UI/ RF Conditioning for Red Pitaya SDR

This describes Laurence Barker G8NJJ's Arduino sketch-controlled RF front end for a Red Pitaya based SDR.

The user interface is needed for monitoring after integration to the SDR software radio. Almost all settings will be set by the SDR software. A "debug" mode allows settings to be controlled for development and calibration.

Settings Available

For the signal switch/routing board:

Setting	Normally Set By	Operational	Debug
TX Antenna (1,2,3)	SDR software, HPSDR Alex norms	Display current antenna selection	Be able to select 1,2 or 3
RX1 (TX ant, aux1, aux2, aux3/TV)	SDR software, HPSDR Alex norms ("bypass" auto selected aux1-3)	Display setting	Be able to select between Tx ant / aux1 / aux2 / aux3
RX2 source	Not set.	Select Ant3 or Aux3. Enduring (eeprom) setting from the UI	Select Ant3 or Aux3.
TX enable	SDR firmware	SDR firmware, ANDed with "VSWR OK" signal	Be able to select TX or RX
RX1 attenuators	SDR software following HPSDR Alex norms	Display setting	Be able to select 0/10/20/30dB
RX2 attenuators	SDR software, possibly; or follow RX1 setting	Display setting	Be able to select 0/10/20/30dB
LPF	SDR firmware (C code in Red Pitaya)	Display current TX band	Force TX band selection
BPF	SDR firmware (C code in Red Pitaya)	Display current RX band	Force RX band selection
Fwd / Rev power	n/a	Display forward power, VSWR as bargraph. High VSWR alert.	Display forward power, Reverse power
TX PA current monitoring	n/a	n/a	Display driver, final current (A)
TX PA temp monitoring	n/a	Display temp, operate fan	Display temp
TX PA heatsink fan	Arduino s/w	none	Be able to turn fan on/off

For the ADC/DAC conditioning board:

Setting	Normally Set By	Operational	Debug
RX1 Atten	SDR software following HPSDR Hermes norms	Display attenuation	Set attenuation 0-31dB
RX2 Atten	SDR software following HPSDR Hermes norms	Display attenuation	Set attenuation 0-31dB
TX Atten	SDR software following HPSDR Pennylane/Hermes norms	n/a	Set attenuation 0-31.5dB (0.5dB steps)
Open Collector outputs	SDR software following HPSDR norms	n/a	Set 7 bits, or as a byte

User Interface

Requirements

Normal Mode, RX	Display: • RX1 antenna =TX ant (show as ANT1-3) or aux/1/2/3 • RX2 Ant3/aux3 selection • band • RX1&2 total attenuation Indicate “normal RX” mode (possibly by omission of “debug”) Provide a way to enter debug mode Provide a way to set RX2 selection & store setting to EEPROM Provide a way to set VSWR, temp trip limits
Normal Mode, TX	Indicate “Normal TX” mode Display selected antenna Display forward power, VSWR numerically and as a bar. Display temp or current. Provide a “high VSWR” indication if detected until T/R strobe removed Provide a “high temp” indication if detected until T/R strobe removed (no user input)
Debug Mode, RX	Indicate debug RX mode Provide a way to set RX1 input= TX antenna 1-3 or aux1/2/3 Provide a way to set RX2 input= antenna3 or aux3 Provide a way to set both switched & variable attenuators for RX1 Provide a way to set both switched & variable attenuators for RX2 Provide a way to change band / frequency Provide a way to set & display the 7 OC outputs Provide a way to assert TX Provide a way to exit debug mode
Debug Mode, TX	Indicate debug TX mode Provide a way to set TX antenna 1-3 Display Fwd power, Rev power, temp, PA currents Provide a way to deassert TX Provide a way to set TX attenuation Provide a way to turn on/off the heatsink fan

User Interface Implementation

The user interface executes on an Arduino Micro module. The interface uses an alphanumeric 4x20 line LCD, and a single rotary encoder with click button. Ordinarily the Arduino boots into “normal” mode with most settings remote controlled; the LCD display shows the settings, and can display power and VSWR in TX mode. If a jumper is installed, the Arduino powers up into “debug” mode where all settings can be manually edited.

