

### **Powering On:**

- Hold down the START button for at least [2] seconds. The green LED will light up immediately, blink very quickly for a moment and then stay on until this timer has expired. Once the LED goes out, you can release the button. Do not press any other buttons during this time as various buttons set features and modes on startup.
- When the system first powers on, a CRC calculation is done on the firmware which is what is happening while the power LED blinks quickly for the first moment. If the firmware has become corrupted, the unit will not stay powered on when you release the START button.

### **Powering Off:**

- Hold down the START button for [3] seconds to power off if the iControlPad is not currently connected over the BT link. Hold down the START button for [6] seconds while connected.
- The system can never turn off while the START button is being held down. Even though the power down sequence will be acknowledged and the system will drop into low power mode, the power is on until your finger is removed.

### **Different Modes:**

- These button combos must be held down during power up. They are sampled right at the end of the power up timer (as the LED goes dark). Press down on the following buttons or button combos and then press the START button to turn the device on. The START button can be released once the LED turns off but any other buttons selected cannot. When the mode is successfully changed, the LED will blink quickly three times in acknowledgement. After this, you can release the buttons. Any buttons or button combinations not listed here are reserved and should not be pressed during start up.
  - a) SPP mode: Y button [**shipping default**]
  - b) HID keyboard: A button
  - c) HID keyboard and mouse: B button
  - d) HID Joystick mode: X button
  - e) HID Mouse: A and B buttons
  - f) HID Keyboard (special packet mode): A and X buttons
  - g) HID Gamepad: X and Y buttons

### **Other startup configurations (advanced users!):**

- These are more button combos that must be held down during power up to activate other features. Press these buttons before holding down the START button and continue to hold them down until the green LED flashes quickly three times. The START button can be released once the green LED turns off and before blinking in acknowledgement of changes.
- Modes only have to be changed once and they are remembered until a different one is selected.

- These configurations are toggles. The default shipping configuration for all of these is OFF. Each time these configurations are activated, they toggle from OFF → ON or ON → OFF.
- a) Dedicated Keyboard Media Controller mode: B and X buttons
- b) Keyboard Media Controller when pressing SELECT mode: A and Y buttons
- c) Alternate Keyboard Configuration mode: B and Y buttons
- d) Auto-reconnect in HID mode: DPAD UP and DPAD RIGHT (also works while not connected and not just at startup but must be held for **[3]** seconds).
- e) Reset to Default: A, B, X and Y buttons
- This will restore ALL settings in the iControlPad to factory default settings.

#### **Automatic shutdown capability:**

- The iControlPad will power itself off if it does not make a connection or has lost a connection for **[2]** minutes.
- The iControlPad will power itself off if no buttons are pressed or the analog nubs are not moved for **[5]** minutes. If the external charger is active, the system will enter low power mode but will stay on to charge the phone until the power is manually turned off, the USB cable is removed from the iControlPad or the battery is drained. **[this can be disabled]**
- The iControlPad will power itself off if the battery becomes drained **[3.3V]**.

#### **Charging the iControlPad:**

- Plugging in the mini-B end of a USB cable providing 5V will begin charging the iControlPad and will stop charging automatically once the battery is full. The red LED on the front of the unit will indicate charging status. The light will be on while charging. The iControlPad does not need to be powered on to charge itself.

#### **Charging a mobile phone using the iControlPad:**

- The external charging circuit is not enabled until a USB cable with mini-A connector is plugged into the bottom of the iControlPad. Charging is not enabled until **[5]** seconds after the unit has successfully powered on. The connection is only checked once every **[5]** seconds so there can be a small delay before charging actually starts when a cable is plugged in.
- Internal charging is **[enabled]** by default and will automatically be disabled when the battery voltage reaches **[3.5V]**. The charger will not turn back on until the battery is charged back up to **[3.8V]**.
- The internal charger will also automatically disable itself if the output voltage happens to fall out of range of the USB spec (4.4V – 5.25V). This can happen if a device that tries to draw too much power is connected.

#### **Power LED (green):**

- The power LED blinks once every **[1]** second when the iControlPad is not connected over a BT link.
- The power LED blinks once every **[5]** seconds when the iControlPad is connected over a BT link.

- When the battery is low [**<3.5V**], the LED will double blink. For example, while not connected and the battery is low, the LED will blink twice in rapid succession every [**1**] second.
- The power LED is also used to acknowledge specific button presses that change modes and settings. This can vary depending on what is being changed but generally involves several rapid blinks in a short period of time.

**Reflashing the main firmware:**

- On power up, hold down the START and SELECT buttons for 1.5 seconds. The power LED will light up immediately and blink off when the hold timer has expired. Make an SPP connection to the PC and run the downloader software.

