bridge uses P1A and P1B and full bridge used P1A,P1B,P1C and P1D pins. See the Enhanced Capture/Capare/PWM (ECCP) section in the micro chip data sheet. Active level of pins is programmable, but pin output mode and pin idle state must be preprogrammed.

Syntax: WR 174 – Enhanced ECCP command

MajorCmd: 102 – 16 Byte Write
MinorCmd: 174 – ECCP command
DataLSB: 10 – Setup the Enhanced PWM
DataHid0:   Sets CCP1CON, except the DC1B0 & DC1B1bites & enabled Timer2 or 0:Off
DataHid1:   Sets ECCP1AS – Autoshut down options – Default is zero
DataHid2:   Sets PWM1CON – Restart option and deadband value - Default is zero
DataHid3:   Sets PSTRCON – PWM output enable - Default is 0x01

MajorCmd: 102 – 16 Byte Write
MinorCmd: 174 – ECCP command
DataLSB: 11 – Sets the Timer2 period prescaler and postscaler
DataHid0:   Prescaler Fosc(12Mhz) 0:/1, 1:/4, 2:/16 default is zero
DataHid1: Postscaler divides value. Range:0-16. Default = 0

MajorCmd: 102 – 16 Byte Write
MinorCmd: 174 – ECCP command
DataLSB: 12 – Load the Period (10bits)
DataHid0:   Period LSB load only the 8 msb of the 10bit value.

MajorCmd: 102 – 16 Byte Write
MinorCmd: 174 – ECCP command
DataLSB: 13 – Load the PWM (10bits)
DataHid0:   PWM LSB
DataHid1: PWM MSB

```
@ECHO OFF
REM Delcom Enhanced PWM DOS batch file Example
REM USBCMDAP.exe must in the directory or search path. Download at ->
REM http://www.delcomproducts.com/productdetails.asp?PartNumber=890601
ECHO ------------------------------------------------------------
ECHO ----- Delcom Enhanced PWM Function Example CMD#174   -----
ECHO ------------------------------------------------------------

REM PWM VALUES - Range: 500KHz(Resolution=96 & Prescaler=1) to
2930Hz(Resolution=1020 & Prescaler=16)
SET "MASTERCLK=12000000"  %= PWM Master Clock Fixed at 12MHz =%
SET "PRESCALER=16         %= PWM Prescaler - Divides 12Mhz clock by 16, 4 or 1 =%
SET "FREQ=7800"           %= PWM Frequncy in Hz =%
SET "PRECENT=95"          %= PWM Duty in percent Range: 0-99 =%



REM PWM MODE SETUP
SET "CCP1CON=15"          %= PWM MODE: Single, Active Low. 0x0F =%
SET "ECCP1AS=0"           %= PWM Auto shutdown option, Not using set to zero =%
SET "PWM1CON=0"           %= PWM Deadband: Not using set to zero =%
SET "PSTRCON=1"           %= PWM port steering, P1A enabled. 0x01 =%



SET "Verbose="            %= v=verbose mode, v:verbose, leave blank for none =%

REM Calculate the values. %= Note batch files can only do integer math, so
expect slight math errors. =%
SET /a "PERIOD=(%MASTERCLK%/%FREQ%/%PRESCALER%)"        %= PWM Period =%
SET /a "RESOLUTION=%PERIOD%*4"
SET /a "DUTY=(%PRECENT%*%PERIOD%*4)/100"         %= PWN duty in 10bits =%
SET /a "DUTYMSB=%DUTY%/256"                      %= calc the DutyLSB =%
SET /a "DUTYLSB=%DUTY%-(256*%DUTYMSB%)"          %= calc the DutyMSB =%

ECHO Setting P0.5 to OUTPUT mode for PWM output. %= Only needed if pin is not
already in out mode. =%
REM Syntax: TID, SN, 102, 130, Port#(0-4), NA, Value, Tris(0:OUT,1:IN),
Pullups(0:off,1:on), Interrupts(0:off,1:on)
call USBCMDAP %Verbose% 0 0 101 130 0 0 255 223 255 0

ECHO PWM setup.....
ECHO FREQ=%FREQ%Hz PERIOD=%PERIOD% PRESCALER=%PRESCALER% PSValue=%PSVALUE%
Resolution=%RESOLUTION%
ECHO DUTY=%PRECENT%  Value=%DUTY% DUTYMSB=%DUTYMSB%  DUTYLSB=%DUTYLSB%
if %PERIOD% GEQ 256 ECHO ERROR: Frequency value too low, try increasing
prescaler!
if %PERIOD% LEQ  0 ECHO ERROR: Frequency value too high, try decreasing
prescaler!
if %PRESCALER% == 16 SET "PSVALUE=3"
if %PRESCALER% == 4 SET  "PSVALUE=1"
if %PRESCALER% == 1 SET  "PSVALUE=0"

call USBCMDAP %Verbose% 0 0 102 174 11 0 %PSVALUE% 0 0 0
   %= Set the PWM prescaler =%
call USBCMDAP %Verbose% 0 0 102 174 12 0 %PERIOD% 0 0 0
   %= Set the PWM Period
call USBCMDAP %Verbose% 0 0 102 174 13 0 %DUTYLSB% %DUTYMSB% 0 0
   %= Set the PWM Duty
call USBCMDAP %Verbose% 0 0 102 174 10 0 %CCP1CON% %ECCP1AS% %PWM1CON% %PSTRCON%
   %= Set the PWM Mode =%

REM PAUSE  %= To stop the PWM send zero to the PWM mode. =%
REM call USBCMDAP %Verbose% 0 0 102 174 10 0 0 0 0 0   %= Stop the PWM =%
```

