Tech Note #316 FERA Readout and a FERA Based Level 1.2 Trigger

John S. Haggerty
Physics Department
Brookhaven National Laboratory, Upton, New York 11973

July 18, 1996

Abstract

During the 1996 run of 787, readout of the FERA ADC's migrated from the venerable old BFI module to a Struck 370 FASTBUS DSP. Using this module to diagnose problems, problems with corrupted FERA data have been largely eliminated.

A Level 1.2 trigger was developed using the DSP to perform a calculation on the range stack FERA data. The implementation of this trigger is described, and its performance is evaluated.

Contents

1 Introduction

Data from the FERA ADC's are read out by a 16 bit ECL serial bus which connects the seven crates of FERA's together. Until the 1996 run, data were pushed into a wire-wrap FASTBUS module designed by Mark Ito and Maged Atiya called the BFI, which formatted the FERA data so that the logical address of an ADC channel appeared in the upper half of the 32 bit data word, and the ADC value in the lower. This module performed valiantly since 1988, but it had become increasingly difficult to find and fix problems with the FERA system, and it was desirable to replace it with a module that would be easier to diagnose. During the 1995 run, there was an increasing incidence of duplicate and corrupt addresses observed in the FERA system, and one of the goals of replacing it was to reduce or eliminate them. Also, there was a desire to make a trigger which could reduce the amount of data and deadtime, and calculations on the FERA data as was done in the old Level 1.5 was thought to be a good candidate. Using a processor would allow some flexibility in making such a trigger, if it could be executed fast enough.

2 Deadtime with Level 1.n

The 787 trigger allows up to 4 different Level 1.n's for eight group A triggers. The Level 1.n's and the triggers are combined in the and/or board, which receives done and return code signals from each of the 4 Level 1.n's, and decides whether to apply the condition to the current trigger. If the Level 1.n fails, the event is failed as soon as the first Level 1.n reports done and a failed return code. If the Level 1.n passes, the trigger results in an SSP start (beginning data acquisition) only after all the Level 1.n done's have been received.

The deadtime with no Level 1.n is given by $n \cdot t_{readout}$ where n is the number of events read, and $t_{readout}$ is the readout deadtime per event. The deadtime with a Level 1.n is given by

$$\frac{n}{r}(t_{readout} + t_{1.n}) + n(1 - \frac{1}{r})t_{1.n} \tag{1}$$

where r is the rejection and $t_{1.n}$ is the execution time of Level 1.n. The first piece is the readout deadtime of the events which pass Level 1.n, increased by the additional time it takes to execute Level 1.n even if the event passes. The second piece is the time it takes to execute Level 1.n on rejected events. Then the ratio of the deadtime with a Level 1.n to that without is given by

$$\frac{1}{r} + f \tag{2}$$

where r is again the rejection, and f is the ratio of the Level 1.n execution time and the readout deadtime, $t_{1.n}/t_{readout}$.

This sets the necessary level of performance for a Level 1.n: a rejection of 2 which takes 50% of the readout deadtime to execute has no effect at all on total deadtime. (Of course, there are half as many events to be read out, which saves tape and may allow more events to be buffered in the SSP's, but without a reduction in deadtime, one cannot handle more incident intensity unless a bottleneck at transferring or recording events causes spills or parts of spills to be missed.)

3 DSP Readout of the FERA System

In the 1996 run, the Struck 370 FASTBUS based DSP board[1] was used for reading the FERA data. This board has up to four Motorola 96002 DSP's,[2] each with a 4096 word FIFO on the front end which receives the serial data from the FERA system. The write strobes (WSO's) from each FERA crate are or'ed to make an over all write strobe input to the DSP.

The most efficient way to copy data from the input FIFO of the DSP to memory readable by FASTBUS is to begin only when the word count is final. In order to accomplish this, the request line of the FERA system (RQO) is used to generate an interrupt to the DSP. Request is a signal which is asserted when the FERA data is pouring out data on the ECL bus; it is manipulated externally to make a TTL pulse when all the data have been written into the FIFO, which is used as IRQ1 (Interrupt Request 1) of the DSP.