**Reflashing the BT module's firmware:**

- This process will not be actively supported as it works, but if the user does something wrong during flashing, the BT module is 'bricked' until physical pins on the PCB are accessed to reprogram the module via its SPI port.

**Explanation of Modes:**

a) SPP mode

- Connects to a computer or phone like a wireless serial port.
- This mode allows setting special control registers and is used to change the various programmable settings (designed for power users only).
- Communications are initiated by the host with custom responses that must be interpreted by a custom driver.

b) HID keyboard

- Game buttons are mapped to normal, lower case keyboard keys. The analog nubs are not available.

	<u>Normal layout</u>	<u>Alternate Layout</u>
DPAD UP	w	Up arrow
DPAD DOWN	s	Down arrow
DPAD LEFT	a	Left arrow
DPAD RIGHT	d	Right arrow
A	j	Space
B	k	Enter
X	m	Page Down
Y	i	Page Up
START	y	Tab
SELECT	r	F5
LEFT TRIGGER	q	Home
RIGHT TRIGGER	p	End

c) HID keyboard and mouse

- Combines the keyboard mode with the mouse mode. By default, only the nubs emulate the mouse and scroll wheel and mouse buttons are not available as game buttons are mapped to keyboard keys.

d) HID Joystick mode

- Standard mode for treating the iControlPad as a gamepad or joystick. This mode is the proper selection for standard gamepad usage.

e) HID Mouse

- This mode simulated a three button mouse with scroll wheel. The left nub is defined as the mouse and the right nub is defined as the scroll wheel by default.  
A            Button 1 (left click)  
X            Button 2 (right click)  
B            Button 3
- The mouse and scroll wheel speed can be configured in this mode ONLY and carries over for all mouse mode usage. Holding down the SELECT button plus DPAD directions allows this. Since the nub usage can be reversed (mouse/scroll), configuration is defined by left or right nub and not functionality.

DPAD UP	Increase sensitivity of right nub
DPAD DOWN	Decrease sensitivity of right nub
DPAD LEFT	Decrease sensitivity of left nub
DPAD RIGHT	Increase sensitivity of left nub

f) HID Keyboard (special packet mode) **[EXPERIMENTAL!!]**

- This mode connects as a standard keyboard and sends readable ASCII characters.
- To decode the ASCII characters in the gamepad data sections, do the following:  
nub X1,Y1,X2,Y2: subtract 64 from binary value of ASCII character  
button set 1,2: subtract 32 from binary value of ASCII character, read individual bits that remain as game buttons (lower 6 active bits per byte)
- Characters are only sent when there is a change in the button or nub states.
- Button data bits can be interpreted as follows:  
<SET 1 - 7:0> = <0><0><SELECT><LEFT TRIG><DOWN><LEFT><RIGHT><UP>  
<SET 2 - 7:0> = <0><0><RIGHT TRIG><B><X><A><Y><START>

**Firmware 1.0.0**

- Every button press or analog movement is sent as an eight byte packet. The packet starts with the signature 'MW' and is followed by six bytes of gamepad data: <nub X1> <nub Y1> <nub X2> <nub Y2> <button set 1> <button set 2>
- Sending this much data in keyboard mode can be disastrous! In testing, the best case was about 13Hz and sometimes can crash/cause errors in the host HID driver or the iCP itself.

**Firmware 1.0.1**

- The default setting will be to only send button data as the throughput can be higher. Nub data can be enabled by special utility using SPP (not described here).
- There are two packet types sent individually. Transmitted bytes are shown here:  
1. <m> <button set 1> <button set 2>  
2. <w> <nub X1> <nub Y1> <nub X2> <nub Y2>
- Button only mode is currently set for 20Hz throughput. Button + Nub is 13Hz.

g) HID Gamepad

- This is an experimental mode that defines the iControlPad as a gamepad instead of a joystick. The reports are identical. Only the Class Of Device (COD) is different. This COD is not always recognized and may require a custom driver.

Explanation of Keyboard Media Controller:

- These are extensions of the keyboard mode. 'Dedicated keyboard media controller mode' overrides the normal keyboard key outputs of the game buttons. 'Keyboard Media Controller when pressing SELECT' mode does not override the keyboard keys unless they are pressed with the SELECT button.

DPAD UP	Next Track
DPAD DOWN	Previous Track
DPAD LEFT	Rewind
DPAD RIGHT	Fast Forward
A	Mute
B	Play/Pause
X	Volume Down
Y	Volume Up
START	Stop

Using SPP Mode

Example: Create a thread that runs forever:

- Wake it up no more than 40 times per second to poll iCP for data (or use auto report mode). The nubs are only updated 40 times per second.
- The digital controls are updated any time you need them.
- The more you poll it, the more power the iCP will use because you won't let it go back to sleep as often.
- If a proper command gets sent with invalid parameters, the iCP does not reply.
- To be sure a bad command is flushed, wait 125ms before sending a new one. That is also the maximum allowed time between every byte sent in a command.