The top three display lines provide normal display. The bottom line can be scrolled (using the encoder) and can allow settings to be edited. To edit a setting – click on it; that will either advance to the next available value, or open numerical edit (which will finish on the next click)

<table>
<tr>
<td>Normal Mode, RX</td>
<td>
<pre>
12345678901234567890
RX 12.35MHz   !65C!
RX1: ANT1   37dB
RX2: ANT3   17dB
</pre>

Basic display

mode (RX, or RXD for debug)
RX freq (2 DP)
Heatsink temp (with exclamation mark if tripped)
RX1 & RX2 input source and total attenuation

Rotary encoder scrolls the bottom row between a number of lines (lines marked in RED are available in DEBUG mode)

<ol start="0">
Blank line (no display or entry)
RX1 ant: 1/2/3/aux1/aux2/aux3 editable in debug mode
RX2 ant: ANT3 (editable between ANT3 and AUX3)
TX ANT: ANT1 (displays current TX ant; editable in debug mode)
RX1 atten1 (step 0/10/20/30dB) editable in debug mode
RX1 atten2 (editable 0 to 31dB) editable in debug mode
RX2 atten1 (step 0/10/20/30dB) editable in debug mode
RX2 atten2 (editable 0 to 31dB) editable in debug mode
TX atten (editable 0 to 31.5dB) editable in debug mode
OC nnnnnnnn (editable o/p bits) editable in debug mode
RX filt nnnnnnnnnnnn (editable filter select bits in debug mode)
TX filt nnnnnnnn (editable o/p bits in debug mode)
Fan on/off (editable in debug mode)
Enter TX mode
VSWR trip: 3.0 (VSWR threshold; editable)
Temp trip: 57C (heatsink temp trip; editable)
Power bar max: 130W (range of power display; editable)
VSWR bar max: 6.0 (range of VSWR display; editable)
Set EEPROM to defaults

(items in red can only be changed in debug mode)
RX2, VSWR trip, temp trip, display bars: stored to EEPROM
</td>
</tr>
</table>

<table>
<tr>
<td>Normal Mode, TX</td>
<td>
<pre>
12345678901234567890
TX   ANT3   14.25MHz
[bar]   102W
[bar]   3.2
43C   21A
</pre>

Basic display

mode (TX)
TX freq (2 DP)
TX antenna
Power, VSWR as numerical values and bars
Heatsink temp (with exclamation mark if tripped)
PA current

<pre>
High VSWR tripped
</pre>
(displays in the VSWR bar; goes away after T/R removed)

<pre>
High Temp tripped
</pre>
(displays in the power bar; goes away after T/R removed but will not respond to TX commands until temp below limit)

No response to encoder up/down/click
</td>
</tr>
<tr>
<td>Debug mode, TX</td>
<td>
<pre>
12345678901234567890
TXD ANT3   14.25MHz
Fwd: 102W
Rev: 17W
43C   21A
</pre>

mode (TXD)
TX freq (2 DP)
TX antenna
Forward, Reverse power numerically (1DP?)
Heatsink temp (with exclamation mark if tripped)
PA current

Encoder click – return to RX mode
No response to encoder up/down
</td>
</tr>
</table>

In RX mode:

