In order to get the Burst to trigger off of the ac test signal you must:

Set Weights and Thresholds…

If you want to see what bin and what value you are getting for a certain ac test signal, you can turn on that signal (in appendix below) and then dump table 5, offset 0 using the command efw\_dumptabl(table=5, offset=0)

Looking at the data in the dumped table- you can make sure your signal is in the correct bin (assuming you have set the filter banks to have 13 bins using the scr.dfb\_std\_config (table=7) given by the following:

13 bins: [0.8-1.5, 1.5-3, 3-6, 6-12, 12-25, 25-50, 50-100, 100-200, 200-400, 400-800, 800-1.6k, 1.6k-3.2k, 3.2k-6.5k]

So 128Hz would be bin 7, 2kHz bin 11, 6kHz bin 12 (start counting bins with zero)

Therefore you can turn on a ac test signal of 128Hz as follows:

cmd.EFW\_ACTESTHIGH(ACTEST2=0, ACTEST1=1, HHHH=0xF)

cmd.EFW\_ACTESTLOW(LSB=0xFF)

And then dump table 8, and looking at the dumped data you should see:

00 00 00 00 00 00 xx 00 00…etc

where xx is the value just below which you should set your threshold. This value will differ depending on how many booms you have hooked up.

So now you want to load certain weights and thresholds into table 1.

NB- the weight is a scaling factor, and the threshold is an actual value. For testing the burst functionality, we have used a scaling factor of one (ff, or full scale).

To do so you must write these values into a table, specifically table 1 (as seen below)

#setting the load address to table 1, weight0, bin 7

 cmd.EFW\_ADRH(0x96)

 cmd.EFW\_ADRL(0x17)

(we figured 96 from dumping table one, offset zero, and the dumped table address was 9600… we got 17 by taking 16 (offset of weight one) and adding 7 (the bin we want) and converting that sum into hex

So 23 (16+7) in decimal becomes 17 in hex. You can see that you have the right address by lookin in the housekeeping apid (267) and seeing a load address of 0x9617.

Then to load the value of ff you send the command:

cmd.EFW\_LOAD(0xFF)

The same is repeated for each weight and threshold (1 and 2) for the different desired values. As seen in this script (there are some waits in there as well)

See commands sent in the script we wrote below:

 #set telem rate to 1 sec update for apid 267

 cmd.efw\_tmpktrate(4,1)

 sleep(5)

 #setting the load address to table 1, weight0, bin 7

 cmd.EFW\_ADRH(0x96)

 cmd.EFW\_ADRL(0x17)

 #waiting until this reads back in apid 267

 wait(APID\_267, lambda: APID\_267.LOADADR==0x9617, flTimeout=10)

 #loading a weight of 1 into the table

 cmd.EFW\_LOAD(0xFF)

 #set the threshold to 0x36, this might change when all booms are hooked up

 cmd.EFW\_ADRH(0x96)

 cmd.EFW\_ADRL(0x57)

 wait(APID\_267, lambda: APID\_267.LOADADR==0x9657, flTimeout=10)

 cmd.EFW\_LOAD(0x20)

 #same for weight1, thresh1

 cmd.EFW\_ADRH(0x96)

 cmd.EFW\_ADRL(0x9B)

 wait(APID\_267, lambda: APID\_267.LOADADR==0x969B, flTimeout=10)

 cmd.EFW\_LOAD(0xFF)

 cmd.EFW\_ADRH(0x96)

 cmd.EFW\_ADRL(0xDB)

 wait(APID\_267, lambda: APID\_267.LOADADR==0x96DB, flTimeout=10)

 cmd.EFW\_LOAD(0x20)

 #same for weight2, thresh2

 cmd.EFW\_ADRH(0x97)

 cmd.EFW\_ADRL(0x1C)

 wait(APID\_267, lambda: APID\_267.LOADADR==0x971C, flTimeout=10)

 cmd.EFW\_LOAD(0xFF)

 cmd.EFW\_ADRH(0x97)

 cmd.EFW\_ADRL(0x5C)

 wait(APID\_267, lambda: APID\_267.LOADADR==0x975C, flTimeout=10)

 cmd.EFW\_LOAD(0x20)

The next step is to run the AC functional script. This turns on the AC test signal at the three frequencies (128Hz, 2kHz, and 6kHz) for one second each. The script dumps table 8 ( the burst data) and reconfigures the burst data so you can see the three values being collected- something like

34 36 44 00 00 00 etc…

Burst 1 Info

The burst 1 must also be set to trigger off the ac functional. This is done in the same way as setting weights and thresholds for burst 2. We have done this in the AC\_functional script by:

Calling the weight\_thresh script, which loads values into weight0, thresh0 (0-3)

You must also enable burst collection and playback.

