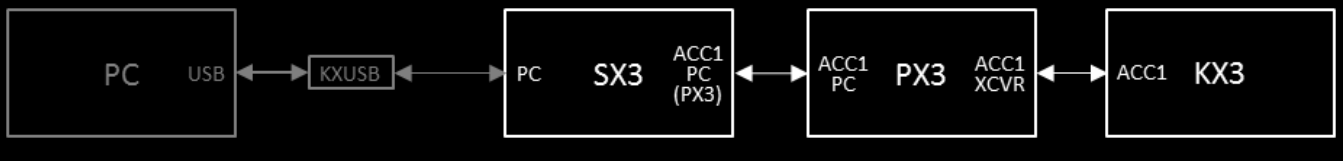
Mouse-n-Click QSY with Zero Beating
Click-n-Mouse Span
(and other Mouse / Keyboard Functions)
with an Elecraft® PX3 and KX3
Proof of Concept

This is a proof of concept of Mouse-n-Click QSY with Zero Beating, Click-n-Mouse Span (and other Mouse / Keyboard Functions) in conjunction with an Elecraft® PX3 and KX3.

Introduction

In the context of this whitepaper, the term “SX3” is used to refer to the prototype hardware and firmware.

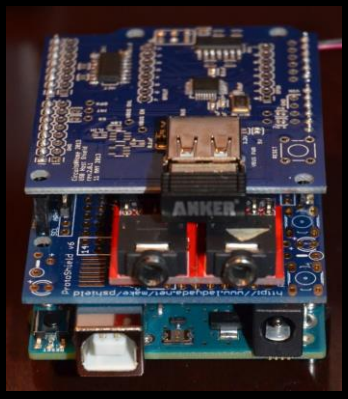
The SX3 connects to the ACC1 port of the PX3,



Note: The RX I/Q connection between the PX3 and KX3 (and optionally between the PC and PX3) is not shown.

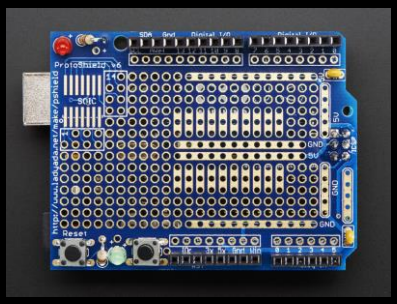
The use of a PC (and Elecraft® KXUSB computer interface cable) in conjunction with the SX3 is strictly optional.

Prototype Hardware

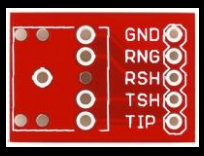


On the bottom of this stack is an Arduino Uno,

On top of the Uno is an Adafruit Arduino prototype shield,



Two SparkFun 3.5 mm audio jack breakout boards are mounted on the prototype shield,

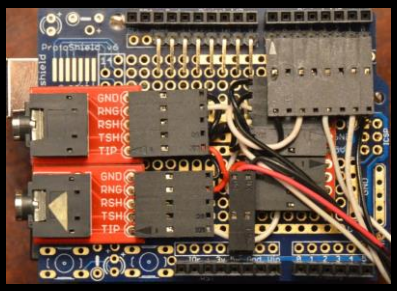


One audio jack connects to a PC, tablet or other device capable of monitoring / controlling the PX3 and KX3. The other audio jack connects to the ACC1 port of the PX3.

The audio jacks are connected to the Uno via a SparkFun MAX3232 transceiver breakout board,

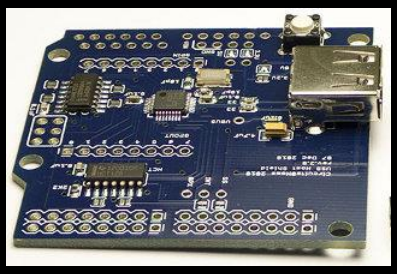
The MAX3232 converts the TTL signals of the Uno to RS232 and vice versa.

The prototype shield with the SparkFun 3.5 mm audio jack and MAX3232 transceiver breakout boards,



Note: You can just barely see the red SparkFun MAX3232 transceiver breakout board peeking out from underneath the connectors.

On the top of the prototype shield is a Circuits@Home USB host shield,



An Anker 2.4 GHz wireless dongle is plugged into the USB host shield. This dongle supports the Anker mini wireless optical mouse and ultra-slim keyboard combo,

Note: A single dongle supports both the mouse and the keyboard.