Here is a simple poll for digital controls:

```
- Send <0xA5> to poll for game button data
#define CMD_GET_DIGITALS 0xA5
- Wait for two bytes to be received: <byte A> and <byte B> in that order
- Byte A has these bits active ('1' = pressed, '0' = released):
#define GB_DPAD_LEFT 2 //A
#define GB_DPAD_RIGHT 1 //A
#define GB_DPAD_UP 0 //A
#define GB_DPAD_DOWN 3 //A
#define GB_SHLDR_LEFT 4 //A
- Byte B has these bits active ('1' = pressed, '0' = released):
#define GB_SHLDR_RIGHT 6 //B
#define GB_BTN_A 3 //B
#define GB_BTN_B 5 //B
#define GB_BTN_X 4 //B
```

```
#define GB_BTN_Y                2    //B
#define GB_BTN_START            1    //B
#define GB_BTN_SELECT           0    //B
```

- All other bits in these bytes will be '0'

Here is a simple poll for analog controls:

```
- Send <0x87> to poll for analog positions
#define CMD_GET_ANALOGS        0x87
- Wait for four bytes to be received: <Nub1 X> <Nub1 Y> <Nub2 X> <Nub2 Y>
- The bytes are signed. The values will range from -32 to +32.
- +32 is full down or full right
- -32 is full up or full left
- 0,0 is center
```

Controlling the charger is done in hardware/iCP firmware by default, but there are manual settings to force it off or let it turn back on. You can also monitor the battery voltage and make a little battery picture in your application or something like that.

```
- You can check the iCP's version number by sending <0x39>
#define CMD_GET_ID              0x39
- The response is a long string of characters ending with a carriage return and line feed.
```

```
- You can check the battery's voltage by sending <0xAA>
#define CMD_GET_BATT_VOLTS     0xAA
- the response will be three readable characters in this example format:
  405 <-- This means 4.05 volts.
```

```
- You can check the battery for a general level by sending <0x55>
#define CMD_GET_BATT_LEVEL     0x55
- The response will be a single byte with the following significance.
```

```
#define BATT_FULL                6
#define BATT_ALMOST_FULL        5
#define BATT_3_4_FULL           4
#define BATT_HALF_FULL          3
#define BATT_1_4_FULL           2
#define BATT_ALMOST_DEAD        1
#define BATT_DEAD                0
```

- The definition for these levels are estimated right now (based on voltage readings) and not expected to be particularly accurate.

```
- You can check the status of the charger to the host device by sending <0xDE>
#define CMD_GET_CHGR_STATUS    0xDE
- The response will be one byte signifying either on or off as follows:
#define SET_ON                  0x01
#define SET_OFF                 0x00
```

- You can force the charger to the host on or off by sending <0x2A>

```
#define CMD_FORCE_CHARGER          0x2A
```

- With either one of these parameter to turn on or off:

```
#define SET_ON                      0x01
```

```
#define SET_OFF                    0x00
```

- The response will be one byte to signify if the action could be completed

```
#define RESP_OKAY                  0x80
```

```
#define RESP_NOT_OKAY             0x81
```

- You can always force the charger off but you cannot force it on if the iCP detects a low battery or an external charger is already plugged in

  

- You can check the charger's voltage (either the USB input voltage when charging the iCP or the output voltage when charging a phone) by sending <0x6F>

```
#define CMD_GET_CHARGER_VOLTS      0x6F
```

- The response will be three readable characters in this example format:  
505 <-- This means 5.05 volts.

  

- The activity of the power LED can be changed manually by sending the command <6D>

```
#define CMD_FORCE_LED_CTRL        0x6D
```

- Then send the parameter to tell the iCP if you want control of the LED or not:

```
#define SET_ON                      0x01
```

```
#define SET_OFF                    0x00
```

- ON means you want to override the iCP's internal control.
- OFF means you want to give back control to the iCP.

  

- The LED will be forced off with either successful command.
- The response will always be positive:

```
#define RESP_OKAY                  0x80
```

- Once control is established, turn the LED ON with the command <0xFF>

```
#define CMD_SET_LED                0xFF
```

- Then send the parameter for on or off:

```
#define SET_ON                      0x01
```

```
#define SET_OFF                    0x00
```

- The response will signify if the action could be completed:

```
#define RESP_OKAY                  0x80
```

```
#define RESP_NOT_OKAY             0x81
```

- If you have not taken control of the LED, your request will be denied with a not okay reply.

