Hermes Lite J16 Filter Control Proposal

This document will outline the current J16 implementation in the HPSDR server for ghpsdr3-alex, and make some proposals that enhance the capability to control different hardware from a common code base.

Comments and code from ozy.c

Here are some snips of code from the recent commit of J16 support into master at ghpsdr3-alex.

/*
J16 pins
=====
17 out1 D bit 0
18 out2 C bit 1
19 out3 B bit 2
20 out4 A bit 3

e.g D switches 15m filter.

Band------------------------------J16 ---------------BPF/LPF DB25 Pins ---
1 ----> 39.85 - 64.4 MHz BC (6m) 1 2

2 ----> 23.2 - 39.85 MHz ABC (12m/10m) 4 2 1

3 ----> 19.6 - 23.2 MHz D (15m) 3

4 ----> 16.2 - 19.6 MHz A D (17m) 4 3

5 ----> 12.1 - 16.2 MHz B D (20m) 1 3

6 ----> 8.7 - 12.1 MHz AB D (30m) 4 1 3

7 ----> 6.2 - 8.7 MHz, CD (40m) 2 3

8 ----> 4.665 - 6.2 MHz A CD (60m) 4 2 3

9 ----> 2.75 - 4.665 MHz BCD (80m) 1 2 3

10 -- > 1.70 - 2.75 MHz ABCD (160m) 4 1 2 3

4. OC6 User open-collector output 7 (23)
5. OC5 User open-collector output 6 (22)
6. OC4 User open-collector output 5 (21)
7. OC3 User open-collector output 4 (20)
8. OC2 User open-collector output 3 (19)
9. OC1 User open-collector output 2 (18).
10. OC0 User open-collector output 1 (17)

*/

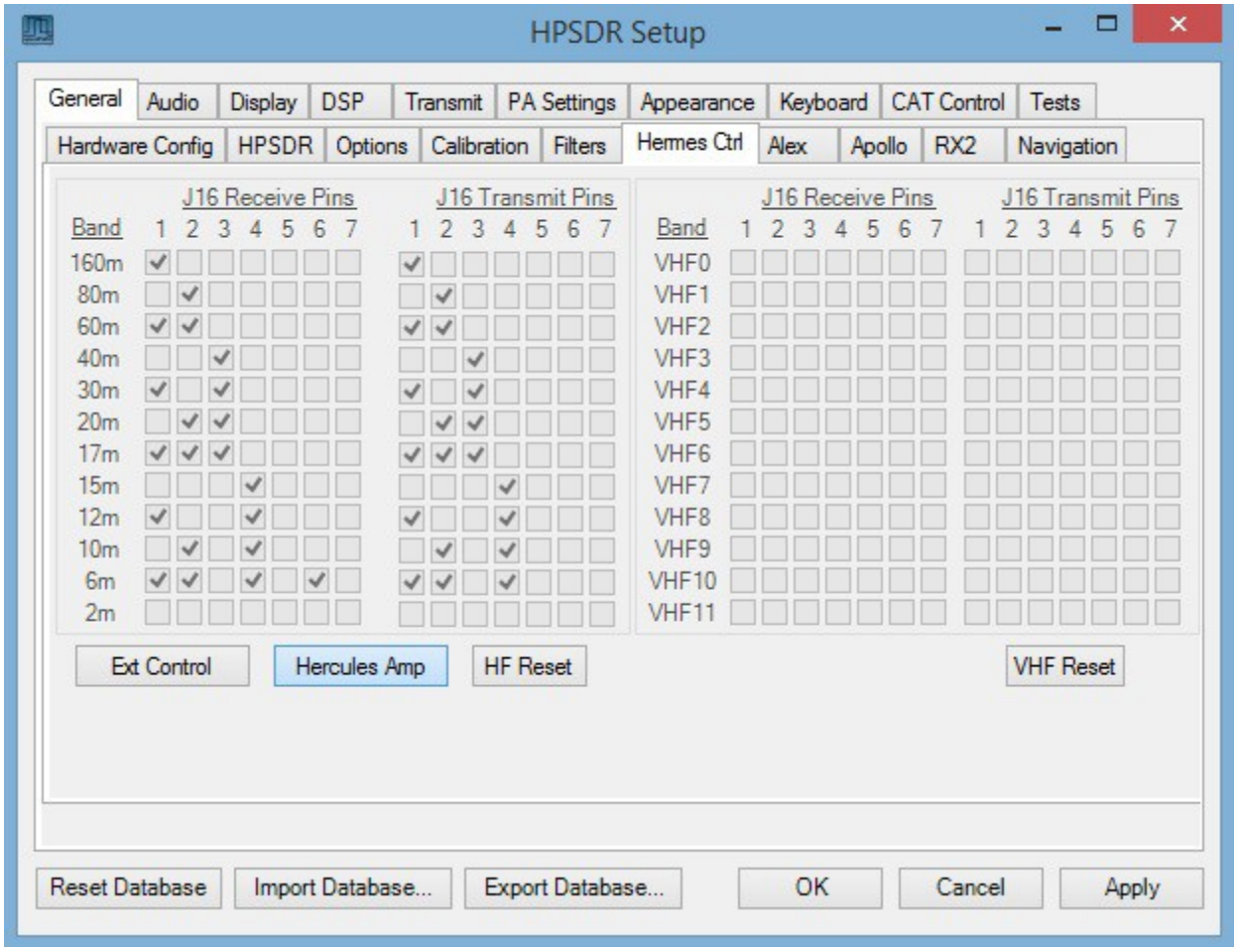
```
typedef struct _filter_j16 {
long f1;
long f2;
unsigned char j16;
} filter_j16;

filter_j16 fltj16_tbl [] =
{
{ 1700000, 2750000, 0x0f },
{ 2750000, 4665000, 0x07 },
{ 4665000, 6200000, 0x0b },
{ 6200000, 8700000, 0x03 },
{ 8700000, 12100000, 0x0d },
{ 12100000, 16200000, 0x05 },
{ 16200000, 19600000, 0x09 },
{ 19600000, 23200000, 0x01 },
{ 23200000, 39850000, 0x0e },
{ 39850000, 64400000, 0x06 },
};
#define ARRAY_SIZE(x) (sizeof(x)/sizeof(x[0]))

int get_j16_from_freq (long f)
{
int i;
for (i = 0; i < ARRAY_SIZE(fltj16_tbl); ++i) {
if ( (f >= fltj16_tbl[i].f1) && (f <=
fltj16_tbl[i].f2) )
return fltj16_tbl[i].j16;
}
return -1;
}
```