3.7   Pulse Command   **Version 52**

This command operates the same as the pulse in the G1 but the timing has changed. The new timing is:

Estimated Period = 4.3us + DELAY# * (0.95us + (Prescaler * 0.34us))

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. There are 5 port pin state change parameters and 4 delay parameters. To change the port data parameters change the port value by executing a XOR with the current port value and the PortDataX value. So to toggle a pin set the PortDataX 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. This functions disables interrupts, therefore it should be keep as short as possible. Timed functions maybe be delay/slewed by this function.

Syntax: WR 76 – Setup User Timer command
MajorCmd: 102 – 16 byte command
MinorCmd: 176 – Pulse command
DataLSB: PortSelect bit 7 0=port0, 1=port2. Bits6:0 = prescaler.
DataMSB: PortData0 – Data to be XOR with the port.
HidData0-3: Not used – Set to zero.
DataExt[0]: Delay0
DataExt[1]: PortData1
DataExt[2]: Delay1
DataExt[3]: PortData2
DataExt[4]: Delay2
DataExt[5]: PortData3
DataExt[6]: Delay3
DataExt[7]: PortData4

```
// 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 _____-----_____-----____
```

3.8   Toggle Command   **Version 52**

This command operates the same as the Toggle function in the G2 but the timing has changed. The new timing is:
Estimated Period = 1.6us + (DELAY* 0.360us)

This command will toggle the GPIO pins. Any GPIO pin on ports port0 and port1 can be toggled. The function will toggle the current pin state X times. The time between toggles is variable with the delay parameter. To toggle a GPIO on port0 set the corresponding pin high in the LSBData parameter. For port1 use the MSBData parameter. The function will toggle all the pins marked in LSBData and MSBData for X number of times. Set DataHid0 equal to the number of times to toggle. So to toggle the pin once set it to zero. Set the Datahid1 to the delay value. Note port0 toggles 1us before port1. This functions disables interrupts, therefore it should be keep as short as possible. Timed functions maybe be delay/slewed by this function.

Syntax: WR 77 – Setup User Timer command
MajorCmd: 102 – 16 byte command
MinorCmd: 77 – Pulse command
DataLSB: Port0Data
DataMSB: Port1Data
DataHid[0]: Count = Number of times to toggle the port
DataHid[1]: Delay = Delay between toggles.

```
// Toggle Pin Command example
MajorCmd =101;     // 8 byte write command
MinorCmd =77;      // Toggle Pin Command
LSBData = 0x01;    // Port0 Pin0
MSBData = 0x02;    // Port1 Pin1
DataHid0 = 9;      // 10 toggles (9+1)
DataHid1 = 10;     // 12.7us Delay = 6us+(0.67us*10)

//This will cause pins P0.0 and P1.1 to toggle 10 times (5 pulse)
//Assuming P0.0 & P1.1 were high to start.
//Port0.0 --__--__--__--__--__--
//Port1.1 --__--__--__--__--__--
```

3.9   Stored Functions Command   **Version 52**

This command stores up to 8 user defined commands into the chip that can then be then be called by triggers. A triggers can be power on, pin level change or timer triggers. For triggers see the event and timer commands. For example you could store a clock command that sound a buzzer and a port change command that lights an LED. And then setup another input to call these two stored functions when the input changes from a low to a high. Or you could setup a timer trigger to sound the buzzer and toggle the LED on a 10 second timer (once or continuously). There is enough space to store 8 eight byte command. Therefore 16 byte commands will require 2 stored functions and you can only store 4 16 byte commands. 16 byte command must be stored in consecutive slots. You will have to issue the 154 command twice to store a 16 byte command. See the save configuration command to make this function permanent.

