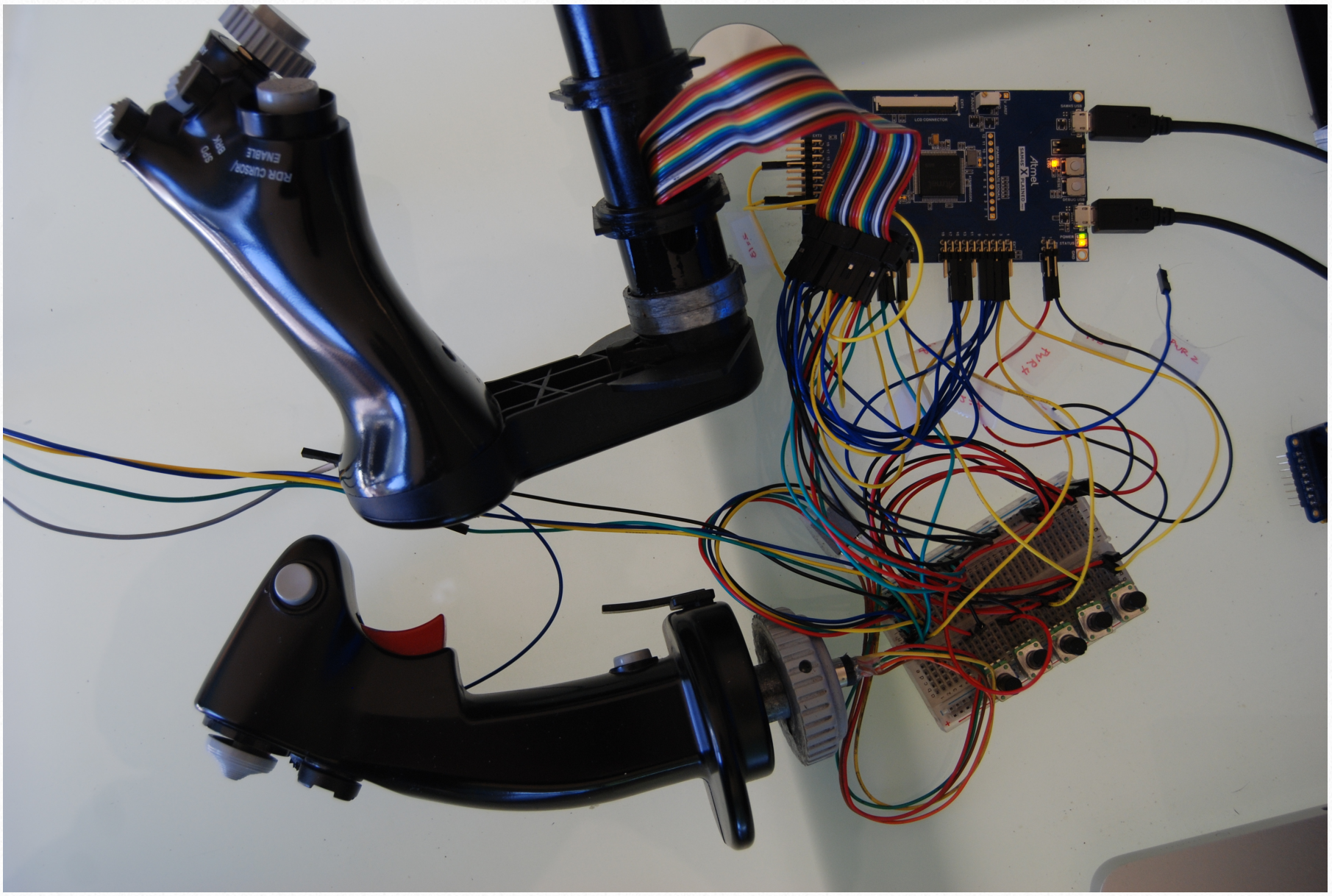
Jonah Tsai

Hempstick Cougar Demo

v.1.0.1



1

Overview

The purpose of this demonstration project & document is to show you how to use Hempstick to upgrade ThrustMaster Cougar's outdated electronics.

I will demonstrate how to configure a stock Hempstick source code for a Cougar, and wire up some representative potentiometers for X, Y, and Z axes, which are the two stick axes, and the throttle axis. Then we will demonstrate wiring up to a real Cougar stick and TQS throttle to a SAM4S XPLAINED Pro (XPro for short) board.