The Uno is connected to a Parallax 2-row by 16-character backlit serial LCD.

Note: The Parallax LCD is not necessarily a recommended technology choice for this application.

Mouse-n-Click QSY with Zero Beating

The mouse is used to control marker A.

Tapping the left mouse button, QSY's the KX3's VFO A frequency to the marker A frequency and sends an “auto spot” command to the KX3 (i.e., automatic zero beating).

If you hold the left mouse button and move the mouse right and left, the marker A frequency and the KX3's VFO A frequency are increased and decreased in one Hertz increments, respectively (i.e., manual zero beating).

Note: Unfortunately, the VFO A frequency on the PX3 is shown in tens of Hertz.

Click-n-Mouse Span

If you hold the middle mouse button (wheel) and move the mouse up and down, the span is decreased and increased between 2, 5, 10, 20, 50, 100 and 200 kHz, respectively. The span increases or decreases one increment per Click-n-Mouse.

Note: Obviously, a better user experience would involve directly controlling the span using the mouse wheel without the need to hold the middle mouse button (wheel) or move the mouse. This will require a little more firmware.

Other Mouse / Keyboard Functions

Unfortunately, the existing PX3 remote-control commands don't support a two-dimensional cursor nor the ability to write to the PX3 display.

The SX3 leverages a “virtual” two-dimensional cursor. The virtual cursor works in conjunction with the external LCD display.

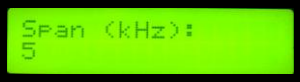
As with the QSY function, marker A is used as a horizontal cursor. However, the SX3 requires you to “visualize” the vertical position of the cursor.

If you scroll the virtual cursor off the PX3 screen to the right, you will see the right-hand switch names (corresponding to the relative vertical position of the cursor) displayed on the external LCD,

If you tap the right mouse button over a switch, the Tap function associated with the switch is executed. If you hold the right mouse button over a switch for longer than 500 ms, the Hold function associated with the switch is executed. For example, if you click the right mouse button over the Display switch, the display toggles between the spectrum and waterfall modes.

Functions such as Display and Marker A / Marker B execute immediately. The other right-hand switch functions require input. The SX3 opts to use the keyboard for input. For example, if you hold the right mouse button over the Span switch, you are prompted to enter the span in kHz,

The span is entered using the Anker wireless keyboard,



If you scroll the virtual cursor to the bottom of the PX3 screen, you will see the bottom switch names (corresponding to the relative horizontal position of the cursor) displayed on the LCD.

As with the right-hand switches, tapping the right mouse button over a switch executes the Tap function. Holding the right mouse button over a switch for longer than 500 ms executes the Hold function.

Of course, any one of the PX3 menu functions can be assigned to any function switch, FN1 through FN8. This is a function of the PX3 and not the SX3.

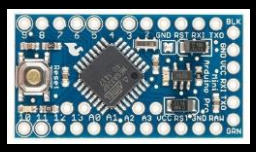
Pass-through

The SX3 may be connected to a PC, tablet or any other device capable of monitoring / controlling the PX3 and KX3. This includes the Elecraft® PX3 and KX3 Utility. Each byte received on the RS232 port connected to the PC is sent to the ACC1 port on the PX3 and vice versa.

Note: The PC should be connected directly to the PX3 ACC1 port in upgrading the PX3 and KX3 firmware in order to ensure the most reliable communication.

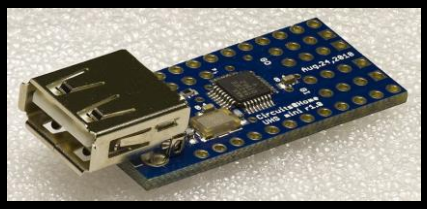
Other Considerations

A more compact SX3 may involve a 3.3 V, 8 MHz, 32 kB Flash, 2 kB RAM and 1 kB EEPROM Arduino Pro Mini,



Dimensions: 1.3" x 0.7" x 0.1"

And a Circuits@Home Arduino Pro Mini USB Host Shield,



Note: There is a 5 V, 16 MHz Arduino Pro Mini, however, it is not compatible with the 3.3 V Circuits@Home Mini USB Host Shield.