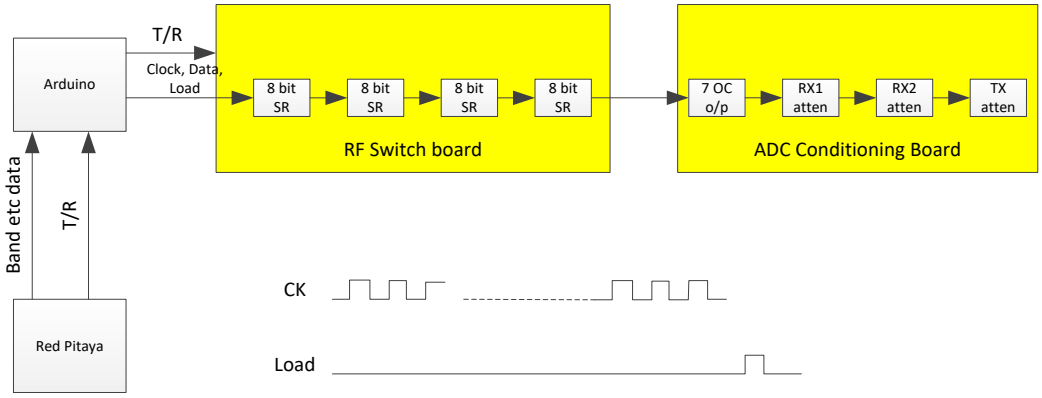
- The encoder up/down starts off scrolling one of the available options (1-18 above).
- If the encoder is clicked enter an appropriate edit mode for that row
- Each edit mode is different and has a way to go back to “not edit mode”
    - Enumerated rows (eg fan on/off ant1/ant2/ant3): each click of the encoder selects the next option; scroll up/down simply moves to previous row
    - Numerical value (eg atten): click enters edit mode; initial atten displayed, cursor placed on the numerical value lowest digit. Encoder up/down

increments/decrements the value between limits, and causes display update. Encoder click exits edit, accepts new value, removes cursor. (Note some numerical values are 1DP fixed point, and TX atten has an increment of 0.5dB).

- Bit field value (eg open collector): click enters edit mode; initial bits displayed, cursor placed on the leftmost bit. Encoder click changes the bit. Encoder up/down scrolls the cursor among the bits; if it moves off the bits to the left, exit edit mode, save bits and remove cursor.

Data Interface (Arduino to RF Hardware)

The "hardware" interface to the Arduino will be an I^2C (2 wire) slave interface and separate T/R strobe. The latter will need to be monitored by interrupt driven software. The interface from the Arduino will be a shift register and separate T/R strobe. The attenuator, filter and switch settings are infrequently changed so one long shift register is OK. The arrangement may be something like:



This leads to shifting 32 bits (RF switching) and 32 bits (ADC conditioning).

Shift Register settings

The RF switch board has a number of relays of its own, and provides drives to external relays / PIN switches to select Lowpass and Bandpass filters. The proposed drive bits (chosen to simplify PCB layout) are:

Byte 7 (last shifted, start of chain)	IC1	0 (last shifted) lsb	BPF 9 (24MHz)
		1	BPF 10 (28MHz)
		2	BPF 11 (50MHz & preamp)
		3	BPF 12 (bypass)
		4	ANT 1
		5	ANT 2
		6	ANT 3
		7	RX BYPASS out
Byte 6	IC2	8 lsb	BPF 5 (10MHz)
		9	BPF 6 (14MHz)
		10	BPF 7 (18MHz)
		11	BPF 8 (21MHz)
		12	RX2 select; 1: Aux3 in; 0: Ant3
		13	RX aux 1 in
		14	RX aux 2 in
		15	RX aux 3 in
Byte 5	IC4	16 lsb	BPF 1 (1.8MHz)
		17	BPF 2 (3.5MHz)
		18	BPF 3 (5MHz)
		19	BPF 4 (7MHz)
		20	RX2 10dB att
		21	RX2 20dB att
		22	RX1 10dB att
		23	RX1 20dB att
Byte 4	IC5	24 lsb	LPF 4 (14 MHz)
		25	LPF 5 (21 MHz)
		26	LPF 6 (28 MHz)
		27	LPF 7 (50MHz)
		29	LPF 1 (1.8MHz)
		29	LPF 2 (3.5MHz)
		30	LPF 3 (7MHz)
		31	Heatsink fan