The program running in the DSP does nothing but wait for interrupts. IRQ1 indicates that data are present in the FIFO, and IRQ0 (the higher priority interrupt) indicates that a clear sequence must take place. The basic readout program simply copies data from the FIFO to a fixed place in memory (0xe000, which is accessible from FASTBUS at secondary address

0x40006000). In this loop over words, the address lookup is performed by forming an address from the Virtual Station Number (or VSN, found in the module's header word) and the subaddress (SA, found in each word's data).

Mark Ito pointed out a complication which occurs when Level 1.2 rejects an event, and a clear sequence is initiated. The clear results in ending the FERA readout, so RQO returns low, which would make a data interrupt (IRQ1) to the DSP, competing with the clear interrupt (IRQ0) which would come at almost the same time. External NIM logic was added to prevent this situation, by using a long clear pulse (1600 ns) which was used to veto the generation of IRQ1.

4 Getting the Stopping Channel into the FERA Data

Most Level 1.n triggers require information beyond the FERA data in order to make a decision. In order to allow for this, a board was developed which allows us to insert data onto the FERA serial bus at a time determined by a strobe input; Figure 1 show the schematic of the board used to get these data to the DSP. This is possible because there is a 12 μs delay from the time the FERA's are gated until any data appears on the ECL bus which allows all FERA modules to convert before readout commences. (This delay actually takes place in the last, or "system" FERA driver 4301 module.) Strobing the stop channel board in this period puts data from the 16 inputs of the stopping channel bus on the output ECL bus, and generates a NIM Write Strobe which can be used to strobe the data into the FIFO. In principle, one can strobe many words of data into the FIFO; in practice, a single Delay-Width-Fanout (DWF) channel was used to generate a strobe to this board before any real FERA data could be written out.

The stop channel data that was sent to the DSP came from two sources. The stopping layer bits came from the trigger transmitter (TT) board (and were fanned out in order to service the Level 1.1 and Level 1.2 triggers). These bits are "backward," in that they encode 22-stop_layer. They can be directly compared to the bits from the Trigger Transmitter register 0xc0000000 bits 10:14 (read into the TRI2 bank as the first word from module id 0x7875).