Syntax: WR 153 - Store User Functions command
MajorCmd: 102 – 16 byte command

MinorCmd: 154 – Store User Functions command
DataLSB: Function# - The function number to store the command at. Range 0-7
DataMSB: N/A – Not used set as zero
HidData0-3: N/A – Not used set as zero
DataExt0-7:The command function to store

Syntax: RD 154 – Reads stored users commands 0-3
Syntax: RD 155 – Reads stored users commands 4-7
Data[0-7] – Stored command 0 or 4
Data[6-15] – Stored command 1 or 5
Data[16-23] – Stored command 2 or 6
Data[24-32] – Stored command 3 or 7

```
// Write 8 byte stored function at #0 to make P0.0 low
MyPacket.Tx.MajorCmd = 102;             // 16 byte write command
MyPacket.Tx.MinorCmd = 154;             // setup ports command
MyPacket.Tx.DataLSB  = 0x00;            // stored cmd #0
MyPacket.Tx.DataExt0 = 101;             // majorcmd 101 – 8 byte cmd
MyPacket.Tx.DataExt1 = 11;              // minorcmd 11 – bitset/reset Port 0
MyPacket.Tx.DataExt2 = 0x01;            // datalsb 0x01 P0.0 low
MyPacket.Tx.DataExt3 = 0x00;            // datamsb 0x00
HidD_SetFeature(hDevice,MyPacket, 16)   // send the packet

// Write 16 byte stored function at #6 and #7 to sound the buzzer
// First send the first 8 bytes to slot 6
MyPacket.Tx.MajorCmd = 102;             // 16 byte write command
MyPacket.Tx.MinorCmd = 154;             // Write Stored Cmd
MyPacket.Tx.DataLSB  = 0x00;            // stored cmd #6
MyPacket.Tx.DataExt0 = 102;             // majorcmd 102 – 16 byte cmd
MyPacket.Tx.DataExt1 = 170;             // minorcmd 170 – bitset/reset Port 0
MyPacket.Tx.DataExt2 = 0x80;            // datalsb select P0.7
MyPacket.Tx.DataExt3 = 0x00;            // datamsb  P1 N/A
MyPacket.Tx.DataExt4 = 47;              // hiddata0 FreqLSB
MyPacket.Tx.DataExt5 = 0;               // hiddata1 FreqMSB
MyPacket.Tx.DataExt6 = 0x00;            // hiddata2 N/A
MyPacket.Tx.DataExt7 = 0x00;            // hiddata3 N/A
HidD_SetFeature(hDevice,MyPacket, 16)   // send the packet
// Then send the last 8 bytes to slot 7
MyPacket.Tx.MajorCmd = 102;             // 16 byte write command
MyPacket.Tx.MinorCmd = 154;             // Write Stored Cmd
MyPacket.Tx.DataLSB  = 7;               // stored cmd #7
MyPacket.Tx.DataExt0 = 2;               // dataext0 – repeat
MyPacket.Tx.DataExt1 = 4;               // dataext1 – time on
MyPacket.Tx.DataExt2 = 2;               // dataext2 – time off
HidD_SetFeature(hDevice,MyPacket, 16)   // send the packet
```

3.10 Timer Event Command   **Version 52**

This command stores up to 4 user defined timer triggers into the chip. The timer can be programmed to be continuous or run once. When the timer times out a selectable stored function is called (see stored functions). Timer resolution units are 1.024ms and the period value is a 32bit integer. The timer range is 1.024ms to 4398046 seconds or 1221 hours. Setting the most significant bit high of the timer number will enable the auto reload (continuous) timer function. Setting it low will disable the auto reload and the timer will only run once. See the save configuration command to make this function permanent.