Of course, again, I will omit the critical part of how to get TARGET to accept Hempstick, in fear of dreaded US DMCA law. Unfortunately, I live in the United States, I must obey the laws of the land of the brave or risk getting sued.

Objectives

The TM Cougar was release in the year 2002. In those days, 24MHz, 8 bit MCUs with 10 bit ADCs are the norm. Today, 120Mhz, 32bit ARM processor with 12bit ADCs are the norm. The current off-the-shelf controllers for the "simmers" are still running at 8 bit, 24Mhz, but with an upgraded 12bit ADC.

At the end of the this demo project, you will end up with a SAM4 XPro board with a burned firmware, and a couple of wiring tables ready for your Cougar upgrade project. The physical wiring and case modification etc. is up to you. This is because, I do not know

where you are going to locate your Hempstick board. There is not enough space in the stick base, unless you make a new base cover to make more room. There seems to be enough space to cramp in the SAM4S XPro board, but it's a very very tight fit. In addition, even if you do put it in the throttle base, you would have to run a long cable to the stick base. That means running two long analog wires to the main X and Y axes, plus 5 wires for the stick, plus the Vcc and GND lines. The two long signal lines to the X & Y axes are a bit worrisome... they WILL pick up noises. That's, I guess, why the new Warthog have two separate MCUs one for the stick, one for the throttle. This is a far superior arrangement. So, even though I am demonstrating how to use one Hempstick SAM4S XPro board to connect both the Cougar stick & throttle, I really recommend that you use 2x Hempstick SAM4S XPro boards, one for the stick, one for the throttle. If you want, add one more for the rudder. It's USD $29 apiece... for the price of a BU0836X, you can almost pay for all 3x of the SAM4S XPro boards.

Assumptions

We will assume that you have already installed Atmel Studio v.6.2 or later by following the instructions in the Hempstick User Guide. We will also assume that you have already downloaded the libHemp and Hempstick and opened them in a directory on disk (both must be under the same directory to avoid having to change the project structure inside Atmel Studio).

I will also assume that you have read through the Hempstick Rudder Demo document, so I will not repeat what you would have learned from that document.

I will also assume that you have completed the steps in the Preparation of the Hempstick Rudder Demo document. That is, you have tested your Atmel Studio, EDBG debugger via USB, and have downloaded a fresh copy of libHemp and Hempstick source code. If you have not, please refer to that document and follow the steps in the Preparation chapter.

In this document, we will not explain the basic concepts and the why things are done that way. These are explained in the User's Guide and the Rudder Demo documents. We will

be very brief in describing what needs to be done specifically for Cougar and why doing this way with Cougar. No more spoon feeding from the corn fed Yanks.

What You Need

- Atmel Studio v.6.2 or newer installed.
- MSysGit & SmartGit (optional)
- libHempstick & Hempstick source code
- An Atmel SAM4S XPLAINED Pro board
- A Thrustmaster Cougar, ready to be operated on.
- Some wires and connectors to connect the potentiometers to the SAM4S XPLAINED Pro board
- Tools to do the wiring (that depends on the methods you decide to do the wiring)

2

Cougar Axes & Buttons Wiring & Mapping

In this chapter we will examine what axes & buttons Cougar has internally and what we intend to do with them, how to connect them to a Hempstick board, and then produce an axis mapping and a button mapping for Hempstick.

Wiring

Potentiometer Wiring

Cougar Axis Name	USB Axes Name	ADC Channel	SAM4S Pin	SAM4S XPro pin	Throttle Connector 261 Position
Stick X	X	0	PA8	Ext1 : 3	n/a
Stick Y	Y	4	PB0	Ext2 : 3	n/a
Throttle	Z	5	PB1	Ext2 : 4	n/a
Radar Cursor X	Rx	7	PB3	Ext 3 : 14	20
Radar Cursor Y	Ry	8	PA21	Ext 2 : 13	19
RNG	Rz	9	PA22	Ext 2 : 14	13
ANT	Slider	13	PC29	Ext 3 : 3	14

Don’t forget to wire up Vcc & GND pins for the 261 header!

GND	5	10	12	16	18	22

Vcc	15	21

The following is the mapping for the header 261 buttons.

