CDF/DOC/TRIGGER/4231
Version 2.0 October 15, 1999

The Anatomy of a DIRAC

John Wahl and Pedro Amaral
University of Chicago

Abstract

This note describes the Digital Information Receive and Compare (DIRAC) board in gory detail. A complete understanding of note CDF/DOC/TRIGGER/4225 is assumed.[1] This note is intended as a guide to maintaining and repairing Dirac boards for future runs.

1 Organization

It is a far far better place you go than you have ever gone. It is a far far better thing you do than you have ever done. Well, keep telling yourself that in any case.

This paper is organized by functionality of the Dirac boards. The first section describes the overall control signals used on Dirac.(The Nervous System) The second section describes the data path of Dirac.(The Digestive Track) The third section describes the clocking of the data on Dirac.(The Circulatory System) The fourth section explains the VME interface.(The Five Senses) The last section describes all other miscellaneous pieces of Dirac.(Liver, Spleen and that stuff) Many acronyms and abbreviations will be used throughout this document, so I have provided a glossary in an appendix.(Just like the real thing, useful, but not necessary) You will need to obtain a copy of the schematics, and I would suggest you obtain a C size copy. B size should be adequate though.(Consider this the first test of whether you should be reading this document!) There are numerous references to technical notes throughout the document. These are listed in appendix B and refer to information that is either VERY detailed and/or pertains to the reasoning behind some feature. A new section was added to help those intending to perfom surgery on Dirac boards.

2 The Nervous System

2.1 Introduction

Like a biological nervous system, Dirac's Brain and nerves span the whole board and control all aspects of operation. The Brain chip is located on schematic sheet 6. The **Control(130:0)** bus extends over the whole board and controls most operations. There is a second control bus that specifically controls the Heart. This bus is called **Heart_Ctrl(16:0)**.

2.2 The Control Bus

This bus is arguably the most important bus on Dirac. Tables 1 and 2 list all bits on the bus with a short description. These signals will be referenced throughout this paper. Most of this bus originates in the Brain, some go to the Error chip(EF* and FF* from the FIFOs) and a few emanate from the Heart.

2.3 The Brain

The Brain chip located on schematic sheet 6 is the global control chip for the board. The Brain is pretty much dormant during data taking and the Heart takes control. The Brain is used to decode the VME address for DAQ readout during data taking. You will need to obtain a copy of the Altera AHDL source code for the programming of the Brain, it is called CONTROL.TDF.

Table 1: Dirac's Control Bus

Bit #	Description
0	OE* for FP Connectors(Differential Drivers)
1	OE* for Tracking Receivers
2	OE* for Tracking FIFO
3	WCLK for Tracking FIFO
4	OE* for Tracking SRAMs
5	CLK for Tracking Diag Register
6	OE* for Tracking Diag Register
7	OE* for Tracking SRAM's Address Buffers
8	OE* for Tracking FIFO's Data Buffer
9	OE* for Tracking SRAM's Data Buffers
10	WE* for Tracking SRAM's
14 : 12	CLK for VME access to EM TowLat latches
17 : 15	CLK for VME access to HAD TowLat latches
18	Bit to select 3 bits used in LSBHAD
24 : 19	EM Tower Masking Bits
30 : 25	HAD Tower Masking Bits
121, 32 : 31	Bits to select what data to read from TowLat
33	Enable* for TowLat Chips
36 : 34	Select for VME Reading from TowLat Chips
37	CLK to Strobe Data through all TowLat Chips
38	Tower Granularity Select Bit
39	RCLK for Alignment FIFOs
40	WCLK for Alignment FIFOs
41	OE* for Alignment FIFOs
42	RESET* for Alignment FIFOs
49 : 43	EF* for Alignment FIFOs
56 : 50	FF* for Alignment FIFOs
59 : 57	OE* for TT SRAM's Address Buffers
62 : 60	WE* for TT SRAMs
63	OE* for TT SRAMs
66 : 64	OE* for TT SRAM's Data Buffers
67	RCLK for L1 FIFOs
68	WCLK for L1 FIFOs
69	RESET* for L1 FIFOs
76 : 70	EF* for L1 FIFOs
83 : 77	FF* for L1 FIFOs