Syntax: WR 156 – Setup User Timer command
MajorCmd: 102 – 16 byte command

MinorCmd: 156 – Setup User Timer command
DataLSB: Timer# - The timer# 0-3. The MSBit enbled auto reload (continuous).
DataMSB: StoredCmd#- The bitwise stored command(s) to call when timer expirers
HidData0-3: Period – The timer period in 1.024ms. 32bit value Range 0-2^32

Syntax: RD 156 – Reads timer parameters 0-3
Data[0-7] – timer parameters 0
Data[6-15] – timer parameters 1
Data[16-23] – timer parameters 2
Data[24-32] – timer parameters 3

```
// Write a 1 second timer to call stored command #2
MyPacket.Tx.MajorCmd = 102;            // 16 byte write command
MyPacket.Tx.MinorCmd = 156;            // Timer Command Setup
MyPacket.Tx.DataLSB  = 0x00;           // Timer #0
MyPacket.Tx.DataMSB  = 0;              // trigger stored cmd #2
MyPacket.Tx.DataHid0 = 0xE8;           // Period 1000 (0x3E8)
MyPacket.Tx.DataHid1 = 0x03;           //
MyPacket.Tx.DataHid2 = 0x00;           //
MyPacket.Tx.DataHid3 = 0x00;           //
HidD_SetFeature(hDevice,MyPacket, 16)  // send the packet
```

3.11 Events Command

This command setup the 8 events on port 0. The port 0 events can be connected to buttons or other driving logic. The event trigger level is from high to low (except of the stored functions which can be trigger on both edges). The events can trigger USB HID keyboard, mouse, joystick axis/button or stored function events. This command configures the events on port 0. The command has four different modes. The mode is determined by the value of DataLSB. Some of the commands are bitwise command, with this commands the LSb is P0.0 and MSb is P0.7. See the save configuration command to make this configuration permanent. To enable event set the event option (see option commands). Events are enabled by default if the family/device type is a 7.

Mode 0 will set the event action and threshold. The event actions are hold/repeat and momentary and are set bitwise. In hold/repeat the device will send a down event when the pin goes low and the up event when the pin goes high. With momentary the down and up events are sent when the pin goes low only, (the pin up state is ignored). The port 0 pins have a hardware and software debounce functions. The software debounce function is controlled by the threshold value. The value 0 turns the software debounce function off. Other values values enable this feature. The range is 0 to 255 where the higher the number the more the pin is software debounced. The default value is 50. This value is global to all the 8 events pins on port 0.
Syntax: WR 132 – Setup Events
MajorCmd: 102 – 16 byte command
MinorCmd: 132 – Setup Events
DataLSB: Mode0 – Set the event threshold and action.
HidData0:   Threshold value
HidData1:   Action value

```
// Set the event mode 0 command – set all actions hold/repeat & debounce @ 50
MyPacket.Tx.MajorCmd = 102;            // 16 byte write command
MyPacket.Tx.MinorCmd = 132;            // Event Setup
MyPacket.Tx.DataLSB  = 0;              // Mode0 – Threshold and Action
MyPacket.Tx.DataHid0 = 50;             // Threshold – debounce 50
MyPacket.Tx.DataHid1 = 0x00;           // Action all pins hold/repeat
```

```
HidD_SetFeature(hDevice,MyPacket, 16)    // send the packet
```

Mode 1-8 will configure the event type, code and modifier. The event pin that is configured is selected by the DataLSB value where 1=P0.0 and 8=P0.7. The modifier values is only used on the keyboard type to pass the modifier keys (alt, ctrl, shift).

EVENT TYPES:
    REPORT_TYPE_UNKNOWN = 0   - This will turn the event off
    REPORT_TYPE_KEYBOARD = 220   - Sends a keyboard code
            CODE: Keyboard code – See http://www.usb.org/developers/hidpage/Hut1_12v2.pdf
            MODIFIER: Keyboard modifier keys (lAlt, lCtrl, lShift, lGui, rAlt, rCtrl,rShift,rGui)
    REPORT_TYPE_JOYSTICK_BUTTON = 221   - Sends a joystick button
            CODE: Joystick button number 1-16
    REPORT_TYPE_JOYSTICK_AXIS = 222   - Sends a joystick axis
            CODE: Joystick axis 1-4 (X,Y,Z,A)
    REPORT_TYPE_MOUSE = 223   - Sends a mouse button
            CODE: Mouse Button (Left:1; Center:2; Right:4)
    REPORT_USER_CMDS_DWN = 250   - Sends a stored cmd on pin low
            CODE: Stored Function – Bitwise e.g 0x81 will call stored functions 0 and 7
    REPORT_USER_CMDS_UP = 251   - Sends a stored cmd on pin high
            CODE: Stored Function

