Interbotix Labs Robot Turret Manual

Introduction

The Interbotix Robot Turrets are a line of high performance Pan/Tilt units powered by the Arbotix Robocontroller. These units ship with a default Comm_EXT firmware preloaded which has a basic serial communication protocol & Instruction Set for controlling the Dynamixels, the DC motor controller, and various I/O. The design intent is that the Robot Turret is to be operated via USB FTDI Cable or Xbee Wireless tether to a PC, and are capable of sending sensory data back to said PC. A USB or Wifi Camera can also be used to transmit video data back to the PC for machine vision applications. A Roborealm software module has been designed to help facilitate this and communicates directly with the Comm_EXT firmware instruction set. There is a GUI within Roborealm for controlling the Robot Turret, as well as .NET based VB & C# demo applications available.

Advanced users may also create their own custom firmware for autonomous operation or other applications. Documentation on custom programming the Arbotix is available on Vanadium Labs' website. Please refer to our assembly guides for instructions on how to build particular model of Robot Turret. Links to all documentation can be found on the Product Page.

Serial Port Communication

The Robot Turret Kit come with an FTDI Cable which is used for both USB Communication with a PC as well as optional custom programming of the Arbotix. Please note that if you are using the FTDI Cable to program your ArbotiX, you must enable the "Set RTS On Close" option in your *Device Manager >> USB Serial Port Properties >> Port Settings Tab >> Advanced menu*. The ArbotiX also supports the use of Xbee radios for wireless communication; more information on setting these up can be found in the Arbotix Documentation.

Comm_EXT Firmware

The Comm_EXT firmware allows full access to all of the features of the Robot Turret using a simple serial protocol and instruction set. This firmware that comes preloaded on Arbotix's that ship with Robot Turret. You do not need to reprogram your Arbotix unless you intend to modify or design your own firmware.

The firmware has two modes of operation built in: Commander Standard mode and Commander Extended mode.

The Commander Standard protocol is built to be plug-n-play compatible with the Arbotix Commander v2.0 Gamepad Controller. It uses the Right Joystick on the Arbotix Commander Gamepad to control the Pan/Tilt and the buttons (R1, R2, R3, L4, L5, L6, RT, LT) are tied directly to digital outputs (d0-d7) on the Robot Turret's Arbotix. Please note that the standard protocol is 'output only', there is no method for reading the I/Os as input. It also has a lower servo resolution (0-255) than the Commander Extended protocol. There is an alternate firmware available for use in this mode, Comm_EXT Proportional which allows for proportional control of the pan/tilt- this is much better for 'aiming' as the joystick will not auto-center and does not directly control the pan/tilt's position. For a full explanation of the Commander Standard protocol, please refer to the Arbotix Commander GamePad Controller Manual.

The Commander Extended protocol is intended to be the preferred PC communication protocol and the default protocol for the Robot Turret. It allows for full resolution (0-1023) control of your pan & tilt servos, access/polling of digital & analog I/O, and DC motor speed & directional control.

The Packet Structure consists of 8 bytes sent to the Arbotix via serial connection. Please refer to the tables below for a breakdown of the Commander Extended protocol.

Commander Extended Packet Structure:

Byte	Name	Value
1	Header	0xFF (255)
2	Tilt H	0x0-0x3 (0-3)
3	Tilt L	0x00-0xFF (0-255)
4	Pan H	0x0-0x3 = (0-3)
5	Pan L	0x00-0xFF (0-255)
6	Button Values	D7-D0=00000000(leave default 0x00 for Commander Ext)
7	Ext Instruction	See Table Below, default is 0x08
8	Checksum	= (255 - (byte2+byte3+byte4+byte5+byte6+byte7) Mod 256)

The Pan and Tilt servos use 0-1023 for positional commands; this value cannot be contained in a single byte so it is spread over a Low and High Byte. The Ext Instruction Byte should default to 0x08, this will keep the Arbotix in "Extended Mode" and retain full resolution of the Pan/Tilt. The Button Values Byte (byte 6) should default to 0, as digital IO is handled by the Ext Instruction Byte while in Extended Mode.

Ext Instruction Byte:

OpCode	Instruction
0x00	Cancel Extended Instruction Mode
0x08	No Action
0x10	Read Analog0 as 8-bit value
0x11	Read Analog1 as 8-bit value
0x1K	Read AnalogK as 8-bit value
0x1B	Read Digital0 through Digital7 as a Byte
0x40	Motors Off
0x50-K	Left Motor Reverse, (K*10)/100% of Speed
0x50	Left Motor Off
0x50+K	Left Motor Forward, (K*10)/100% of Speed
0x70-K	Right Motor Reverse, (K*10)/100% of Speed
0x70	Right Motor Off
0x70+K	Right Motor Forward, (K*10)/100% of Speed
0x80	Set Digital0 as low and input
0x81	Set Digital0 as high and input (pullup enabled)
0x82	Set Digital0 as low and output
0x83	Set Digital0 as high and output
0x84	Set Digital1 as low and input (and so on)
0x90	Set Digital5 as low and input (and so on)

Return Packet Structure:

Anytime the Ext Instruction Byte is used to read an input (analog or digital), a return packet is sent from the Arbotix to the PC.

Byte	Name	Value
1	Header	0xFF (255)
2	Ext Byte	Ext Instruction that was processed (0x11, 0x1B, etc)
3	Ext Byte Value	Value of the Ext Instruction that was requested.
4	Checksum	= (255 - (byte2+byte3) Mod 256)