Table 2: Dirac's Control Bus

Bit #	Description
84	L2 Buffer Read Address B
85	L2 Buffer Read Address A
86	L2 Buffer Write Address B
87	L2 Buffer Write Address A
90 : 88	GR* for Reading L2 Buffers
91	GW* for Writing L2 Buffers
92	OE* for Csum Chips
93	CLK to Strobe Data into Csum Chips
94	OE* for CSUM Data Buffers
95	Latch Enable for CSUM MUX chip
96	Clock for CSUM MUX Internal Register
97	Enable* for VME Reading of Data in CSUM MUX
98	OE* for CSUM Backplane Drivers
99	CLK for EtInp Chips
101 : 100	Energy Summing Threshold
102	OE* for EtOut Chip
103	OE* for EtOut Data Buffer
104	OE* for EtSum Backplane Driver
105	CLK for EtSum Backplane Driver
106	GR* for Bunch Counter L2 Buffers
107	RESET* for Bunch Counter
108	CLK to Write Offset in Bunch Counter
109	RESET* for Error Chip
111 : 110	VME Reading Adress for Error Chip
112	CLK to Strobe Data into Error Chip
113	REN* for All FIFOs
114	WEN* for All FIFOs
115	OE* for Error Chip Data Access
116	RCLK for Tracking FIFO
117	FF* for Tracking FIFO
118	EF* for Tracking FIFO
119	Enable for AUX Card Summing 0. 5+12=5
120	Enable for AUX Card Summing 1. 7+12=7
120	AUX Card Spare
121	Enable input from AUX Card
130 : 121	Spare(Unrouted)

The first few pages list the input and output pins for the Brain. Any signal that is on the Control Bus is listed on the far right side of the pages. An important piece of this list is the **HEART_CTRL[9..0]** BIDIR pins. This is the bus that goes from the Brain to the Heart. Three things are done over this bus:

1. Calorimetry and Tracking delays. These define the position of Dirac in the L1 Pipeline.

2. Read–back. This allows reading out the delays downloaded to the Heart via VME.

3. Strobes. The infamous strobe register in the Brain merely echoes the data along this bus telling the Heart to issue the appropriate clock signal.

There are several other control signals that talk to the Heart and all signals are preceded by **HEART**.

The next section is the internal node definitions. These are convenience definitions to make the rest of the code more readable. The **ADD_FLAG[54..0]** LCELL is an array of bits that tell which piece of Dirac is being addressed. This is the result of all address decoding done by the Brain. Virtually all control signals distributed to Dirac depend on this array at some level. **OUT_DATA** and **OUT_HEART** are both internal busses for VME Data and Heart Data. The nodes preceded by **ZERO** are used in the address decoding process. Many of the addresses used contain many zeros in the bits, so these nodes are on when the bits listed are ALL zero. For example, **ZERO_6_19** is on when bits 6 through 19 of the address are zero. Next are most of the internal Control registers listed in the memory map; **CTRLZERO[9..0]** is **CONTROL_0**, etc.... The final definitions are for the **IDPROM**. There are many 8–bit sets defined. Each set defines a byte of the **IDPROM**. Currently they are set so that the IDPROM registers read (on the upper 8 data bits of each address): 00XX 005 DIRAC. For more details read technical note 1.

The logic definition section is next. The first section are defaults for some of the signals in the Brain. These define the state of the board at powerup. The **WRITE** signal from VME is registered at the start of a VME transaction. There is a fairly large section for the address decoding. First the internal **ZERO** nodes are defined. Then the address decoding begins. Address lines 23–20 are used to decode the 'Block' being addressed. The lower address bits are decoded and finally a bit of the **ADD_FLAG** list is set. Most of the decoding includes checking the state of the **WRITE** signal. An address space that is read–only can be distinct from a write–only space but occupy the same numerical addresses. This is not un–common on Dirac. This continues for several pages, there is no good way to explain this section. Read and learn.