Note:

Andrea implemented a fixed array that is the same for both TX and RX. PowerSDR provides 2 tables of values in a configuration dialog that provides flexible control of filters for both RX and TX as shown here:



The first 2 Hermes Lite PA designs do not need this feature. However we should consider implementing this to remain functionally compatible with other radio control applications. Otherwise we risk causing QtRadio to be excluded from controlling some future design that implements unique different encoding values for RX vs. TX.

Megaband Filter Design Overview

Here is a filter design overview for Megaband, the 160M to 10M 5W Power amplifier.

The filters are driven from an open collector BCD decoder. We use signals USEROUT0 thru USEROUT3 for the decode. This corresponds to the OC0 thru OC3 signals on J16. The board does not latch the filter data as is done in the Alex protocol. J16 has the capability of providing different filter encodes for receive and transmit enabled by the state of PTT. This is not necessary for Megaband. We designed the logic to use the same encoding values for receive and transmit to avoid having relays chattering between TX and RX.

This would be the data needed for Megaband.

Code Band

0000 160
0001 80
0010 60
0011 40
0100 30
0101 20
0110 17
0111 15
1000 12
1001 10
1010 - 1111 = Bypass

There are 6 LPFs for TX. We wire-OR the open collector outputs of the BCD decoder to select the appropriate LPF thru a relay switch circuit. Presenting an invalid (bypass) BCD encode causes all of the relays banks to be off, resulting in a bypass in the 160M relay to be active.

There are 5 BPFs for RX. We wire-OR the open collector outputs of the BCD decoder to select the appropriate BPF thru a diode switch circuit. Presenting an invalid (bypass) BCD encode causes all of the filter banks to be off, resulting in a bypass circuit comprised of a diode OR array to activate a bypass.

Thus, the array needs to look like this after being loaded by whatever schema we determine. The frequency values are chosen to align roughly with the RX BPF filter passbands, with some arbitrary splitting to enforce groupings by individual bands. Note that the 10M frequency span is larger than the actual band. This is to allow the use of a transverter for 2M to 10M to be used without causing a filter issue within mid-band.

```
filter_j16 fltj16_tbl [] =
{
{ 1700000, 2750000, 0x00 },
{ 2750000, 4665000, 0x01 },
{ 4665000, 6500000, 0x02 },
{ 6500000, 8700000, 0x03 },
{ 8700000, 12100000, 0x04 },
{ 12100000, 17500000, 0x05 },
{ 17500000, 19600000, 0x06 },
{ 19600000, 23200000, 0x07 },
{ 23200000, 27500000, 0x08 },
{ 27500000, 32000000, 0x09 },
};
```

Superband Filter Design Overview

Here is a filter design overview for Superband, the 5W Power amplifier that provides user select-able

BPFs to build at user discretion.

The BPFs cover 160, 80/75, 60/40, 30/20, 17/15 and 12/10 and the BPFs are shared for both RX and TX. These can be built and grouped in any combination, however the most likely groupings will be 160 with 80/75, 60/40 with 30/20 and 17/15 with 12/10. Those will be the groupings that I will document here.

The filters are driven from a single FPGA signal called USEROUT0. There is no open collector decoder provided since the board attaches directly into the PCI-E connector on Hermes Lite. The board does not latch the filter data as is done in the Alex protocol. J16 has the capability of providing different filter encodes for receive and transmit enabled by the state of PTT. This is not necessary for Superband. We designed the logic to use the same encoding values for receive and transmit.

This would be the data needed for Superband. Bank denotes the filter group as defined on the Superband schematic.

Code	Band	Bank #
0000	160	0
0001	80	1
0000	60	0
0001	40	1
0000	30	0
0001	20	1
0000	17	0
0001	15	1
0000	12	0
0001	10	1

1010 - 1111 = none provided, will follow the state of the least significant bit.

Thus, the array needs to look like this after being loaded by whatever schema we determine.

```
filter_j16 fltj16_tbl [] =
{
{ 1700000, 2750000, 0x00 },
{ 2750000, 4665000, 0x01 },
{ 4665000, 6500000, 0x00 },
{ 6500000, 8700000, 0x01 },
{ 8700000, 12100000, 0x00 },
{ 12100000, 17500000, 0x01 },
{ 17500000, 19600000, 0x00 },
{ 19600000, 23200000, 0x01 },
{ 23200000, 27500000, 0x00 },
{ 27500000, 32000000, 0x01 },
};
```