SAM4S Pin	Permanent USB Button Number (0-index)	SAM4S XPro Header Pin	Cougar Header 261 pin (1-index)	Corresonding GND pin on header 261 (1-index)
PA24	18	Ext 1: 5	17 (Enable)	18
PA25	19	Ext 1: 6	1 (VHF)	5
PA23	20	Ext 1: 7	2 (UHF)	5
PA1	21	Ext 1: 9	3 (IFF IN)	5
PA11	22	Ext 1: 15	4 (IFF OUT)	5
PA13	23	Ext 1: 16	11 (Uncage)	12
PA12	24	Ext 1: 17	6 (Dogfight AFT)	10
PA14	25	Ext 1: 18	7 (Dogfight FWD)	10
PC19	26	Ext 2 : 7	8 (SPD BRK AFT)	10
PC27	27	Ext 2 : 10	9 (SPD BRK FWD)	10

We are not using any original Cougar PCBs. We ditch them altogether. You wire directly between the SAM4S pins and the Cougar pots and buttons. I have decided to omit the 3 rudder potentiometers because DirectInput can only take up to 8 axes and with the 3 rudder axes we would have total of 9 axes, even though Hempstick SAM4S XPro can support up to 16 ADC channels.

For the potentiometers, you need to wire the 3.3V power to one side, the GND to the other, and the the middle to the ADC channel pins.

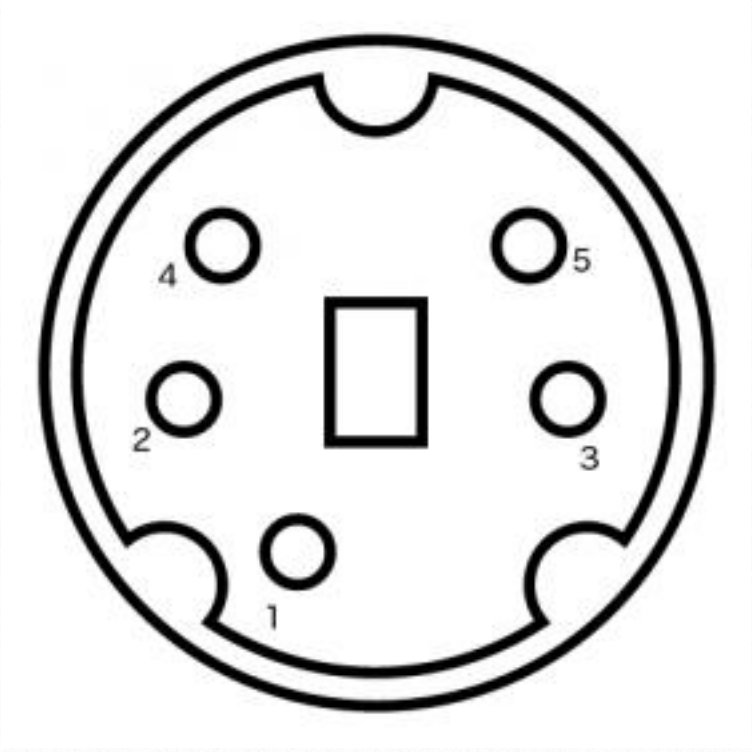
CAUTION: The SAM4S MCU is a 3.3V device. NO 5V!!!

The Hempstick SAM4S XPro uses an on-die hardware SSC module to read the TMStick. However this requires some special wiring.

SAM4S Pin Name	SAM4S XPro Header Pin	Connects to	Stick PS2 Pin	SSC Signal Name
PC26	Ext 2 : 9	Ext 2 : 8	2	
PA20	Ext 2 : 8	Ext 2 : 9	2	RF
PA18	Ext 1: 4		4	RD
PA19	Ext 1 : 8		3	RK
Vcc 3.3V			1	
GND			5	

PC26 (Ext 2 : 9) is the Timer Clock TIAO5, i.e TC1 channel 2. This is used to generate a 1KHz to trigger the SSC module. PA20 pin is the SSC trigger pin RF. This signal is also used as the lock signal to the 3x cascaded 8 bit jam type parallel in serial out buffer on TMStick. The PA18 pin is the read pin of the SSC when data is shifted out of the TMStick buffer chip. The PA19 pin is the SSC clock pin. It sends the clock pulse for shifting out serial bits.

TMStick PS2 Connector Pinout



TMStick Color Code for Cougar

Pin	Color
1	Brown
2	Red
3	Orange
4	Yellow
5	Green

TMStick Color Code for Warthog

Pin	Color
1	Black
2	Brown
3	Red
4	Orange
5	Yellow

Please note that the color code on the PS2 connector is based on my samples of Cougars and Warthogs, yours might not be exactly the same.

That’s all the wiring you need.

Shortcut Using Preconfigured Set of Configuration Files

If you don't want to change anything, but just want it to work with a set of preconfigured files, ignore the following Axis Mapping & Button Mapping sections, and do the followings.