Many of the internal registers and busses are handled next. First the **IDPROM** is defined in ASCII. The inputs for the control and masking registers are connected to the VME Data lines, **D[9..0]**. The strobe register is defined. There are several pieces of Dirac that require a strobe from the Heart. These are easy to pick out in this section. For example, if the L1 FIFOs are being written to(**ADD_FLAG25**, **26** or **27**) then strobe bit 5 is turned on. The heart then sends a WCLK to the L1 FIFOs and the data is written into the FIFO. There are 4 inputs to this bus, the Alignment FIFOs, the L1 FIFOs, the EtSum backplane driver and the normal VME Data lines. **OUT_HEART** and **OUT_DATA** define the logic for all the data going to the Heart chip and back to VME. These data are then used to feed tri–state buffers, **BUFFZERO** and **BUFFONE**, that drive the actual outputs of the Brain. The buffers are enabled if the piece of memory being addressed requires outputing data to either the Heart or VME. The enables for the buffers are **BUFFZERO.en** and **BUFFONE.en**. The internal register clocking is defined next. The registers are clocked by a delayed signal from an external chip. The **DS*** VME signal is sent to the chip labeled **VME Timing Taps** on schematic sheet 6. This chip then sends delayed versions of **DS*** to the Brain, called **Brain_Taps(4:0)**. Each bit of this bus is delayed by 25ns. The internal registers are clocked by **TAP2**, 75ns after the VME cycle starts, allowing plenty of time for the data to arrive at the Brain. The 0th bit of the control registers and the whole strobe register are write–only registers. Therefore, the **.clrn** bit is turned on at the end of a VME cycle, clearing the registers.

The last section of the Brain is the logic to control the board. Ironically, I am not going to discuss this section. You should refer back to this logic as I discuss various parts of Dirac. The logic to turn on an enable or set a particular bit is probably found in this section. I will comment on the last bit of logic in this section, the **ACK** signal. For a VME transaction to be concluded Dirac must return an ACK(acknowledge) signal to VME Master. The **TAP** used to clock **ACK_FF** determines the length of the VME cycle. This must be longer than the slowest transaction that Dirac will perform. Currently it is set to **TAP3** or 100ns.

3 The Digestive Track

3.1 Introduction

This section describes the actual data path followed by the data as it is digested by Dirac. Calorimetry data from the ADMEM FE boards enter Dirac in the **Custom_P3** section of schematic sheet 1.[2] The data is processed by the **Tower** block. Tracking data from the XTRP boards enter from the **Front_Panel** section and proceeds through the **Track** section.[3] The **Tower** section combines the Tracking and Calorimetry data for trigger decisions on Dirac. The resulting decisions are packed into the **CSUM** bus. **CardSum** summarizes the trigger decisions and sends them through **Custom_P3** to a CrateSum(CS) board in the crate.[4] Each trigger tower is divided into an electromagnetic and hadronic component. These energies are summed and sent to the **EtSum** section from the **Tower** section. The **EtSum** section finishes summing the energies and passes the total energy for Dirac to the CS trough **Custom_P3**. The energies used by L2 for calorimetry clustering are obtained from Dirac. The **FTOWERS** bus is the data going to L2 through the **Front_Panel** from the **Tower** section. Finally, data can be obtained from the **Tower** section through the **Data** bus to VME. The **CSUM** bus data are stored in L1 FIFOs and L2 Buffers for later readout by the CDF DAQ system in the event of a L2 accept. That should be sufficient for your understanding. . .just kidding, the following sections fully describe all these functions.