  

- The activity of the power LED can be changed to alter its behavior when the iCP hardware has control of it. Currently, the iCP uses the LED to indicate power status, BT link status and internal battery low conditions.
- CMD\_FORCE\_LED\_CTRL must be OFF for this to work.
- Send the command <0xE4>

```
#define CMD_SET_LED_MODE           0xE4
```

- Send one of the parameters to change the mode:
- ```
#define LED_PULSE_DOUBLE          0
#define LED_PULSE_INVERSE        2
#define LED_LOW_BATT_IND         4
#define LED_PULSE_DQUICK         5
```
- Pulse double: LED is normally off but blinks once every 5 seconds. It will do a double blink if the battery is low.
  - Pulse inverse: LED is normally on but blinks off once every 5 seconds when battery is low
  - Low batt indicator: LED is always off unless battery is low. Then it stays on until battery is recharged (to a certain threshold).
  - Pulse double quick: LED is normally off but blinks once every 1 second. It will do a double blink if the battery is low.
  - The iCP's default usage of the LED is restored when power is cycled or if the BT link is dropped. The default is to use PULSE\_DOUBLE when BT is not connected and PULSE\_DQUICK when connected.

There are 480 bytes of non-volatile memory in the iCP that can be used to store anything. Another 32 bytes above this is reserved for iCP usage only and can only be read but not written.

- To read a byte of memory send the command <0x9A>
- ```
#define CMD_READ_EEPROM          0x9A
```
- Then send the two byte address of the single byte to read. The range is 0 - 511: <low byte><high byte>
  - The response will be the data as a single byte reply. If the address is out of range, the response will be a <0x00>.
  - To write a byte, send the command <0x7A>
- ```
#define CMD_WRITE_EEPROM        0x7A
```
- Then send the two byte address of the single byte to write. The range is currently 0 - 479. The address is sent in this format: <low byte> <high byte>
  - Next send a security byte to verify this operation is to happen:
- ```
#define SECURE_EE_BYTE          0x19
```
- Then send the single byte of data to write
  - The response will signify if the action could be completed:
- ```
#define RESP_OKAY               0x80
#define RESP_NOT_OKAY           0x81
```
- Attempting to write out of range will be denied.

The baud rate of communication can be changed to one of several values. The default on power up will always be 9600bps for the bootloader but will use programmed settings for the main application. The shipping firmware operates at 57600bps to reduce latency from sampling to transmission. It isn't recommended to change the speed unless it benefits the user in SPP mode somehow.

- To set the baud rate, send the command <0xC2>
- ```
#define CMD_CHANGE_BAUD_RATE    0xC2
```
- Then send the single byte parameter for the new rate:

```

#define BAUD_RATE_9600          4
#define BAUD_RATE_19200       3
#define BAUD_RATE_38400       2
#define BAUD_RATE_57600       1
- The iCP will immediate reply at the current baud rate if the action can be done:
#define RESP_OKAY              0x80
- If the parameter sent is not supported, the action will be denied and no response is
  given.
- The baud rate will not actually change until the power is cycled.

- The iCP can be powered off manually by sending the command <0x94> and then
  a three byte confirmation sequence:
#define CMD_POWER_OFF          0x94
#define PWR_OFF_CHK_BYTE1     0x27
#define PWR_OFF_CHK_BYTE2     0x6A
#define PWR_OFF_CHK_BYTE3     0xFE
- The iCP will reply with an acknowledgement byte. Shortly after that it will drop
  the BT link and power down.
#define RESP_OKAY              0x80

```

An autonomous mode for getting game button data can be activated. It will remain active until any other command is sent or until this command turns it off.

```

- To toggle the automatic reports, send the command <0xAD>
#define CMD_SPP_GP_REPORTS    0xAD
- Then send an ON or OFF state byte:
#define SET_ON                 0x01
#define SET_OFF                0x00
- Game button packets will only be sent whenever there is a change in the data and
  are scheduled with a minimum time of 25ms between them.
- The response is a six byte packet:
  <nub X1> <nub Y1> <nub X2> <nub Y2> <button data 1> <button data 2>
- The button data is identical to the format presented above in the manually polled
  instructions.

```

## **Joystick/Gamepad Mode output format**

<u>Button</u>	<u>iControlPad Input</u>
1	DPAD UP
2	DPAD RIGHT
3	DPAD LEFT
4	DPAD DOWN
5	LEFT TRIGGER
6	(not used, always off)
7	(not used, always off)
8	(not used, always off)
9	SELECT
10	START
11	Y
12	A
13	X
14	B
15	RIGHT TRIGGER
16	(not used, always off)

The analog nubs are mapped to the following axis:

LEFT NUB	X,Y
RIGHT NUB	Z,Rx