The stop sector data came from Jesse Stone's Stopping Channel Finder

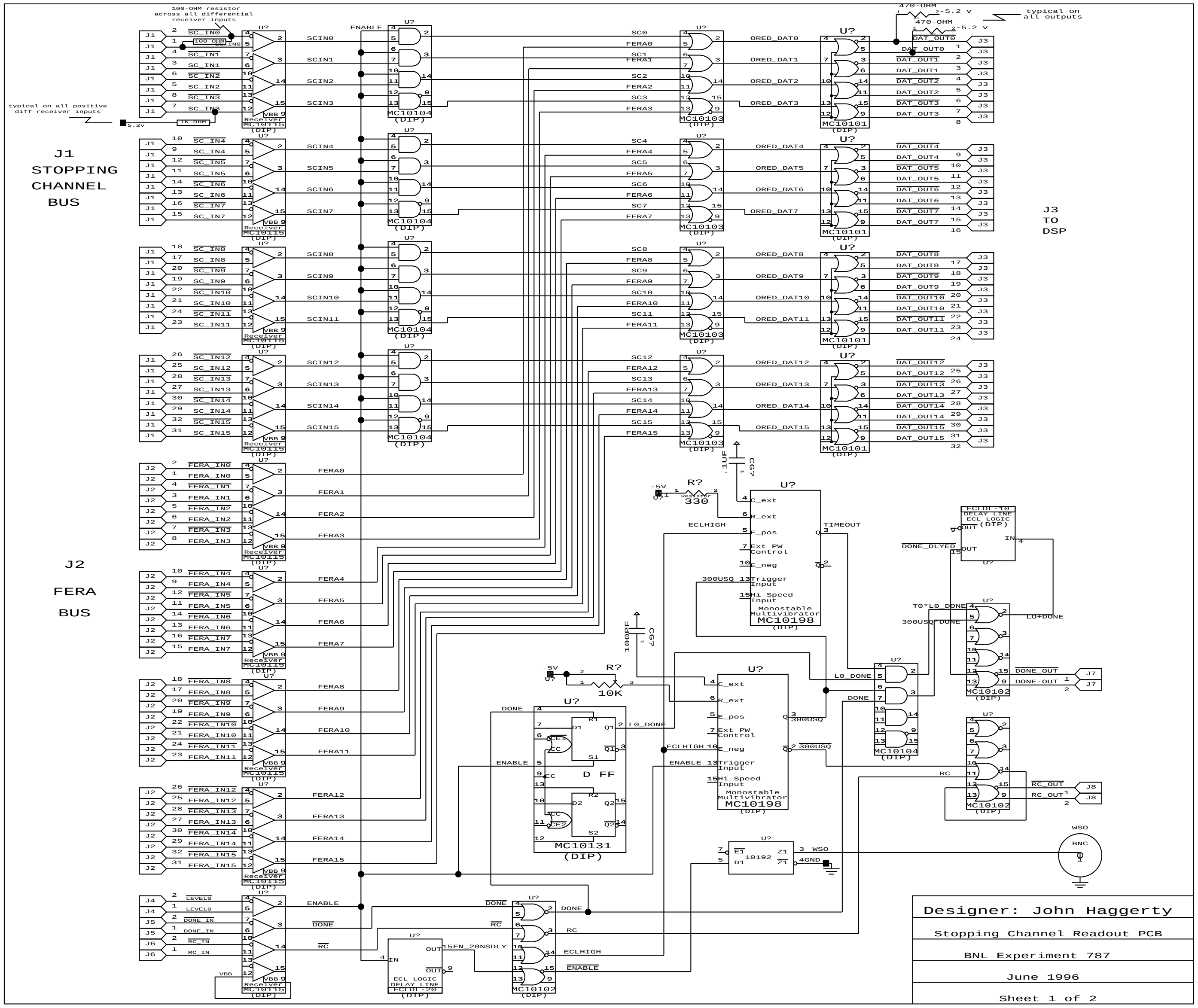

Figure 1: Schematic of the stopping channel readout board. This schematic was drawn by Marty Van Lith.

Bits	Origin	Meaning
0:2 (3 bits)	SCF	Stopping hextant (1-6)
3:4 (2 bits)	SCF	Stopping sector within stopping hextant (0-3)
5:9 (5 bits)	TT	22-stop_layer

Table 1: Bits sent to DSP FIFO.

board but have been rearranged slightly to make a single 5 bit stop sector. They can be compared to the SCF word read by the first I/O board in the TRIG bank.

The stop channel board also took the Level 1.2 done signal as an input, and issued an output done if either the input done toggled, or the timeout period expired. The timeout could be adjusted with a front panel potentiometer to make certain that the Level 1.2 done came while it was still possible to clear the 1876 TDC's. [After the 1876 TDC's are stopped, the event can be cleared until either a pulse appears on the Fast Clear Window (FCW) input, or an internal timeout occurs, at a maximum time of about 500 μs. After that, the data are transferred to the internal multi-event buffer, and can't be cleared, except by reading the data.] The return code of the Level 1.2 trigger is made to pass the event by default, so if no trigger decision has been made before the timeout, the event passes. This, of course, tends to reduce the rejection of the trigger.

5 Level 1.2

5.1 Level 1.2 Algorithm

The Level 1.2 algorithm has evolved from work done by Masa Aoki and Akira Konaka in 1995, and by Peter Meyers in 1996. This algorithm has been modified somewhat for speed and implementation in the 96002.

The algorithm is done mainly with lookup tables. The only range stack layers involved are the stopping layer, and the layer after the stopping layer. The "layer lookup" takes the 5 bit stopping layer and the 5 bit layer number from the FERA channel address, forms an offset address into the lookup