3.2 The Mouth

Data, like any good food, is received through the mouth. Dirac's mouth is the **Custom_P3** section shown in schematic sheet 11. The top 5 rows of data are the 16–bits of trigger summary information to CS, **TRSUM(15:0)**. So this is output data, I never said the mouth analogy was perfect! Rows 6–39 are the data coming into Dirac through the backplane from the AUX card.[5] The data are divided into cables coming from the detector. Each cable carries 10 bits of energy differentially. The AUX card receives the data and passes 10 bits to Dirac. There are 12 cables for each AUX[1] for a total of 120 bits entering Dirac. Dirac organizes this information into 6 trigger towers containing 10 bits of hadronic and 10 bits of electromagnetic energy. The **TOWIN(119:0)** bus is all these energies. The last 2.5 rows contain the total energy of Dirac that goes to CS across P3, **ETSUM(7:0)**. In the lower rows, there are also 3 bits from the Control chip. One bit enables the inputs from the AUX card, the other two enable the summing of towers 5+12 = 5 (AUX_CTRL0) and of towers 7+12 = 7 (AUX_CTRL1). These bits are controlled from the **Control_0** and **Control_1** registers.

3.3 The Stomach

Schematic sheet 14 is a detailed representation of the **Tower** block. The data first enters the **TOWLAT** section. Sheet 15 shows the actual tower latch chip. There is 1 chip per trigger tower, receiving the EM and HAD energy for each tower. The control lines used by the TOWLAT chip are shown in the upper left of the schematic; the control lines are detailed on most schematics as a quick reference. Before describing the internal operation of the TOWLAT chip, the operation of **TDO_TDI(6:0)** should be described. This bus is used to feed the ISP serial chain for TOWLAT. The first bit is TDI for the first chip, the second bit is TDO for first chip and TDI for second chip, etc. . .. This was represented on the schematic in this fashion because of the frame surrounding the TOWLAT chip. Now on to the internals.

You will need to obtain a copy of the Altera schematics for this design, it is called TOWMASK.GDF. In the upper left of the schematic, the data gets first passed through a multiplexer. This multiplexes the calorimetry data with VME data to allow a user to write fake data to the tower latches. **TOWENA** determines which data gets passed. **TOWENA** comes directly from the Brain and is on if Dirac is in **RUN** mode and off for **LOAD** mode. The data, **IEM** and **IHAD**, are now sent to the tower latches. The latches are divided into groups of 5–bits because this is the width of the VME data lines to the **TOWLAT** chips. There are 2 ways to clock the latches. The Heart will clock all

[1]Actually some AUX cards have a 13th cable from an Endwall tower. See reference [5] for more detail

four latches during normal data taking, via the **TOWCLK** control signal. A user can also write data to each set of 5–bits via the **HADVME** and **EMVME** control signals, plus the **VMESEL0** signal. Using these three signals, unique data can be downloaded to all tower latches. This is very important for diagnostic programs that wish to check a unique configuration of the input data. The **MASKEM** and **MASKHAD** bits also control the state of the tower latches. There are 12 control lines from the Brain that turn off the latches for an individual tower, 6 towers $\times$ HAD+EM. If the mask signal is on, the tower energy is cleared and kept cleared until the mask signal is off. Thus the selected tower no longer contributes to decisions on Dirac. Currently there are no plans for a trigger based on the lack of energy in a tower at L1!

Now the energy bits are called **L2HAD** and **L2EM**. These bits are sent directly off the chip to form the bus called **FTOWERS** and sent to L2 DCAS via **Front_Panel**.[6] The middle right of the TOWLAT schematic selects off the L1 bits for use by Dirac. The signal **BIT_SEL** is the **TOW_GRAN** bit from the **Control_1** register. This bit selects either the upper 8 bits or the middle 8 bits from both the EM and HAD energy and produces **EM8** and **HAD8**. As you can tell, this bit literally determines the granularity of the tower energies. The data from the ADMEM cards has a resolution of 0.125GeV, so this bit chooses either 0.250GeV or 0.500GeV granularity for the L1 energies. Note: L2 has already been sent all 10 bits so still receives a resolution of 0.125GeV. In the lower right of the TOWLAT schematic is the energy sum and a little logic to check for overflows. The box labeled **spe8xor2** is a special block that checks the state of bit 9. If the bit is on and the middle 8 bits of the energy have been chosen, then all 8 bits are turned on. Obviously, if the 9th bit is on, we can not just ignore it, so we max out the energy for that tower. The bus **TOWEM** is sent off chip as the bus of the same name on schematic sheet 14. This is the EM energy used by Dirac to form trigger decisions. On the TOWLAT chip, the **TOWEM** bus is summed with the HAD energy to form **TOWTOT**. Likewise this is sent off chip and represents the jet energy used to form trigger decisions. Both **TOWEM** and **TOWTOT** will therefore have LSB of 0.250 (0.500 GeV), with a full range of 64 (128) GeV, depending on the granularity chosen. In the lower left corner of the TOWLAT schematic is the logic to select the **LSBHAD** bits. The signal **LSBSEL**(from **Control_1**) selects whether the lower 3 bits, 0–2, or bits 1–3 are used to form the **LSBHAD** bus. This is the crude HAD/EM bits used to form trigger decisions. Essentially, Dirac only looks to see if there is a little hadronic energy in the tower, the scale of how little is determined by the data in the Decision SRAMs. However, being physicists, we can place an upper limit on 'little'. 'Little' is either less than 0.875GeV(7 counts $\times$ 0.125GeV) or 1.750GeV depending on the state of **LSBSEL**.