1. Go to the src/config/Cougar/ directory, copy all files there and paste into src/config/ directory, overwrite all the files there.

2. Launch Atmel Studio, open the Hempstick.atsln file, select the Hempstick_SAM4S_XPLAINED_Pro project.

3. Plug in the SAM4S XPro board's Debug USB and SAM4S USB port to the host computer.

4. Click [Start Debugging...] button or do menu [Debug] -> [Start Without Debugging].

This will compile the whole firmware with the set of preconfigured files, and burn the firmware to the SAM4S XPro board, if there is no error. If everything goes well, the new Hempstick with Cougar will show up in your Devices & Printers page.

If you want to do some light configuration, then read on.

Axis Mapping

Please consult the axes mapping table and make sure the axis mappings in the file src/config/conf_hempstead.h match.

```
#ifdef CONF_BOARD_SAM4S_XPLAIN_PRO
        .channel_flags = {ADC_CHANNEL_ENABLE_MASK, 0, 0, 0, ADC_CHANNEL_ENABLE_MASK, ADC_CHANNEL_ENABLE_MASK, 0, ADC_CHANNEL_ENABLE_MASK, ADC_CHANNEL_ENABLE_MASK, ADC_CHANNEL_ENABLE_MASK, 0, 0, 0, ADC_CHANNEL_ENABLE_MASK, ADC_CHANNEL_ENABLE_MASK, 0},
        .channel_mapping = {0, UINT8_MAX,  UINT8_MAX,  UINT8_MAX, 1, 2,  UINT8_MAX, 3, 4, 5,  UINT8_MAX,  UINT8_MAX,  UINT8_MAX, 6, 7,  UINT8_MAX},
#elif defined(CONF_BOARD_ARDUINO_DUE)
```

First of all there is an ADC enabling table, the .channel_flags. This, as usual, is a C 0-indexed array. Element 0 is for ADC0 channel. Element 1 is for ADC1 channel, etc. You use this array to enabled each of the 16 ADC channels.

The second table is the ADC channel to USB axis mapping, .channel_mapping. The values are only used if the corresponding channel is enabled. The values specify which ADC channels map to which USB axis, 0-index again.

Button Mapping

There two kinds of buttons. One is the real buttons, each hardwired to an MCU pin enabled for GPIO. The other kind is synthesized buttons like the ones from TMStick. These are basically fake buttons that we can pull out of the thin air. You don't have to have any real button, USB does not care. It only care whether a bit position in the report it receives has a 1 or 0 value. Even though the TMStick has real physical buttons, they are not really hardwired to MCU pins or a matrix. If you use a matrix of button, this would be the kind of buttons you implement.

For hardwired buttons, you need to do two things.

1. Enabled the MCU pin as GPIO input, with a pull-up resistor, or pull-down resistor, and you need to configure it to use the built-in hardware debouncer.

2. You need to specify which USB button bit each of these pins map to.

For #1 above. You need to modify the hw_pin_configuration_table in the file src/config/conf_hempstead.c file.

```
hw_pin_configuration_table g_hw_pin_conf_table = {
                .mutex = NULL,
```

```
.pin = {
    ...
    {.pin = PIO_PA24_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PA25_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PA23_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PA1_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PA11_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PA13_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PA12_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PA14_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PC19_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)},
    {.pin = PIO_PC27_IDX, .conf = HW_PIN_ENABLE_MASK, .mode = (PIO_TYPE_PIO_INPUT | PIO_PULLUP |
PIO_DEBOUNCE)}
    }
    ...
```

For each entry, you specify which pin you wish to configure. You enable it, and then specify the flag (or mode) the pin should be configured with. The above table are the button pins that need to be configured for hardwired buttons for a Cougar. Please cross check this table with the header 261 button table listed previously.

The total number of entries (rows) must also be specified in the src/config/conf_hempstead.h.

```
// PIN Configuration
#ifdef CONF_BOARD_SAM4S_XPLAIN_PRO
#       define CONF_NUM_PINS                                   23
```

This number must match exactly the number of rows in the above table. If they don't match, you will most likely have a crash.

This is yukky! It will change in the future. But for now, make sure they match.

