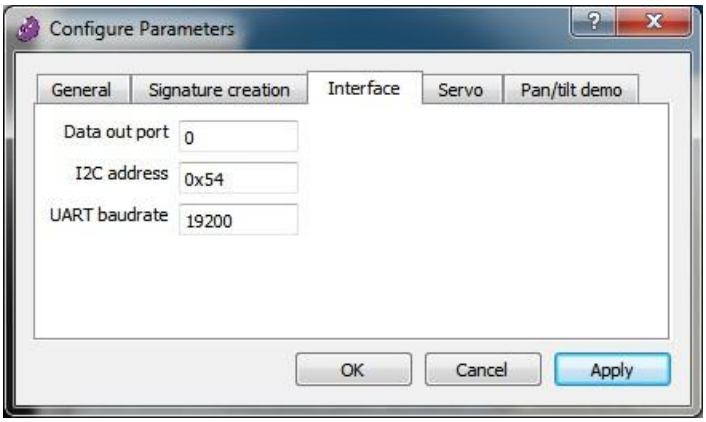
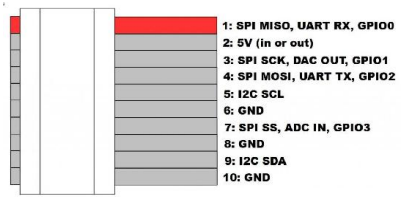
Pixy Serial Protocol

Pixy will output objects that it detects through one of several interfaces that you choose, and it will do this every 20 ms. It supports SPI, I2C, UART, and analog/digital I/O through its 10-pin I/O connector. Pixy also supports USB 2.0 through its mini-USB connector. You can configure which interface Pixy uses through the configure dialog in PixyMon. The "Data out port" parameter in the "Interface" tab determines the output port.



If you hover the mouse pointer over the "Data out port" text, a help string will be displayed that describes which value corresponds to which type of port.

- SPI - this is the default port that uses 3 wires (pins 1, 3, and 4 of the I/O connector) and is used to communicate with Arduino
- I2C - this is a multi-drop 2-wire port (pins 5 and 9 of the I/O connector) that allows a single master to communicate with up to 127 slaves (up to 127 Pixys).
- UART - this is the common "serial port" (pins 1 and 4 of the I/O connector). Pixy receives data via pin 1 (input) and transmits data via pin 4 (output).
- analog/digital x - this will output the x value of the largest detected object as an analog value between 0 and 3.3V (pin 3). It also outputs whether an object is detected or not as a digital signal (pin 1 of the I/O connector).
- analog/digital y - this will output the y value of the largest detected object as an analog value between 0 and 3.3V (pin 3). It also outputs whether an object is detected or not as a digital signal (pin 1 of the I/O connector).



The serial protocol

Whether you're using SPI, I2C or UART serial, the protocol is exactly the same.

- The protocol is data-efficient binary.
- The objects in each frame are sorted by size, with the largest objects sent first.
- You can configure the maximum number of objects sent per image frame ("Max blocks" parameter).
- SPI and I2C operate in "slave mode" and rely on polling to receive updates.
- Each object is sent in an "object block" (see below).
- A new image frame is indicated by two sync words (0xaa55) sent together.
- All values in the object block are 16-bit words, sent least-signifcant byte first (little endian). So, for example, to send the sync word 0xaa55, you need to send 0x55 (first byte) then 0xaa (second byte).

Object block format

Bytes	16-bit words	Description
0, 1	0	sync (0xaa55)
2, 3	1	checksum (sum of all 16-bit words 2-6)
4, 5	2	signature number
6, 7	3	x center of object
8, 9	4	y center of object
10, 11	5	width of object
12, 13	6	height of object

So, a typical way to parse the serial stream is to wait for two sync words and then start parsing the object block,

using the sync words to indicate the start of the next object block, and so on. Looking at the source code for Arduino will help – all of the parsing code is there. In particular, look at TPixy.h.

Pixy sends block information to Arduino at 1 Mbits/second which means Pixy can send more than 6000 detected objects per second or 135 detected objects per frame (Pixy processes at 50 frames per second.) But it's possible that the serial link is too slow to send back all of the objects before the next frame is processed and new objects are ready to send over serial. This can especially happen with UART serial, which is typically only 19.2 kbaud (more or less). When this happens, the unsent object blocks from the previous frame are tossed and the new object blocks from the new frame are sent instead. This way, the most recent information is prioritized. And since the objects are sorted by size, the larger the object, the more priority it gets also.

Servo control over SPI

You can also control Pixy's servos over SPI by sending simple little 3-word packets. Like above, each word consists of 16 bytes, little endian.

Bytes	16-bit words	Description
0, 1	0	sync (0xff00)
2, 3	1	servo 0 (pan) position, between 0 and 1000
4, 5	2	servo 1 (tilt) position, between 0 and 1000

You repeat this pattern to update the servo positions on-the-fly.