We then load a chosen weight/thresh combination into Table 9, offset 0, which is the ‘Burst 1 Trigger Function’ with the following commands:

cmd.EFW\_TABLE(TABLE=9, OFFSET=0)

cmd.EFW\_LOAD(0)

You can then turn on the AC test signal, and you should see the burst 1 pointer move forwards.

Once it stops moving, you can playback the burst 1 with one of two commands from the util script, SET\_B1PLAYBACK, or QUEUE\_B1PLAYBACK

\*\*note- when using a command from the util script, you must first import util. Also, capitalization matters! So if its in all caps in the util script, you must use all caps when entering the command, (see below)

From the util script:

def SET\_B1PLAYBACK( block, leng ):

 cmd.EFW\_B1\_SETPLAYPTRL( block & 0xff )

 cmd.EFW\_B1\_SETPLAYPTRH((block >> 8 )& 0xff )

 cmd.EFW\_B1\_SETPLAYPTRE((block >> 16)& 0xff )

 cmd.EFW\_B1\_SETPLAYCTRL( leng & 0xff )

 cmd.EFW\_B1\_SETPLAYCTRH((leng >> 8 )& 0xff )

NOTE: for queuing, it gets rid of the last two bytes so you can only set it for every 4 bytes… ie, if you want to start at an address that ends in …63, you will either have to start at 60, or 64 (and lose that first packet) the playback can start at any address whether a multiple of four or not.

def QUEUE\_B1PLAYBACK( block, leng ):

 cmd.EFW\_B1\_SETREQDTA( (block >> 2) & 0xff )

 cmd.EFW\_B1\_SETREQDTA( (block >> 10)& 0xff )

 cmd.EFW\_B1\_SETREQDTA( leng & 0xff )

 cmd.EFW\_B1\_SETREQDTA( (leng >> 8 ) & 0xff )

Below find copies of some of the tables referenced above, as well as some notes from John Bonnell.

RBSP-EFW

Burst System Operation and Trigger Function Testing

**Interface and Commanding Basics:**

The parametric interface to the RBSP-EFW burst control system is implemented as a set of FSW tables [ref 1, tab TABLES].

Trigger Functions are defined by parameters stored in FSW Table 1 (Triggering Parameters):



For the WEIGHTn and THRESHn, one has that the byte offset into Table 1 is:

OFFSET = 16 + (n-1)\*128 + (0, if WEIGHT; 64, if THRESH)

+ (m-3)\*26 + (0, if AVG; 13 if PEAK) + i,

where the fixed 16-byte offset comes from the table header, n is the function index (n=1..6), m is the FB\_INT index (m=3,4), and i is the frequency bin index (i=0..12).

For example, to set the WEIGHT and THRESHOLD as desired for one of the setups described later in this document, (n,m,i) = (2,3,10), and so the offsets are:

 WEIGHT = 16 + 128 + 0 + 0 + 0 + 10 = 154 (0x9A)

 THRESH = 16 + 128 + 64 + 0 + 0 + 10 = 218 (0xDA)

The WEIGHTn and THRESHn tables are related to the on-board data in Table 5 via the following mappings [ref 2, tables 2 and 1]:

|  |  |  |
| --- | --- | --- |
| Function Number | Name | Formula |
| 0 | Sum1 | Sum( w0i\*TBLi\*H( TBLi – THRESH0i)) |
| 1 | Sum2 | Max( w1i\*TBLi\*H( TBLi – THRESH1i)) |
| 2 | Sum3 | Sum( w2i\*TBLi\*H( TBLi – THRESH2i)) |
| 3 | Max1 | Max( w3i\*TBLi\*H( TBLi – THRESH3i)) |
| 4 | Max2 | Max( w4i\*TBLi\*H( TBLi – THRESH4i)) |
| 5 | Max3 | Max( w5i\*TBLi\*H( TBLi – THRESH5i)) |
| 6 | Test1 | Returns TestValue LSB (located at 0xDDF9) |
| 7 | Test2 | Returns TestValue MSB (located at 0xDDFA) |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Indices | Description | Source | Comments |
| 0..12 | FB\_INT3\_avg[ 0..12] | DFB (APID 0x242) | Hearts |
| 13..25 | FB\_INT3\_pk[0..12] | DFB (APID 0x242) | Clubs |
| 26..38 | FB\_INT4\_avg[0..12] | DFB (APID 0x242) | Diamonds |
| 39..51 | FB\_INT4\_pk[0..12] | DFB (APID 0x242) | Spades |
| 52 | EQV\_ECT | FSW | Derived from ECT Shared Data in APID 0x100. |
| 53 | EQV\_EMFISIS | FSW | Derived from EMFISIS Shared Data in APID 0x100. |
| 54 | EQV\_RBSPICE | FSW | Derived from EMFISIS Shared Data in APID 0x100. |
| 55 | EFW\_CCC | FSW | Derived from setting of internal EFW Conjunction and Campaign Control flag. |
| 56..63 | SPARE | TBD | For expansion. |