Syntax: WR 132 – Setup Events
MajorCmd: 102 – 16 byte command
MinorCmd: 132 – Setup Events
DataLSB: Mode1-8 – Event pin #
HidData0: Event Type
HidData1: Event Code
HidData2: Event Modifier

```
// Set the event pin P0.3 to send the space keycode
MyPacket.Tx.MajorCmd = 102;          // 16 byte write command
MyPacket.Tx.MinorCmd = 132;          // Event Setup
MyPacket.Tx.DataLSB  = 4;            // Mode4 -P0.3
MyPacket.Tx.DataHid0 = 220;          // Type – Keyboard 220
MyPacket.Tx.DataHid1 = 44;           // Code – Space Bar code
MyPacket.Tx.DataHid1 = 0;            // Modifier – node
HidD_SetFeature(hDevice,MyPacket, 16)    // send the packet
```

Mode 9 will set the event group. This will combine events to trigger off of a lower event. For example say you wanted to send all there events P.0, P0.1 and P0.2 when P0.0 went low. To do this set the group value to 0x03. The lowest event is what will trigger the group. In the above example a low on P0.0 will trigger the P0.0,P0.1 and P0.2 events. A low on P0.1 will only trigger P0.1 and P0.2 events. And a low on P0.2 will only trigger P0.2. For events types of keyboard, mouse and joystick the maximum you can group is 5.

Syntax: WR 132 – Setup Events
MajorCmd: 102 – 16 byte command
MinorCmd: 132 – Setup Events
DataLSB: Mode9 – Set the event group value.
HidData0: Group value

```
// Set the event mode 9 command – Group events P0.0, P0.1 and P0.2
MyPacket.Tx.MajorCmd = 102;          // 16 byte write command
MyPacket.Tx.MinorCmd = 132;          // Event Setup
MyPacket.Tx.DataLSB  = 9;            // Mode0 – Group
MyPacket.Tx.DataHid0 = 0x07;         // Group value
HidD_SetFeature(hDevice,MyPacket, 16)    // send the packet
```

Mode 10 will set the event LED controls. There are four LED on pins P1.4-P1.7 and corisponde to event pins P0.0-P0.3. This command will enable the LED to toggle on either the event pin going low, high or both. This command works in a bitwise fashion where LSb is P0.0 and MSb is P0.7. Note the power up state(on/off) of the LED is configured with the port setup command (#130).

Syntax: WR 132 – Setup Events
MajorCmd: 102 – 16 byte command
MinorCmd: 132 – Setup Events
DataLSB: Mode10 – Set the event group value.
HidData0:   LED Down Control
HidData1:   LED Up Control

```
// Set the event mode 10 command - LED control setup. Set LED on P1.4 on toggle
// on both a down and up event
MyPacket.Tx.MajorCmd = 102;             // 16 byte write command
MyPacket.Tx.MinorCmd = 132;             // Event Setup
MyPacket.Tx.DataLSB  = 10;                    // Mode0 - Group
MyPacket.Tx.DataHid0 = 0x01;            // LED Down Value
MyPacket.Tx.DataHid1 = 0x01;            // LED Up Value
HidD_SetFeature(hDevice,MyPacket, 16)   // send the packet
```

All the event parameters can be read with the 132 read command.
Syntax: RD 132 – Read Event parameters
Return Data
Data[0]:   Event Threshold Value – Software debounce – Default =50
Data[1]:   Event Actions – Bitwise 0:hold/repeat 1:momentary
Data[2+(2*Event#)]:   Event# type
Data[3+(2*Event#)]:   Event# code
Data[4+(2*Event#)]:   Event# modifier
Data[27]:   Event group value
Data[28]:   Event LED down value
Data[29]:   Event LED up value
Data[30]:   Not Used
Data[31]:   Not Used

```
MyPacket.Rx.Cmd = 132;                  // Read Event Parameters
HidD_GetFeature(hDevice,MyPacket, 16)   // send the packet
```

3.12 Options and Tasks Command

The G3 device has two 32 bit registers used for enabling and disabling options and tasks. The G2 device had only a one byte options register. The command uses a bitwise set/reset mechanism to enable/disable the options and tasks. Unintelligent sets and resets can cause the system to be unstable. To use this command set DataLSB value to 'O' for Options or 'T' for Tasks. Set the DataMSB to 'S' to set or 'R' to reset. The value of DataHid[0-3] is then set or reset.