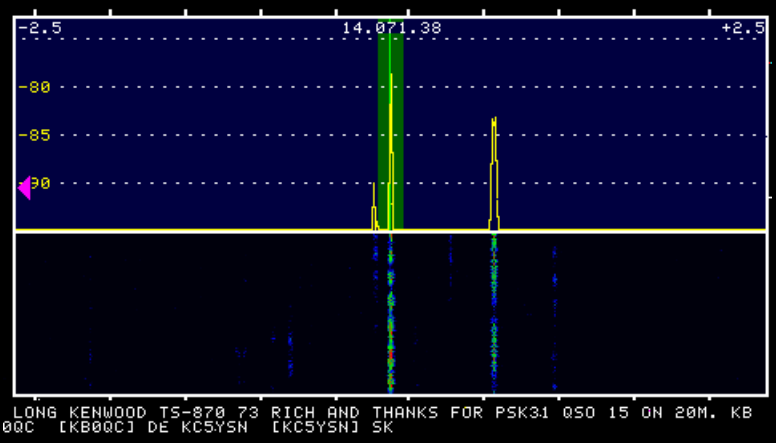
The Arduino Pro Mini riding directly on the Circuits@Home Mini USB Host Shield,

Note: The 3.3 V Arduino Pro Mini / Circuits@Home Mini USB Host Shield combination is functionally equivalent to / compatible with the Arduino Uno / Circuits@Home (full-size) USB Host Shield combination only it runs at 8 MHz, rather than 16 Mhz.

The same SX3 hardware can be inserted in between the PX3 and the KX3,

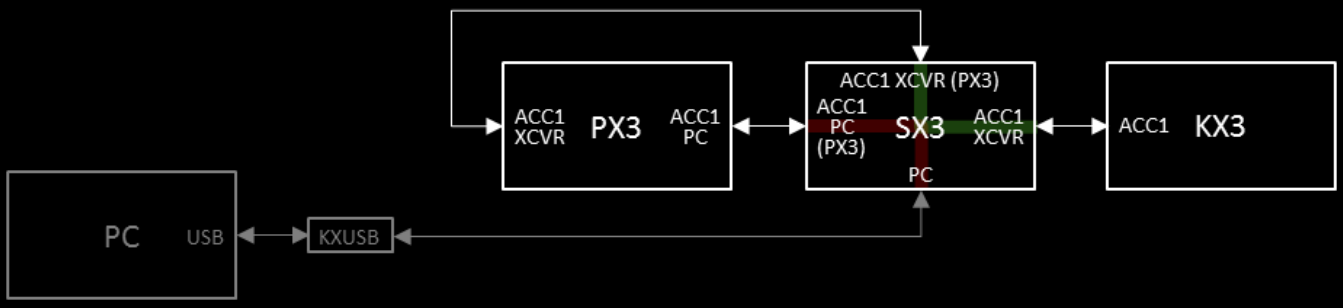
This allows the SX3 to monitor remote-control commands (and responses) sent between the PX3 and SX3. For example, the SX3 can monitor decoded text sent from the KX3 to the PX3.

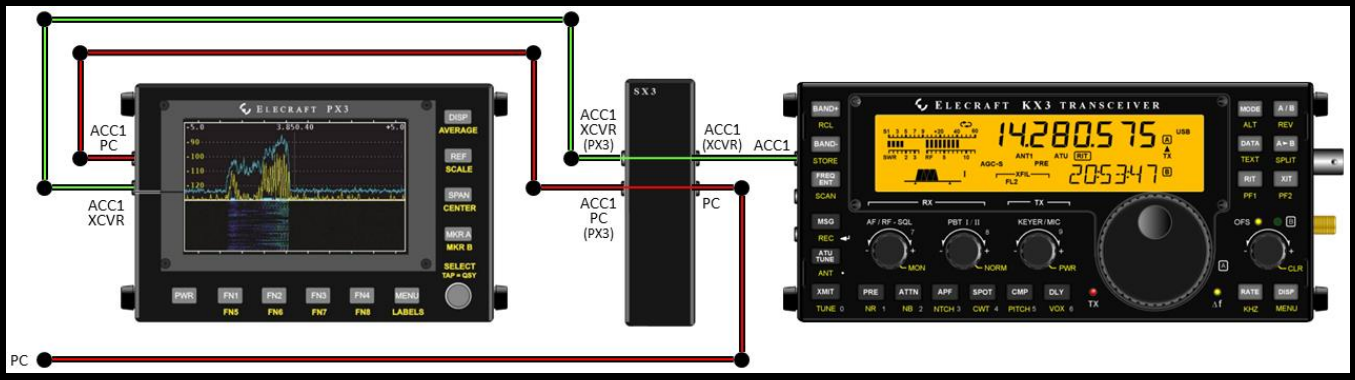
This is a native PX3 image capture (captured using the Elecraft® PX3 Utility) of a QSO between KB0QC and KC5YSN. The SX3 firmware supports call sign recognition. The SX3 inserts the recognized call sign (in brackets) into the decoded text stream,