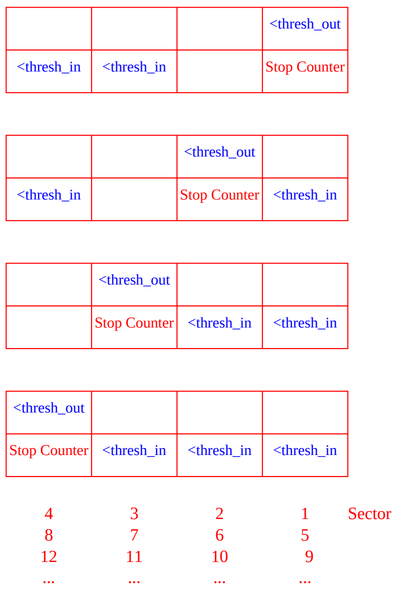

Figure 2: This figure shows which counters within a hextant are required to be "quiet" to pass the event. The threshold in the stop layer was 16 ADC counts, and 32 counts in the layer behind the stop layer. The counters indicated must be below threshold for the event to be passed.

table, and does nothing if the looked up value is zero. Similarly, the 5 bit stopping sector and the 5 bit sector number in the RS address part of the data word are used to look up the threshold value to reject the event. The bits are ordered so that very little bit manipulation of the FERA address word is needed. Very little calculation is added, except to words in the stopping layer or the layer behind the stopping layer. Figure 2 shows which counters the cuts were applied on. Events were rejected as soon as one of the counters exceeded the threshold.

5.2 Performance

Figure 3 shows the performance attained by Level 1.2 in the DSP. As a bonus, the true execution time of the Level 1.1 trigger has been measured to be about 14 μs, close to the estimation made by eyeing the scope. This version of Level 1.2 is the most highly evolved, which could issue a done as soon as the loop over data encountered a data word which would cause it to reject the event. With the timeout circuit, the rejection was about 1.55, and the fraction of the readout time (f) was about 0.19, for a ratio of deadtimes of about 0.83.

There was a problem with this trigger when it was run in combination with the Level 1.1 trigger which remains unsolved, and which prevents it from being used in normal data taking, even if the execution time can be improved. Individually, each Level 1.n allowed the trigger to cycle and events to be acquired without error. With both of them capable of cutting events, errors were encountered with either the FERA readout or the FASTBUS ADC's and TDC's. These problems are not understood at the time of writing and it is not even clear whether the problem is with the trigger or the particular done and return code signals that are sent to the trigger. The problem is not that rare, however, and solving it may even illuminate the cause of some very similar, but much rarer, errors which occur even without Level 1.2.

6 Offline Level 1.2

An offline version of Level 1.2 was developed which operated directly on the FERA data, and could be used to compare with the results calculated online. For this purpose, a bank (L12B) was produced by the DSP with information about which ADC word failed the event, and some information about how the stop channel lookup was done. The offline simulator agrees perfectly with the online version of the Level 1.2. The lookup table which is loaded into the dsp is used offline in order to keep the two versions as close as possible to one another. Programs to check the online Level 1.2 are in /e787/local/online/fastbus/dspinit.lv12/lv12_offline.

7 Software

7.1 DSP Compiler

The 96002 cross assembler runs on a Sun. It is installed on bnlku9x, with /usr/local/dsp/bin needed in the search path. The distribution includes the (gnu) C compiler, although I have programmed exclusively in 96000 assembler (necessary for some low level code, and desirable for speed in any case).