Byte 3	IC1	0 lsb	Open collector 0
		1	Open collector 1
		2	Open collector 2
		3	Open collector 3
		4	Open collector 4
		5	Open collector 5
		6	Open collector 6
		7	Open collector 7
Byte 2	IC2	8 lsb	RX1 atten 0
		9	RX1 atten 1
		10	RX1 atten 2
		11	RX1 atten 3
		12	RX1 atten 4
		13	RX1 atten 5
		14	RX2 atten 0
		15	RX2 atten 1
Byte 1	IC4	16 lsb	RX2 atten 2
		17	RX2 atten 3
		18	RX2 atten 4
		19	RX2 atten 5
		20	TX atten 0
		21	TX atten 1
		22	TX atten 2
		23	TX atten 3
Byte 0 (first shifted, far end of chain)	IC5	24 lsb	TX atten 4
		25	TX atten 5
		26	0
		27	0
		29	0
		29	0
		30	0
		31	0

(Note the open collector bits, and the antenna select bits, can be different between TX and RX; possibly also RX1 atten, if that needs to be varied to set a sensible ADC input level)

The ADC conditioning board uses Analog Devices HMC624 attenuators. 3 x 6 bit attenuators form an 18 bit word; suggested we should shift whole bytes ie 24 bits, right aligned. Additional 8 bits OC output. The 32 bit Attenuator control word is shifted MSB first; the word would be:

31 0
000000<6 bits TX><6 bits RX2><6 bits RX1>< 8 bits OC>

Arduino Signals & Pin Allocation

The signals required at the Arduino will be:

Interface	Digital	Analogue
LCD out	6	
Rotary encoder in	3	
Serial out clock/data/load	3	
I^2C in (note 3.3V logic at other end!)	2 (I^2C pins)	
T/R in	1 (interrupt)	
T/R out	1	
Fwd, Reverse Power		2
PA current		1
Temp		1

Pin allocation for Arduino Micro:

Interface	Pin
LCD out	EN: D9 RS: D8 D(7:4): D13:10
Rotary Encoder in	A,B: D6. D5 Click: A4
Serial out clock/data/load	MOSI: SPI pin 4 SCK: SPI pin 3 Load: D4
I2C in (3.3V at RP)	SCL=D3; SDA=D2
T/R request in (3.3V at RP)	D0 (note active low)
T/R enable out	D1
Fwd, Reverse Power	A0, A1
PA current	A2
Temp	A3
Power up Jumper	A5. 0=jumper installed use debug mode
Spare	D7

Quad 3.3v to 5V level translator: TI TXS0104ED

Temp sensor: LM35DT or LM35DZ; linear output 10mV / C

The Arduino “clickencoder” library looks suitable. Doesn’t need I/O pin interrupts, and gives other events for the pushbutton eg double clicked. It uses the Timerone library to give a 1ms tick which I may need anyway.

Analogue Inputs, Scaling

Analogue reference = 5V; so a reading of 1023 = 5V

Input	Purpose	Input gain	Scale (at CPU i/p)	Calculation
A0	Fwd power	1	Full scale: VRMS = 110V (approx. 242W)	Vrms = 110N/1024 (V) (Power = $Vrms^2/50$) Scale=0.10742
A1	Rev power	1	Full scale: VRMS = 103V (approx. 212W)	Vrms = 110N/1024 (V) (Power = $Vrms^2/50$) Scale=0.10059
A2	PA current	1	Full scale = 25A	Current = 25N/1024 (A) Scale=0.02441
A3	Temp	2	20mV/C; 1024 = 250C	Temp = 250N/1024 (C) Scale=0.2441

VSWR = (Vf+Vr)/(Vf-Vr) ignoring phase

The TX coupled path to RX1 input has 56dB attenuation (26dB in the Stockton bridge, and 30dB additional attenuation). Breakthrough not through the coupled path is approx. -30dB below that.