The last section in the middle left is the VME readout logic. **CHIP_SEL** is on if this chip is selected for reading. It turns on tri–state buffers that then drive data onto the VME Data lines. What data is driven is a little more complicated. Table 3 shows the decoding of the data based on the state of **VMESEL** which comes from the Brain. Please read technical note 2 for the explanation why this decoding is so complicated. This completes the explanation of the TOWLAT chip. The Brain and this chip are 2

Table 3: TOWLAT Bit Packing

VMESEL state	Output Data
0	L2EM(4:0)
1	L2EM(9:5)
2	L2HAD(4:0)
3	L2HAD(9:5)
4	TOWEM(4:0)
5	LSBHAD(1:0),TOWEM(7:5)
6	TOWTOT(4:0)
7	0,LSBHAD(2),TOWTOT(7:5)

of the most complicated chips on Dirac, so have a little hope that you will survive this paper.

3.4 The Lower Stomach

The analogy gets a little vague here, but I am going to try to maintain the illusion! Data from the TOWLAT chips scatter all over Dirac. The **FTOWERS** bus speeds to the **Front_Panel** and **TOWTOT** heads to **EtSum**. These are discussed later. The fate of the set **TOWTOT**, **TOWEM** and **LSBHAD** is the subject of this section. These busses go to the **AL_FIFO** section, schematic sheet 16.

This section is pathetically simple! The data is read into a FIFO to align the calorimetry data with tracking data. Then the data is sent out to continue on its merry way. The purpose of this section is, surprisingly enough, alignment. It has now been decided that Calorimetry will arrive at Dirac BEFORE tracking, so that no FIFO capability for Tracking exists any longer...

The clocks are controlled by the Heart chip. Since calorimetry arrives first, the alignment FIFO will store events until the tracking data arrives and then start clocking data out again. Tracking Info MUST arrive at least 3 Clocks AFTER Calorimetry Info. You should read the data sheet for the FIFOs completely to truly understand this situation and how this logic works. Next!

3.5 The Small Intestine

If one considers the trigger decisions to be the nutrients that keep the trigger alive and running, then the **Decision** block on schematic sheet 17 is definitely the small intestine. The **Decision** section transforms the energy and tracking information into actual trigger decisions. This is performed by an array of SRAMs that are downloaded at the beginning of each run. There are 2 SRAM's per trigger tower, one to perform the electromagnetic and one for jet decisions.

The upper SRAM in the schematic is the EM decision SRAM. The address of the SRAM is formed by the 8–bits of EM L1 Energy, the 4 bits of tracking information for this tower and the 3 bits of **LSBHAD** derived from the hadron energy. Thus the SRAM needs a 15–bit address or 32K. Since there are 8 EM triggers the SRAM should be 32 KBytes. The lower SRAM is used to make the jet decisions. The address if formed from the 8–bit total tower energy and the same 4 bits of tracking data. This is only an 12–bit memory or 4K. There are also 8 jet trigger decisions to be made here so the memory must be at least 4 KBytes in size. I chose to use the same 32 KByte memories for the jet SRAM as well to reduce the number of different types of parts and reduce the cost(larger quantity). Thus the upper 3 address bits are grounded. There are address buffers for talking to the SRAMs on the left of the schematic. The accompanying data transceivers for each SRAM are located on the right.