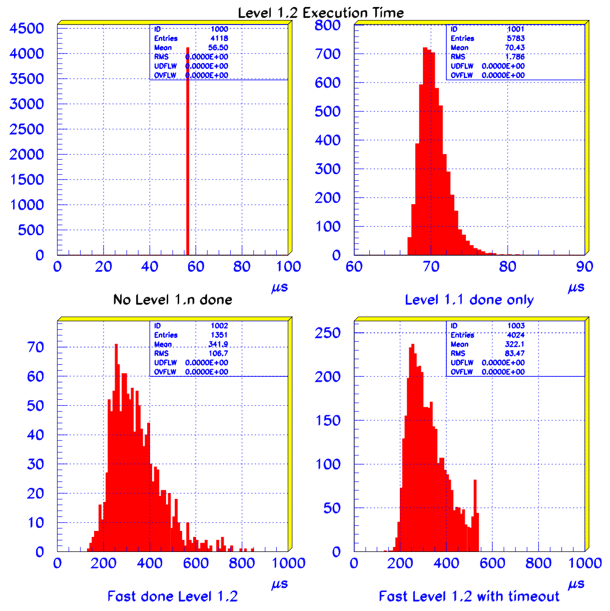

Figure 3: Execution time of Level 1.2. Top left: With no Level 1.n, there is a fixed time of 56.5 μs from Level 0 to the first read of an I/O board clock. Top right: Level 1.1 alone takes about 14 μs more. Lower left: Level 1.2 takes about 300 μs to execute. Lower right: Level 1.2 must be forced to conclude before the 1876 TDC's can no longer be cleared; this shows the effect of the timeout circuit in the stop channel board.

Below is an example Makefile which takes a source file (fera.asm) and produces a file ready to download via FASTBUS (fera.dmp).

```
.SUFFIXES: .asm .cln
PROG1 = fera
OBJ1 = fera.cln

all:    $(PROG1)

$(PROG1):       $(OBJ1)
        dsplnk -b$(PROG1).cld -m$(PROG1).map -v $(OBJ1)
        cofdmp -s -d$(PROG1).dmp $(PROG1).cld

.asm.cln:
        asm96000 -b$*.cln -l$*.lst -v $<

clean:
        rm -f *.cln *.lst *.cld *.map *.dmp
```

7.2 FASTBUS Loader

The FASTBUS program loader is in $DSPINIT_PROGRAMS, which is derived from a loader developed by Tino Haupke at Struck. The default version also loads the address map for the FERA's, which it derives from the file pointed at by the environment variable $FERA_ADD. (This file contains the correspondence between FERA VSN+SA and "logical" address.)

The Level 1.2 trigger requires loading a lookup table into the DSP which tells which counters are examined by Level 1.2. This is also done by dspinit, in the version in /e787/local/online/fastbus/dspinit.lv12.

7.3 DSP Program

The essential part of the loop over data words in the FIFO is shown below. Of course, there is lots of initialization and interrupt handling not shown, but this is the essential part of the code into which was inserted the Level 1.2 trigger algorithm. The code itself is in /home/bnlku4/haggerty/dsp/fera on bnlku9x, although local copies of the executable image (.dmp) are kept in $DSPINIT_DATA.

```
        DO      X:(R2),_INP_RLOOP

        MOVE    Y:A_INPFIFO,D4.L        ; read FIFO

        BTST    #15,D4.L                ; module header if bit 15 set
        BCC     _DATA_WORD

        MOVE    #VSN_MASK,D5.L          ; module header word
        AND     D4.L,D5.L               ; D5.L is now the latest VSN
        LSL     #4,D5.L
        MOVE    #MAP,D6.L               ; form base address for this vsn
        ADD     D6.L,D5.L
        NOP
        MOVE    D5.L,R3                 ; R2 is base displacement for this vsn
        BRA     _STORE_RAW_DATA

_DATA_WORD
        MOVE    #SA_MASK,D5.L           ; data word
        AND     D4.L,D5.L
        LSR     #11,D5.L
        MOVE    D5.L,N3
        NOP
        MOVE    X:(R3+N3),D5.L          ; lookup address for this vsn+sa
        OR      D5.L,D4.L

_STORE_RAW_DATA
        MOVE    D4.L,X:(R1)+            ; store in RAW_DATA

_INP_RLOOP
```

7.4 FERA Pedestal Run

The FERA pedestal run, feraped, has been modified to read data from the DSP. This program now works directly on SPRA bank, which is the rawest form of FERA data. The FERA pedestal run also checks the format of the data, using components of the dsp_check program described below. Since the stop channel board adds an extra word of data, there is a very slightly modified version of feraped which takes this word into account when Level 1.2 is in use.

7.5 Verifying the Structure of FERA Data

An analysis program was developed which checked that the FERA data was formed currectly. The program is in $DSPINIT_PROGRAMS/dsp_check, and there are slightly different version depending on whether the stopping channel word is present or not. In a check of 48,425 events, there were 525 "errors" in the data at the end of the run, all of them with either FERA address 0x2115 (crate 14, module 3, channel 13) or 0x204c (crate 14, module 10, channel 14) which reported zero ADC data, even though the channel should have been suppressed if the ADC is really zero. Several other modules were replaced with this disease, but since it occurred at a low level, and it is very hard to see this effect (except in normal data acquisition), the modules were not replaced.