The raw on-board data used by the trigger functions is stored in FSW Table 5 (Trigger Function Data Table (SYS\_016)):



How the burst system uses the trigger functions to select bursts for recording and playback is controlled by parameters stored in FSW Table 9 (SSR Control and Information):



The valuations of the Burst-2 (B2) burst segments are stored in FSW Table 8 (Burst Values):



The Burst-1 (B1) playback request table is stored in FSW Table 10 (Burst1 Playback Request Table):



The FSW interface for writing to and reading from FSW tables are the two commands, TABLE and DUMPTABL [ref 1, tab CMDS], used to set the location of the write (load) or read out (dump) operation:





In addition, the FSW LOAD, TMDUMPSIZE, and TMDUMPCTR commands are used to actually write (LOAD) or readout (TMDUMPSIZE, TMDUMPCTR) the table data:





Some elements of these tables are routinely included as part of the EFW HSK output data in the following APIDs and GSEOS screens:

(supporting graphics to be included later).

**Testing With ACTEST:**

In the stowed configuration, a square wave can be applied to the ODD or EVEN EFW sensor inputs. This ACTEST signal has a frequency that is given f [Hz] = (524288 Hz)/N, where N is a divide-down factor.

Both the selection of which EFW sensors to excite (ODD, EVEN, or both) and the divide-down factor are set using the FSW ACTESTLOW and ACTESTHIGH commands:



For example- this set of commands gives you a 6kHz ac test signal:

 cmd.EFW\_ACTESTHIGH(ACTEST2=0, ACTEST1=1, HHHH=0x00)

 cmd.EFW\_ACTESTLOW(LSB=0x57)

(106 gives you a 2kHz signal, and FFF gives you a 128Hz signal)

The amplitude of the excitation signal depends upon how many ACTEST lines are active, and how many boom units are connected to the signal, as the ACTEST signal has only a single output buffer on the DCB, which then sees all the connected loads (EFW sensor input impedances and stowed sensor impedances to ground) in parallel.

An example of the FB output (APID 0x241) during a Flight CPT is shown below, in which 2 SPB and 1 AXB are excited at ACTEST frequencies of 128 Hz, 2 kHz and 6 kHz:



From these results, one can see that the ACTEST excitation will provide for a significant (>100 count) response at the peak of the response spectrum, with some significant response at higher frequencies due to the harmonic content of the square wave input.

The nominal frequency ranges corresponding to the 7- and 13-bin FB data products are as follows:



There are two separate filter bank channels, each with a peak and average component for each frequency bin, giving a total of 52 filter bank related parameters in each function. The addition of the parameters for the weighting of external factors, as well as spares, brings the number of parameters up to 64 for each of the 6 non-test trigger functions.

For the initial test of B2 triggering, we will use function 1, SUM2, with both weights and thresholds keyed to the 800-1.6k band (band 10), AVG response of FB\_INT3 (EDC12?). We will also set the EFW configuration bytes using CONFIG0 and CONFIG1:

 CONFIG0, 0x00 (default BEB config).

 CONFIG1, 0xF0 (test configuration 0).

DO we need to load the Table with zeros first, or is that guaranteed by FSW?

Using the offset formula, set the WEIGHT and THRESH to 1 (0x01) and 50 (0x32) respectively for the bin in question.

Enable B2 recording.

Run the ACTEST up in frequency from 128 Hz up to several kHz in some number of steps, with a dwell time of at least N seconds, verifying using MEMDUMP that the computed trigger function value is as expected from the parameters.

Disable B2 recording.

Dump the B2 burst value table and verify that the range of values in it make sense.

Enable B2 playback and verify that the bursts playback in the correct order, and that the B2 table is updated appropriately (value of played back burst goes to zero?).

**References:**

[1] RBSP\_EFW\_FSW\_005\_CTM.xls

[2] RBSP\_EFW\_SYS\_016B.doc

Revision History:

* Rev A, JWB, UCB SSL, 4 April 2011.
	+ initial revision, trying to pull together all the info and references into one place for ease of use, as well as to define the initial attempt at the test.