Syntax: WR 131 Set/Reset the task or Option register
MajorCmd: 102 – 16 byte command
MinorCmd: 131 Set/Reset the task or Option register

DataLSB: 'O'=Options, 'T'=Tasks
DataMSB: 'S'=Set, 'R'=Reset timer expirers
HidData0-3: The value to bitwise set or reset

Syntax: RD 131 Read the task or Option register
ReadCmd: 131 – read task & options
Data[0-3] Task Value - 32 bit
Data[4-7] Option Value - 32 bit

```
// Set the event task bit on to enable the button events
MyPacket.Tx.MajorCmd = 102;              // 16 byte write command
MyPacket.Tx.MinorCmd = 131;              // Set/Reset
MyPacket.Tx.DataLSB  = 'T';              // Timer #0
MyPacket.Tx.DataMSB  = 'S';              // trigger stored cmd #2
MyPacket.Tx.DataHid0 = 0x01;             // The event task bit is bit0
MyPacket.Tx.DataHid1 = 0x00;             //
MyPacket.Tx.DataHid2 = 0x00;             //
MyPacket.Tx.DataHid3 = 0x00;             //
HidD_SetFeature(hDevice,MyPacket, 16)    // send the packet
```

Task Values:
Byte 0.0: Events – Enables the port 0 events
Byte 0.1: Not used
Byte 0.2: Mic – Enables Red Green MIC LEDs
Byte 0.3: Test – Used for debugging
Byte 0.4: Runs fast clock on bootup.
Byte 0.5: Auto Clear - Clears the light on button P0.0 press
Byte 0.6: Auto Comfirm – Sounds the buzzer on button P0.0 press
Byte 0.7: Enables ADC – Enables the analog to digital converter
Byte 4.0-7: Enables stored functions to run on boot up

Option Values:
Byte 0.0: Not used – Was I2C open drain mode in G2
Byte 0.1: Not used – Was SPI open drain mode in G2
Byte 0.2: Mic – Enables Red Green MIC LEDs
Byte 2.0: P1SwapNibble – Enables Port 1 top and bottom nibble swap
Byte 2.1: RS232 – Enables RS232 port
Byte 2.2: RS232Debug – Enables Debug mode over RS232 port
Byte 2.3: I2CMasterMode – Enables master I2C mode on boot up
Byte 2.4: I2CSlaveMode – Enables slave I2C on mode boot up
Byte 2.5: I2CSlewMode – Enables I2C Slew mode
Byte 2.6: Not used – Was event logic direction in G2
Byte 2.7: Port1Invert – Enables port 1 invert logic
Byte 3.0: KB_MS_JS – Enables KB,MS & JS
Byte 3.1: USB_SN - Enables USB SN to be set on USB Enumeration
Byte 4.0: SkipBootScan – Disables boot scan. Used to stop Linux from prescaning reports
Byte 4.4: AutoReloadT0 = Enables auto reload on timer 0
Byte 4.5: AutoReloadT1 = Enables auto reload on timer 1
Byte 4.6: AutoReloadT2 = Enables auto reload on timer 2
Byte 4.7: AutoReloadT3 = Enables auto reload on timer 3

3.13 ADC Command

The G3 devices support an analog to digital convertor with a resolution 1024 bits. See the Pin Mapping section to see winch pins support the ADC feature. To use the ADC you first need to put the pin in input mode, disable pullups and enable ADC mode. Use the Setup Ports command 130 to do this. Once the port pin is setup you can repeatedly call the ADC start conversion with command WR 39 and read the converted analog value with the RD 100.

Syntax: WR 130 Set up ADC channel
MajorCmd: 102 – 16 byte command
MinorCmd: 130 Set/Reset the task or Option register
HidData0-1: ADC Channel number (2 bytes), e.g. AN4 = 0x04

Syntax: WR 39 Start the ADC conversion
MajorCmd: 102 – 16 byte command
MinorCmd: 39 Start the ADC conversion
DataLSB:: ADC Channel number shifted 2 bytes to the right (1 byte), e.g. AN4 >>2 = 0x01