Information exchange from RP to Arduino

Data	Format	HPSDR packet	Pihpsdr sends?
RX 1 Frequency	16 bits, LSB=1KHz	C0=0x4	Y, case 2 (L809)
RX 2 Frequency	16 bits, LSB=1KHz	C0=0x6	Y, case 2 (L809)
TX Frequency	16 bits, LSB=1KHz	C0=0x2	Y, case 1 (L796)
RX OC settings	7 bits	C0=0x0	Y, case 0 (L710) *1
TX OC settings	7 bits	C0=0x0	Y, case 0 (L706) *1
TX attenuation	6 bits, LSB=0.5dB	C0=0x12	Y, case 3
TX mode RX att	5 bits, LSB=1dB	C0= 0x14	Y, case 6 *3
RX1 attenuation	8 bits, LSB=1dB	C0=0x14	Y, case 4 *2
RX2 attenuation	8 bits, LSB=1dB	C0=0x16	Y, case 5 *2
Antenna Select	5 bits: 0RRR00TT TX=TT bits: 00=1; 01=2; 10=3 RX = RRR bits: 000=1; 001=2; 010=3; 011=aux1; 100=aux2; 101=aux3	C0=0x0	Y, case 0 *4
Disable PA	1 bit; = 1 to disable TX	C0=0x12	Y, case 3
6m LNA	1 bit; = 1 to enable LNA	C0=0x12	Y, case 3

Notes:

*1: same data field, but TX settings valid when transmitting bit set (C0 bit 0)
*2: preamp and Alex 0/1/20/30dB attenuator settings not sent by pihpsdr, but could be factored in
*3: this is zeroed in pihpsdr at the moment, but must be there for a reason!
*4: pihpsdr swaps the bit for aux1 and aux2. I may need to swap them back in the Arduino.

Message	**Format**	**Meaning**	**HPSDR packet**
RX 1 Frequency	Byte 1: 0x1 Byte 2: Bottom 8 bits Byte 3: Top 8 bits		C0=0x4
RX 2 Frequency	Byte 1: 0x2 Byte 2: Bottom 8 bits Byte 3: Top 8 bits		C0=0x6
TX Frequency	Byte 1: 0x3 Byte 2: Bottom 8 bits Byte 3: Top 8 bits		C0=0x2
OC settings, Antenna select	Byte 1: 0x4 Byte 2: 7 bits TX OC (bit7=0) Byte 3: 7 bits RX OC (bit7=0) Byte 4: 0b0RRR00TT	RRR=RX; TT=TX TT bits: 00=1; 01=2; 10=3 RRR bits: 000=1; 001=2; 010=3; 011=aux1; 100=aux2; 101=aux3	C0=0x0
TX attenuation	Byte 1: 0x5 Byte 2: 6 bits TX att (LSB=0.5dB) Byte 3: 5 bits RX att when in TX Byte 4: 0b000000BA	B: disable PA TX (1=disable) A: enable 6m LNA (1=enable)	 C0=0x12 C0= 0x1C C0=0x12
RX attenuation	Byte 1: 0x6 Byte 2: 5 bits RX1 att (LSB=1dB) Byte 3: 5 bits RX2 att (LSB 1dB)		 C0=0x14 C0=0x16

RF Hardware Connector Pinouts

Antenna Switch Board

Serial Data Connector:

1	Serial Data	2	GND
3	Serial Clock	4	GND
5	Load	6	GND
7	Serial Data out	8	GND
9	TX enable	10	GND

LPF Connector:

1	LPF7 (50MHz)	2	GND
3	LPF6 (28 MHz)	4	GND
5	LPF5 (21 MHz)	6	GND
7	LPF4 (14 MHz)	8	GND
9	LPF3 (7MHz)	10	GND
11	LPF2 (3.5MHz)	12	GND
13	LPF1 (1.8MHz)	14	GND
15	+12V	16	+12V

BPF Connector:

1	BPF12 bypass	2	GND
3	BPF11 (50MHz + preamp)	4	GND
5	BPF10 (28MHz)	6	GND
7	BPF9 (24MHz)	8	GND
9	BPF8 (21MHz)	10	GND
11	BPF7 (18MHz)	12	GND
13	BPF6 (14MHz)	14	GND
15	BPF5 (10MHz)	16	GND
17	BPF4 (7 MHz)	18	GND
19	BPF3 (5 MHz)	20	GND
21	BPF2 (3.5 MHz)	22	GND
23	BPF1 (1.8 MHz)	24	GND
25	+12V	26	+12V

Note that a change of filter allocation needs (simple) code changes!