7.6 FERA Data Format

Below is the working document which summarizes the FERA data format as implemented in the DSP. The raw data bank, SPRA, is produced in the DSP and read without modification by the FERA SSP program. The PA banks are produced by the dr's on the way through the SGI to tape by code written by Stephen Adler.

```
FERA Data Format Proposal
John Haggerty, BNL, Feb 20, 1996/March 11, 1996

March 11, 1996: Changed bank name to SPRA so as not to conflict with
FERA bank of raw data from BFI.  Added event number and some status
information to the SPRA bank.  Proposed changing the pedestal margin
from 2 (has been 4).

March 20, 1996: Made sure that the inclusive word count from DSP n
```

is 1 if there are no FERA data in that DSP.

The FERA data comes into the FIFO like this:

```
1 1 1 1 1 1
5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
-------------------------------
1 --XWC-- 0 0 0 ------VSN------
0 ---SA-- -------DATA----------
0 ---SA-- -------DATA----------
...
0 ---SA-- -------DATA----------
1 --XWC-- 0 0 0 ------VSN------
0 ---SA-- -------DATA----------
0 ---SA-- -------DATA----------
...
0 ---SA-- -------DATA----------
```

where XWC=0 to 15, 0 means 16, and the upper half-word is packed with 0.

[Note that one can make the DSP pack the words densely using MODE32
instead of MODE16, but this seems to me to have no advantage in
packing or unpacking for us; one saves in word count, but the FERA
word count is not yet an issue.]

I propose to make data that looks like this:

```
3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1   1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6   5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
-----------------------------------------------------------------
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   1 --XWC-- 0 0 0 -----VSN------- (VSN word)
--SYS-- -----Logical Address---   0 ---SA-- -------DATA----------
```

Note that now the *PA banks made in the BFI and SSP have the format:

```
0 0 0 0 ---Logical Address----   0 0 0 0 0  -------DATA----------
```

although the upper four bits of logical address (SYS) were actually present in the BFI, and used to assign words to subsystems (RD, BV, etc.). The correspondence to the banks is:

SYS	Bank
0	NOPA (the non-bank, since it isn't supposed to appear (but does))
1	RDPA
2	BVPA
3	E2PA
4	BMPA
5	COPA
6	EHPA
7	ICPA
8-F	not assigned

A further complication is that it would be good to provide for separate blocks from each DSP in use (we have 2 now, but initially I will push all the data through one DSP). I have shoved all the FERA data into one bank (SPRA) with this format:

```
- Event number
- Number of DSP's (1 or 2, in the future as large as 8)
- Inclusive WC from DSP 0--this includes status and data words and this word
  In the event of no FERA data from DSP 0, this will be 1.
- DSP 0: A side status register before copy to b ram
- DSP 0: A side info register before copy to b ram
- DSP 0: first data word from fera in above format
...
- DSP 0: last data word from fera in above format
- DSP 0: A side status register after copy to b ram
- DSP 0: A side info register after copy to b ram
- Inclusive WC from DSP 1
  In the event of no FERA data from DSP 1, this will be 1.
- DSP 1: A side status register before copy to b ram
- DSP 1: A side info register before copy to b ram
```

- DSP 1: first data word from fera in above format
...
- DSP 1: last data word from fera in above format
- DSP 1: A side status register after copy to b ram
- DSP 1: A side info register after copy to b ram

Words marked DSP 0 or DSP 1 are made exclusively in the DSP; the other words are constructed in the SSP. Note that the inclusive wc's can be 0 for events with no fera data at all (which can happen for some pathological triggers, like the pulser).

In order to create the *PA banks, one need ignore words with bit 15 set, mask off bits 11-14 in the data words, and map the upper four "system" bits into the appropriate bank (RDPA, BVPA, etc.).