Syntax: RD 100 Read the ADC value
ReadCmd: 100 – Read ports and ADC values
Data[8-9] ADC value, resolution 2^10 (1024bits)

```
// Read the ADC on channel AN4 (P0.0 on the PIC18F14k50)
// First set P0.0 to input mode and pullup off.
MyPacket.Tx.MajorCmd = 102;            // 16 byte write command
MyPacket.Tx.MinorCmd = 130;            // Setup ports command
MyPacket.Tx.DataLSB  = 0x00;           // Select port 0
MyPacket.Tx.DataHid0 = 0xFF;           // Initial port value all high
MyPacket.Tx.DataHid1 = 0xFF;           // Port direction input
MyPacket.Tx.DataHid2 = 0x00;           // Pullups off
MyPacket.Tx.DataHid3 = 0x00;           // Interrupts off
HidD_SetFeature(hDevice,MyPacket, 16)  // send the packet

// Second enable the ADC Channel AN4(0x0003)
MyPacket.Tx.MajorCmd = 102;            // 16 byte write command
MyPacket.Tx.MinorCmd = 130;            // Setup ports command
MyPacket.Tx.DataLSB  = 30;             // Enable ADC Ch command
MyPacket.Tx.DataHid0 = 0x04;           // ADC Channel LSB
MyPacket.Tx.DataHid1 = 0x00;           // ADC Channel MSB
HidD_SetFeature(hDevice,MyPacket, 16)  // send the packet

// Lasty Start the ADC conversion and read the value.
MyPacket.Tx.MajorCmd = 102;            // 16 byte write command
MyPacket.Tx.MinorCmd = 39;             // Start ADC Conversion command
MyPacket.Tx.DataLSB  = (0x04>>2);      // ADC Ch command (right shift by 2)
HidD_SetFeature(hDevice,MyPacket, 16)  // send the packet

// Read the ADC value. ValueLSB=Data[8] and ValueMSB- Data[9]
MyPacket.Rx.Cmd = 8;                   // Read Event Counter command
HidD_GetFeature(hDevice,MyPacket, 16)  // send the packet
// ValueLSB = MyPacket[8]
// ValueMSB = MyPacket[9]
```

3.14 SPI Command   Version 55

The SPI functions allow the usb chip(master) to interface to an SPI compliant device(slave). SPI WR Commands 90, 93,94 and 95 will write data to the SPI port. And SPI read command 90 will return the data read from the SPI device.

If your using the default SPI SS pin (see SPI Pin Table below) then you will not have to setup the SPI SS pins, this is automatically done. In this case all you need to do is send a SPI write command (90,93,94 or 95), followed by the optional SPI read command 90.

If your using a nonstandard SS port, then you will need to manually setup the ports pins for SCK, SDI, SDO and SS before calling the SPI WR commands. See example below.

SPI Pin Table

PIN \ MICRO	PIC18F14K50	PIC18F2550
SCK	11 – P1.6	22 – P1.1
SDO	9 – P0.7	18 – P0.7
SDI	13 – P1.4	21 – P1.0
SS	8 – P0.6	7 – P2.5

The default SPI mode is Mode 0,0 and clock rate is 750KHz. The can be changed with WR command 92, see https://www.delcomproducts.com/downloads/pic18f2550.pdf section 19.0 for more details. Default SS is active low, but this can be changed by presetting the all SPI pins with the port setup commands and setting the SS pin to low (nonactive state). The WR command 93,94 and 95 command will toggle the pin from the current level.

Syntax: WR 90 SPI Write w/o SS
MajorCmd: 102   16 byte command
MinorCmd: 90   SPI Write 8-16 bytes
DataExt[0-15]   Data to write to SPI device

Syntax: WR 91 OLD SPI Setup commend not used/ ignored, left for backwards compatibility.
MajorCmd: 102   16 byte command
MinorCmd: 91   OLD SPI Setup Command

Syntax: WR 92 SPI setup Command
See Section 19.0 in https://www.delcomproducts.com/downloads/pic18f2550.pdf
MajorCmd: 102   16 byte command
MinorCmd: 92   SPI Setup
DataLSB: SSPCON1   Sets SSPCON1
DataMSB: SSPTAT1   Sets SSPSTAT

Syntax: WR 93 SPI Write with SS on Port 0
MajorCmd: 102   16 byte command
MinorCmd: 93   SPI Write 8-16 bytes
DataLSB: SS   Set bit high to toggle SS on Port0
DataExt[0-15]   Data to write to SPI device

Syntax: WR 94 SPI Write with SS on Port 1
MajorCmd: 102   16 byte command
MinorCmd: 94   SPI Write 8-16 bytes
DataLSB: SS   Set bit high to toggle SS on Port1
DataExt[0-15]   Data to write to SPI device

Syntax: WR 95 SPI Write with SS on Port 2
MajorCmd: 102   16 byte command
MinorCmd: 95   SPI Write 8-16 bytes
DataLSB: SS   Set bit high to toggle SS on Port2
DataExt[0-15]   Data to write to SPI device

Syntax: RD 90   SPI Read 8 to 32 bytes
ReadCmd: 90
Data[0-32]   Data read from the last SPI command.