Note: Since each tower takes 16 bits of data, it is logical to put 2 towers into the same address space. So what? The problem is the memories are different sizes for EM and jet decisions. If one proceeds to download the memories, one would create some translation between the energies and decisions. For the first 4K there is no problem. At the 2K barrier, what do you do? Continue with your translation table for the EM decisions for the remaining 28K, that is easy. However you will be simultaneously writing to the jet SRAMs. You must make sure to write the same data that you wrote in the first 4K block to the remaining 7 4K blocks. Be wary.

There are a couple of special bits in **Control_0** that pertain directly to the SRAMs. The **MLU_FIFO_EN** bit turns on the SRAMs and the FIFOs on Dirac.(Obviously this affects the FIFOs in the previous section also) This bit is used to turn on the FIFOs and SRAMs so that data can freely flow through the chips. Dirac acts as if this bit is ALWAYS on during **RUN** mode, so these bits would not affect the running state of the board.[2] The **MLU_READ_MODE** bit puts the SRAMs into the read state. That is, any address passed to the SRAM will be a lookup and drive the data lines with the value stored in the SRAM at that address. Why, why you might ask? It is easier to implement these bits than to have all the proper lines turned on when certain memory regions are accessed. For example, if I want to read the alignment FIFOs data, but do not want to affect the FIFO, I need to enable the FIFO, put the **Decision** SRAM into read mode, enable the SRAM and enable the SRAM data transceivers. That is alot of stuff to do, so I put these bits into the Control registers and now all the Brain must do is enable the data transceivers.

[2]The bit is not actually turned on. The signals that are asserted when this bit is turned on are also asserted during **RUN** mode. **RUN** mode overrides the state of this register.

3.6 The Large Intestine

Almost to the end of this path for the food, err data. The **DAQ** section on schematic sheet 18 is the holding pit for the data. This schematic contains the L1 FIFOs and L2 buffers. The main difference between the **DAQ** section and the large intestine is the **DAQ** section keeps data that we actually want to use later. If the event makes its way through L2 and a L2 accept is issued then any data that we want to add to the event stream is read from these L2 buffers. How does this work?

All trigger decisions(8 EM and 8 jet) come flying out of the **Decision** section every 132ns. The data are all written into the L1 FIFOs. Each FIFO is 16 bits wide, so there must be 6 per Dirac. This is automatic. What happens to this data? There must be a L1 accept or reject(**CDF_L1A** or **CDF_L1R**) signal from the TSI at some point.[7][8] The total length of the L1 pipeline is 42 segments, therefore this decision must come before 42 events pass through Dirac. In fact, since Dirac is about 12 segments into the pipeline, we expect a L1 decision about 30 clocks later.[3] Therefore, the FIFO must be 30 or so events deep. These FIFOs are 256 deep, plenty for our needs. If a FIFO fills up with data, it signifies a serious error on Dirac(this includes the alignment FIFOs). This means that a FIFO is not receiving a clock or may be losing an occasional clock. In any case, it is very bad. The FF*(Full Flag*) from all FIFOs are sent to the Error chip located on schematic sheet 10. The handling of this error is discussed later.