Next, we must tackle the pin to USB button mapping. You must specify which GPIO pin value should be mapped to which USB button. Any voltage level change event will trigger the button task to read the pin value and set the corresponding USB button bit accordingly. For Cougar, according to the button mapping table listed previously, you should have the button mapping as the following in src/config/conf_hempstick.c.

```
rtos_button_data_t g_rtos_button_data = {
        .data = NULL,
        .num_button = 0,
        .hat_data = NULL,
        .num_hat = CONF_NUM_HAT,
        .mutex = NULL,
        .rtos_internal_task_semaphore = NULL,
        .rtos_task_semaphore = NULL,
#ifdef ID_PIOA
        .ports[0].button_conf[0].flags =  0x0000,
        .ports[0].button_conf[1].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[0].button_conf[1].data_position = 21,
        .ports[0].button_conf[2].flags =  0x0000,
        .ports[0].button_conf[3].flags =  0x0000,
        .ports[0].button_conf[4].flags =  0x0000,
        .ports[0].button_conf[5].flags =  0x0000,
        .ports[0].button_conf[6].flags =  0x0000,
        .ports[0].button_conf[7].flags =  0x0000,
        .ports[0].button_conf[8].flags =  0x0000,
        .ports[0].button_conf[9].flags =  0x0000,
        .ports[0].button_conf[10].flags =  0x0000,
        .ports[0].button_conf[11].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[0].button_conf[11].data_position = 22,
        .ports[0].button_conf[12].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[0].button_conf[12].data_position = 24,
        .ports[0].button_conf[13].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[0].button_conf[13].data_position = 23,
        .ports[0].button_conf[14].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[0].button_conf[14].data_position = 25,
        .ports[0].button_conf[15].flags =  0x0000,
        .ports[0].button_conf[16].flags =  0x0000,
```

```
        .ports[0].button_conf[17].flags =  0x0000,
        .ports[0].button_conf[18].flags =  0x0000,
        .ports[0].button_conf[19].flags =  0x0000,
        .ports[0].button_conf[20].flags =  0x0000,
        .ports[0].button_conf[21].flags =  0x0000,
        .ports[0].button_conf[22].flags =  0x0000,
        .ports[0].button_conf[23].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[0].button_conf[23].data_position = 20,
        .ports[0].button_conf[24].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[0].button_conf[24].data_position = 18,
        .ports[0].button_conf[25].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[0].button_conf[25].data_position = 19,
        .ports[0].button_conf[26].flags =  0x0000,
        .ports[0].button_coßnf[27].flags =  0x0000,
        .ports[0].button_conf[28].flags =  0x0000,
        .ports[0].button_conf[29].flags =  0x0000,
        .ports[0].button_conf[30].flags =  0x0000,
        .ports[0].button_conf[31].flags =  0x0000,
#endif


#ifdef ID_PIOC
        .ports[2].button_conf[0].flags =  0x0000,
        .ports[2].button_conf[1].flags =  0x0000,
        .ports[2].button_conf[2].flags =  0x0000,
        .ports[2].button_conf[3].flags =  0x0000,
        .ports[2].button_conf[4].flags =  0x0000,
        .ports[2].button_conf[5].flags =  0x0000,
        .ports[2].button_conf[6].flags =  0x0000,
        .ports[2].button_conf[7].flags =  0x0000,
        .ports[2].button_conf[8].flags =  0x0000,
        .ports[2].button_conf[9].flags =  0x0000,
        .ports[2].button_conf[10].flags =  0x0000,
        .ports[2].button_conf[11].flags =  0x0000,
        .ports[2].button_conf[12].flags =  0x0000,
        .ports[2].button_conf[13].flags =  0x0000,
        .ports[2].button_conf[14].flags =  0x0000,
        .ports[2].button_conf[15].flags =  0x0000,
        .ports[2].button_conf[16].flags =  0x0000,
        .ports[2].button_conf[17].flags =  0x0000,
        .ports[2].button_conf[18].flags =  0x0000,
        .ports[2].button_conf[19].flags =  RTOS_BUTTON_PIN_ENABLED_MASK, .ports[2].button_conf[19].data_position = 26,
        .ports[2].button_conf[20].flags =  0x0000,
        .ports[2].button_conf[21].flags =  0x0000,
```

```
        .ports[2].button_conf[22].flags =  0x0000,
        .ports[2].button_conf[23].flags =  0x0000,
        .ports[2].button_conf[24].flags =  0x0000,
        .ports[2].button_conf[25].flags =  0x0000,
        .ports[2].button_conf[26].flags =  0x0000,
        .ports[2].button_conf[27].flags =  RTOS_BUT-
TON_PIN_ENABLED_MASK,
.ports[2].button_conf[27].data_position = 27,
        .ports[2].button_conf[28].flags =  0x0000,
        .ports[2].button_conf[29].flags =  0x0000,
        .ports[2].button_conf[30].flags =  0x0000,
        .ports[2].button_conf[31].flags =  0x0000,
#endif
```

Please double check these entries match the mapping table listed previously.