Furthermore, the pedestals that are loaded into the FERA memory are 4 counts higher than the measured pedestal in order to suppress pedestal noise. I suggest we change this to 2 counts (about 3 sigma on average). Effectively, we were subtracting adc - (measured pedestal + 4) from the ADC value, so 2 counts have to be added back to the raw data in order to account for this. In the past, this was done in the SSP; I propose that we now do this during the formation of the *PA banks. The only reason to do this online is if we ever want to do this channel-by-channel, which would be much cleaner if done in the DSP.

The data format can be checked for consistency:

o There should always be XWC words following a word with bit 15 set,
  followed by another word with bit 15 set.

o There should never be duplicate VSN's.

o There should not be duplicate SA's within a single VSN.

o In principle, one could check that the logical address correctly
  corresponds to the VSN+SA, but that will be wrung out by debugging

```
  and the pedestal run.  That would require that  keep the address maps
  in CFM, which would be painful.

o Bits 8,9,10 should be zero in vsn header words.
```

7.7 Other Software

Some other software developed for the dsp is in /e787u/haggerty/online/qdp. The test subbdirectory contains a Struck 370 test program written by Tino Haupke at Struck, and ported to our FASTBUS system. The exercise subdirectory contains some programs which can be used to read the dsp data "by hand," and make a quick check on the data format.

8 Future Possibilities

There are a number of possibilities for increasing the performance of the DSP readout and trigger. The Level 1.2 trigger could be sped up by dividing the range stack FERA's into smaller readout chunks. This would make it faster to get to the ADC words that are capable of failing the event. A natural division would be into four pieces, since there are four range stack FERA crates, and there are four possible DSP's on a single FASTBUS board. There is some complication introduced by the Struck 370, since only the top DSP can directly toggle the ECL outputs used to produce done and return code, but this could be handled in (DSP) software. In principle, it should be possible to speed up writing the data into the FIFO as well, since it is now writing at about 270 ns/word, and in principle should be able to write at 100 ns/word. Since writing into the FIFO and operating on the data happen sequentially, it is desirable to both have less data to loop through, and to push it into the FIFO faster. It is posssible to imagine getting the execution time down to around 100 μs; as we reduce the readout time toward 1 ms, this means that we may be able to push f down to around 0.1.

Another approach to Level 1.2 would be do it in hardware using either a new board or a general purpose programmable logic module like the LeCroy 2366. In that case, one would hope to be able to handle the data "on the fly" without slowing down the ECL bus, and come to a trigger decision as the words are strobed past. This was the philosophy of the "Level 1.5" trigger used in the 1990 and 1991 runs, and implemented with LeCroy MLU's and ALU's. The complication in using these modules is that the ECL bus has to be broken into pieces and sent several places, since the same bits have different meaning at different times (for example, the VSN bits and the data are in the same position, only at different times). It would be better to do this kind of routing internally on a FPGA or a PC board. Another possibility for simplification would be to modify the stop channel board so that it can redrive several copies of necessary signals, and thus route signals on this board rather than with external cables. It is not clear whether the FPGA in the LeCroy 2366 (the Xilinx 4005PG156) is fast enough or has enough gates to carry out the existing algorithm; some preliminary design work is needed to decide that.

Breaking up the FERA readout into pieces by CAMAC crates would also isolate problems with the readout to a small number of modules, although after the readout was switched to the DSP, there was no evidence of any data corruption like that which occurred in 1995. The cost is not negligible, since a 4 DSP FASTBUS modules is listed for $21,760, and 2 additional 96002 boards for the existing Struck 370 would cost $6820 (February, 1996 LeCroy price list). It may be necessary to break up the FERA bus at some time in the future in order to keep the readout deadtime caused by the FERA's below that of other systems, but there is no indication that this is necessary yet.

9 Acknowledgements

Thanks to Marty Van Lith for building the wire-wrap stop channel board, and producing the fancy schematic (ex-post facto). Tino Haupke at Struck provided invaluable guidance in the early stages of testing, programming, and using the Struck 370 module.

References

[1] Haupke, Tino, *Struck 370 Users Guide*, Struck.

[2] Motorola, *DSP96002 IEEE Floating-Point Dual-Port Processor Users Manual*, Motorola, Phoenix, Arizona (1989).