Back to the data. If Dirac receives a **CDF_L1R**, the L1 FIFO's WCLK is strobed, but the data is ignored by the L2 buffers. The data is trashed, sent to /dev/null, consigned to oblivion, pick your favorite phrase! If there is a **CDF_L1A**, then the WCLK is strobed again, but now the L2 buffers are also strobed to read the data into the buffer address specified by **CDF_L1BA(1:0)** from the TSI. The **CDF_L1A** or **CDF_L1R** is sent to the Heart chip and used to generate the appropriate clocks. The **CDF_L1BA** is sent to the Brain and echoed through the control bus bits called Write Address shown in the upper right corner of the schematic. The Brain also allows a user to fake a **CDF_L1A**. The write–only bit **L1A** in **Control_1** generates a fake L1A through the Heart. The value in the **L1_ADDR** register of **Control_1** is used to fake the **CDF_L1BA(1:0)** signals. Next, Dirac awaits a L2 decision. If the TSI decides to reject the event, the buffer is overwritten. If the TSI decides to accept the event, the buffer is read out via VME. Hence the direct connection of the L2 buffer data lines to the VME **Data** lines. In either case, Dirac is ignorant of the L2 decision made, it is a drone and merely responds to the commands of the TSI or the VRC.

This is a dead end path for the data. I prefer not to carry the biological analogy any further. The VME interface actually performs relatively well! Time to take a step back and examine the tracking data.

---

[3] The exact position of Dirac is something you may have to determine as it requires actually putting the system together and measuring the delay.

3.7 The IV Drug

Some say information is an addiction, for CDF this is definitely true. The tracking information available at L1 is very new for CDF and people already drool over the power it gives the L1 trigger. But that is the subject of other papers.

The addiction begins on sheet 19, the **Front_Panel**. The needle is located in the lower left of the schematic and is called the **TrackIn Receiver**. The data is sent over a 16 conductor cable and received by 2 4–bit differential receivers. The receivers are terminated with 2 100 Ohm resistors in series(200 Ohms). See technical note 3. The cable goes to two adjacent Dirac boards. The equivalent resistence seen by the driving end is 100 Ohms. The single–ended signals are bound into the **TRACKIN(7:0)** bus and sent to the **Track** section on schematic sheet 22.

The data is latched in the TrackIn Latch at a time t_{Tracking} after the interaction. This time is set in the Heart chip.

The data then proceeds through the bank of 3 SRAMs called **Tracking Look–up Memory** on the right side of the schematic. These SRAMs allow an arbitrary mapping of the 8 bits from the XTRP board to 3–bits per tower for Dirac. The data arriving from the XTRP board represents 8 tracking triggers. These can vary in definition by momentum, eta or isolation. Therefore a close coordination between the tracking definitions and the patterns downloaded to these memories is necessary. The previously mentioned bus **TR** is then sent to the **Decision** block within **Tower**.

The diagnostics for **Track** are extensive. The **Diag Register** on the lower left is used to fake tracking data from the front panel connector. The user can write to this register, turn off the receivers for **TRACKIN** and turn on this register. Then issue a fake tracking FIFO WCLK, viola the fake data is written into the tracking FIFO. The buffers in the lower center are for address lines to the SRAMs. There is one interesting combination worth mentioning. Dirac can pass the VME address through the **Load Address** buffer to the **TrackIn Data** and back to VME data. Therefore, the operation of these buffers can be tested by confirming that the data read back is just the address of the transaction.

The buffers on the lower right of the schematic are the SRAM data buffers that allow uploading and downloading of the SRAMs.

3.8 CO_2

The exhalation of Dirac is the L2 energies passed through the TOWLAT chips. In the **Front_Panel** section, schematic sheet 19, the **FTOWERS** bus is driven onto cables by the differential drivers on the far left of the schematic. These drivers can be disabled by the Brain via the **Control_0** bit **FTOW_EN**. Schematic sheet 20 shows the connectors that take the energies to L2. There are 3 80–pin connectors. Each connector carries 2 trigger towers. The mapping is pretty simple and there are no subtleties.

3.9 The 2nd Intestine

The analogy is really breaking down here! The TOWLAT chip sums the 2 8–bit L1 energies per trigger tower into 1 8–bit number with a resolution of 0.250 GeV or 0.500 GeV. Dirac must complete the summing of all energies on board and pass the information on to CS. This is done in the **EtSum** section, schematic sheet 21. The first stage consists of three **EtInp** chips.(Et Input) The final stage is a single **EtOut** chip.(yea, Et Output) The data is then passed through a latch onto the backplane to CS.