In and of itself, this use of call sign recognition is worthless. But imagine leveraging the recognized call sign as part of a macro in responding to a CQ.

In theory, the same SX3 could control the PX3 and monitor the remote-control commands / responses sent between the PX3 and SX3 (aka, “dual-mode”). This would require another pair of RS232 ports,





The Arduino Uno has one hardware UART. In addition, the SX3 leverages a software serial port. The “dual-mode” SX3 configuration would benefit from an Arduino Due with its hardware UART and three hardware USARTs.

Note: The Circuits@Home USB host shield is pin-compatible with the Arduino Due.

Alternatively, a more compact “dual-mode” SX3 may leverage a 3.3 V, 72 MHz, 256 kB Flash, 64 kB RAM and 2 kB EEPROM Teensy 3.1 USB Development Board with its three hardware UARTs,

Dimensions: 1.4" x 0.7"

Note: The Teensy 3.1 is compatible with the Circuits@Home Mini USB Host Shield, but not pin-compatible.

In theory, the SX3 can be powered via the PX3 KEYBD (keyboard) port.

Conclusion

There is no need for a separate LCD display given the PX3's 480- by 272-pixel color LCD display. Hopefully, the good folks at Elecraft® will externalize a remote-control command to write to the PX3 menu area.

The PX3 firmware should be modified to display the KX3 VFO A frequency in Hertz, rather than tens of Hertz.

The basic Mouse-n-Click QSY functionality involves ~25 lines of firmware (in addition to the Arduino USB Host Library),

```
#include <SoftwareSerial.h>
#include <hidboot.h>
#include <usbhub.h>

#define PX3_RX_PIN 7
#define PX3_TX_PIN 5

SoftwareSerial PX3(PX3_RX_PIN, PX3_TX_PIN);

#define ELECRAFT_MAX_CMD_LEN 17

        long marker_A_frequency    = 14070000;
        int  span                  = 5;  // KHz
unsigned long left_button_down_time = 0;

void MouseRptParser::OnMouseMove(MOUSEINFO *mi) {
   char temp[ELECRAFT_MAX_CMD_LEN + 1];
   if (mi->dX) {
     if (left_button_down_time)
       marker_A_frequency += mi->dX;
     else
       marker_A_frequency += mi->dX * span;
     sprintf(temp, "#MFA+%011ld;", marker_A_frequency);
     PX3.write(temp);
     if (left_button_down_time)
       PX3.write("#QSY1;");
   }
```

```
}

void MouseRptParser::OnLeftButtonUp(MOUSEINFO *mi) {
   left_button_down_time = 0;
}

void MouseRptParser::OnLeftButtonDown(MOUSEINFO *mi) {
   left_button_down_time = millis();
   PX3.write("#QSY1;");
}

void setup() {
   PX3.begin(38400);
   Usb.Init();
   HidComposite.SetReportParser(0, (HIDReportParser*)&KbdPrs);
   HidComposite.SetReportParser(1, (HIDReportParser*)&MousePrs);
   HidKeyboard.SetReportParser(0, (HIDReportParser*)&KbdPrs);
   HidMouse.SetReportParser(0, (HIDReportParser*)&MousePrs);
   PX3.write("#MKA1;");             // Marker A On
   PX3.write("#MFA+00014070000;");  // Marker A Frequency 14.070.000
   PX3.write("#QSY1;");             // QSY to Marker A Frequency
   PX3.write("#SPN000050;");        // Span to 5,000 Hz
}
```

This concludes my proof of concept of Mouse-n-Click QSY with Zero Beating, Click-n-Mouse Span and other Mouse / Keyboard Functions in conjunction with an Elecraft® PX3 and KX3.

73's

Joe Stone
KF5WBO