Also, please make sure the TMStick task is enabled in the src/config_hempstead.h.

```
#define CONF_ENABLE_TM_STICK_IN_BUTTON
                        1
```

When the TMStick is enabled, It will map the stick buttons to the first 19 or 20 buttons, plus an 8 way Hat Switch.

Here's the table of TMStick buttons for your reference. The buttons must be mapped to USB in exactly in the following listing, otherwise if you use TARGET, the button name constants in TARGET will get you the wrong USB buttons.

Position in Windows	Position in Hempstick Internally	Cougar Stick Function	Warthog Stick Function
1	0	Trigger 1st Stage	Trigger 1st Stage
19	1	N/C	CMS Push
20	2	N/C	N/C
2	3	WPN/REL	WPN/REL
3	4	Nose Wheel	Nose Wheel
4	5	Pinky Shift	Pinky Shift
5	6	Master Mode Control	Master Mode Control

6	7	Trigger 2nd Stage	Trigger 2nd Stage
HAT N	8	Trim Up	Trim Up
HAT E	9	Trim RWD	Trim RWD
HAT S	10	Trim Dn	Trim Dn
HAT W	11	Trim LWD	Trim LWD
7	12	TMS Up	TMS Up
8	13	TMS RWD	TMS RWD
9	14	TMS Dn	TMS Dn
10	15	TMS LWD	TMS LWD
11	16	DMS Up	DMS Up
12	17	DMS RWD	DMS RWD
13	18	DMS Dn	DMS Dn
14	19	DMS LWD	DMS LWD
15	20	CMS Up	CMS Up
16	21	CMS RWD	CMS RWD
17	22	CMS Dn	CMS Dn
18	23	CMS LWD	CMS LWD

Before you compile and burn the firmware, be sure to change the VID/PID and device name to what you like in the file src/config/conf_usb.h. Put your call sign on the device name and have it show up in Windows Devices & Printers.

However, I cannot tell you which VID/PID to use. You must chose wisely to avoid conflicts in your OS. A good place to find a large list of VID/PID is the Linux USB ID Repository. Go there, find one that is suitable and program yours.

A word of advise on choosing the VID/PID pair. It's probably best to choose one that is a USB HID class device that requires no custom drivers to be loaded in the Windows (or other OS). This is because the VID/PID pair is used on Windows to identify which driver needs to be loaded to drive the newly plugged in USB device. If you choose a pair that has a custom driver that come OOTB with Windows, Windows might actually load it for your Hempstick and attempt to send custom data for that VID/PID to Hempstick. Obviously Hempstick will not understand that custom data output to it. Even worse, that custom driver might expect certain USB endpoints in the USB device product it think it is connecting to. A badly written driver might actually crash, and drag down the whole OS with it, because these are kernel mode drivers. You very likely will get a blue screen for a kernel mode driver crash.

Remember, when you use other people's VID/PID, you are masquerading as that USB device. The behavior of Hempstick and that one must be similar enough for it to work right. Simple enough, just choose another joystick's VID/PID that you don't have.

That's it. Compile, burn, and Test.

Again, I cannot tell you how to make Hempstick show up in Target. But it's possible. I have tested it and found it working fine even in the TARGET Device Analyzer. I must also warn you, that it might be illegal under DMCA to disseminate the information on how to use Hempstick or any other USB device in TARGET. Although it's not illegal for you to program Hempstick with other people's VID/PID, it might be illegal to use TARGET in such an

unauthorized manner. I don't know, I am not a lawyer. I am just a bit cautious about stepping out of line against the LAW.

I would also highly recommend that you wire up one thing, test, and the wire up another. For instance, start with the X axis. Wire up just 3 wires to the X pot, and see if that works. If it works, then, wire up more pots and test. This way, you don't end up sorting with a tangle of mess of wires.

I would also recommend using a ribbon cable and a 24 socket IDC header to connect to the throttle header 261 and split the other end to connect to the SAM4S XPro board. For this split ends, you'd either need to solder them or use single header connector sockets like this one, http://www.digikey.com/product-search/en?x=0&y=0&lang=en&site=us&keywords=A26962.

Again, how you are going to wire it up is up to you.