SPI Write and Read Example using SS on P2.5 (default) on PIC18F2550

```
// SPI Write with SS on P2.5, 16 bytes
MyPacket.Tx.MajorCmd = 102;           // 16 byte write command
MyPacket.Tx.MinorCmd = 95;            // SPI Write with SS on port2
MyPacket.Tx.DataLSB  = 0x20;          // Toggle SS on P2.5
MyPacket.Tx.DataExt0 = 0x40;          // SPI Data Byte 0
MyPacket.Tx.DataExt1 = 0x20;          // SPI Data Byte 1
MyPacket.Tx.DataExt1 = 0x00;          // SPI Data Byte 2
HidD_SetFeature(hDevice,MyPacket, 16) // send the packet

// Read the ADC value.
MyPacket.Rx.Cmd = 90;                 // Read SPI data
HidD_GetFeature(hDevice,MyPacket, 16) // send the packet
// SPIDAT[0-15] = MyPacket[0-15]
```

SPI Write and Read Example using SS on P0.0 (non default) on PIC18F2550

```
// Port Configuration - Only needs to be done once
// Note when using non default SS you must setup all SPI pins(SCK,SDO,SDI,SS)
// PORT 0 CFG
MyPacket.Tx.MajorCmd = 102;           // 16 byte write command
MyPacket.Tx.MinorCmd = 130;           // Setup ports command
MyPacket.Tx.DataLSB  = 0x00;          // Select port 0
MyPacket.Tx.DataHid0 = 0xFF;          // Initial port value all high
MyPacket.Tx.DataHid1 = 0x7E;          // Port dir. All input but SD0&SS=OUT
MyPacket.Tx.DataHid2 = 0xFF;          // Pullups on
MyPacket.Tx.DataHid3 = 0x00;          // Interrupts off
HidD_SetFeature(hDevice,MyPacket, 16) // send the packet

// PORT 1 CFG
MyPacket.Tx.MajorCmd = 102;           // 16 byte write command
MyPacket.Tx.MinorCmd = 130;           // Setup ports command
MyPacket.Tx.DataLSB  = 0x00;          // Select port 0
MyPacket.Tx.DataHid0 = 0xFF;          // Initial port value all high
MyPacket.Tx.DataHid1 = 0xFC;          // Port dir. All input but SCK=OUT
MyPacket.Tx.DataHid2 = 0xFF;          // Pullups on
MyPacket.Tx.DataHid3 = 0x00;          // Interrupts off
HidD_SetFeature(hDevice,MyPacket, 16) // send the packet


// SPI Write with SS on P0.0, 16 bytes
MyPacket.Tx.MajorCmd = 102;           // 16 byte write command
MyPacket.Tx.MinorCmd = 93;            // SPI Write with SS on port2
MyPacket.Tx.DataLSB  = 0x01;          // Toggle SS on P0.0
MyPacket.Tx.DataExt0 = 0x40;          // SPI Data Byte 0
MyPacket.Tx.DataExt1 = 0x20;          // SPI Data Byte 1
MyPacket.Tx.DataExt1 = 0x00;          // SPI Data Byte 2
HidD_SetFeature(hDevice,MyPacket, 16) // send the packet

// Read the ADC value.
MyPacket.Rx.Cmd = 90;                 // Read SPI data
HidD_GetFeature(hDevice,MyPacket, 16) // send the packet
// SPIDAT[0-15] = MyPacket[0-15]
```

4   Unsupported Commands

The following commands are currently not supported at this time in G3. These may be added at a future releases.

4.1.1   Write Commands

3	Setup Read Strobe
8	Setup 64 bit read command
13	Write with high strobe
14	Write with low strobe
15	Write 8byte with high strobe
16	Write 8byte with low strobe
17	Write 64 bit command
30	P0 Enable pullups
31	P1 Enable pullups
32	Setup P0 sink current
33	Setup P1 sink current
36	Setup read buffer command
37	Write scratch pad
71	H-bridge control – See new 174 PWM Command.
77	Toggle pin command

4.1.2   Read Commands

1	Read P0 with high strobe
2	Read P0 with low strobe
5	Read buffer
7	Read scratch pad
17	Read 64 byte command
18	Write 2 bytes read 8 bytes command
50	Read RS232 buffer
50	Read RS232 buffer

Revision History

Rev	Date	Author	Description
0	03/04/2015	DL	Preliminary Release
A	03/30/2018	DL	Added Commands
B	02/25/2019	DL	Added SPI commands

Errata

Errata#G3.100
PROBLEM – Back to back 8 byte write commands sometimes fails.
ENVORIOMENT - Windows 7 OS with Intel USB 3.0 Extensible Host Controller hardware.
SOLUTION 1 – Always use 16byte write (DataMajor=102) commands for all write commands.
SOLUTION 2 – Update software. New releases of the EHCI has resolved this problem.