ADC/DAC Conditioning Board

Serial Data Connector:

1	NC	2	GND
3	Serial Clock	4	GND
5	Load	6	GND
7	Serial Data in	8	GND
9	NC	10	GND

Open Collector Output Connector:

1	OC 1	2	GND
3	OC 2	4	GND
5	OC 3	6	GND
7	OC 4	8	GND
9	OC 5	10	GND
11	OC 6	12	GND
13	OC 7	14	GND
15	OC 8	16	GND

Arduino / Red Pitaya Shield

A “shield” PCB sits between the red Pitaya and an Arduino Micro. Includes connections for Arduino I/O, provision for a cooling fan and 122.88MHz VCXO.

Connectors

CONN1: LCD 16W ribbon cable

(follows the standard for alphanumeric LCD displays)

1	GND	2	VCC
3	Vo (contrast)	4	RS
5	R/W~	6	E

7	D0	8	D1
9	D2	10	D3
11	D4	12	D5
13	D6	14	D7
15	Backlight_anode	16	Backlight_cathode

CONN2: Serial Data Connector 10W ribbon cable

1	Serial Data	2	GND
3	Serial Clock	4	GND
5	Load	6	GND
7	Serial Data out	8	GND
9	TX enable	10	GND

PL1: CODEC: 9 pin SIL

(designed for MIKROE506 CODEC evaluation board)

1	SCK
2	MISO
3	MOSI
4	ADCL
5	DACL
6	SDA
7	SCL
8	3V3
9	GND

PL2: Rotary encoder: 5 pin SIL strip

1	GND
2	+5V
3	Click
4	A
5	B

PL3: Arduino VSWR: 3 pin SIL strip

1	FWD (5V full scale reading = 212W RMS)
2	GND
3	REV (5V full scale reading = 212W RMS)

PL4: Munin PA: 6 pin SIL strip

1	Driver current
2	PA current (1V equivalent to 5A current)
3	Preamp current
4	No connection
5	PTT~ (Open collector; ground = TX)
6	GND

PL5: Temp sensor: 4 pin SIL strip

1	GND
2	+5V
3	(gap)
4	Temp input (10mV/C)

PL6: PTT out (for transverter): 2 pin SIL strip

1	GND
2	PTTreq~ (low if PTT requested by RedPitaya, not gated by Arduino)

PL7: PTT/Keyer input; 5 pin SIL

1	GND	
2	NC	
3	PTT~	OC input; 10K pullup. GND= activate PTT
4	DASH~	OC input; 10K pullup. GND= activate DASH
5	DOT~	OC input; 10K pullup. GND= activate DOT

PL8: RP VSWR: 3 pin SIL strip

1	FWD (3.5V full scale equivalent to 165W RMS)
2	GND
3	REV (3.5V full scale equivalent to 165W RMS)

PL9: Unused RP Analog I/O: 7 pin SIL (not loaded)

1	GND
2	Analog_In_2
3	Analog_In_3
4	Analog_Out_0
5	Analog_Out_1
6	Analog_Out_2
7	Analog_Out_3

PL10: Unused RP GPIO: 9 pin SIL (not loaded)

1	GND
2	DIO3_N
3	DIO1_P
4	DIO2_P
5	DIO3_P
6	DIO4_P
7	DIO5_P
8	DIO6_P
9	DIO7_P

PL12: Fan power: 2 pin SIL strip

1	GND
2	+5V

J1: USB

Provides +5v power input to shield.

Jumpers

JP1	Arduino Debug	If inserted, Arduino boots into DEBUG mode
JP2	USB Power	If inserted, USB power connector J1 connected to +5v (provided to allow power off easily)
JP3	Arduino Power	If inserted, Arduino takes power from Red Pitaya (normal use). Can be removed for debug through PC USB connection.