The **EtInp** chips are Altera EPLDs; you should obtain the schematics for these chips, called BIGADDER.GDF. Each chip receives 2 trigger towers' energy and sums them into a single 8–bit sum with 0.500GeV resolution if the TOWTOT energies have a granularity of 0.250 GeV (TOW_GRAN bit in Control1 OFF), or sums then into a 8-bit sum with a 1 GeV resolution if the TOWTOT energies have a granularity of 0.500 GeV (TOW_GRAN bit in Control1 ON). The **EtInp** chip applies a threshold, stored in **Control_1** as **ET_THRESH**. This 2–bit register represents the size in bits of the threshold. For example, if the register stores 10(binary) then the threshold is less than 3 counts. This is easy to see by looking at the lower section of the BIGADDER.GDF. The 3 lines **T(2:0)** are generated from the 2 bit **ADDR(2:0)**. If the 2 bits are 10 then **T(2:0)** is 100. This value is ANDed with the data if none of the higher bits in the energy are on. That is, if the energy is below 4 counts, 11 or less, then the threshold is applied. So if the energy is 00000010 and the threshold is set at 10, 00000010 (logical AND) 100 = 0. If the energy is 00010010, then the energy is still 00010010 since a higher bit is on. Now we have summed energies with a threshold applied. The sum is latched into the **DFF** at the center of BIGADDER.GDF. This designates the end of a pipeline stage.

The 3 8–bit sums then go to the **EtOut** chip for the final summing stage. This chip, as all EPLDs on Dirac, is an Altera EPLD. Note it is NOT part of the ISP daisy chain, and therefore its program can't be changed. The schematic is called ETOUT.GDF. There are two summing stages in the chip. The first stage sums 2 of the **EtInp** chip outputs and the second stage sums the output of the 1st stage with the 3rd **EtInp** chip. The **COUT** bit from both stages is ORed. If either summing stage overflows the final sum is ORed with both **COUT** bits. This represents an overflow which CS understands. The output of **EtOut** is passed through a tri–state buffer that is enabled by the Brain. This buffer lets the **EtOut** chip to be 'turned off' allowing data to be written to the backplane latch on schematic sheet 21.

Still sheet 21. On the far right of the schematic is the backplane latch. The clock signal for this latch comes from the Heart chip. The **ETSUM_EN** bit of **Control_1** controls OC* for the backplane latch, allowing a user to turn off the output to CS. The buffer on the lower part of the schematic allows a user to either read data from the **EtOut** chip or write data to the backplane latch. The Brain knows to generate a clock for the backplane latch through the Heart at the appropriate time. The data, called **ETSUM(7:0)** is now sent to **Custom_P3** and off to CS.

3.10 The 3rd Intestine?

The analogy is dead by this point. The **Decision** block within **Tower** produces the **CSUM** bus. This bus contains 16 bits representing which triggers passed for each trigger tower.(8 EM + 8 Jet) In addition to these signals being stored in the **DAQ** block, they are sent to the **CardSum** section for final summary. Schematic sheet 12 shows all the gory details of **CardSum**. This section is also referred to as the trigger summary section since it performs the trigger summaries for Dirac. The 96 (6×16) trigger bits are passed to 4 Altera EPLDs which sum all 16 triggers to a 2–bit value. Then the sums are passed to the **CSUM_MUX** chip to chose which of the 32 bits are actually passed along the 16 data lines to CS.

On sheet 12, in the upper left corner, are the 4 **CSUM** chips. The schematic for these Altera chips is called CSUM311.GDF. Do not ask what the 311 means, even I have forgotten. Each **CSUM** chip receives the trigger bits for 4 triggers, 2 EM and 2 Jet. There are 6 inputs per trigger(6 trigger towers) and these 6 bits are summed to form a 2–bit answer. These bits tell whether 0, 1, 2 or more than 2 towers pass a particular trigger. The overflow bit in the summers is ORed with the sum to indicate that more than 2 trigger towers fired the trigger. The resulting 8 bits per chip are latched by a clock sent from the Heart. After latching the outputs are passed through a tri–state buffer back outside the chip. Note that these 4 chips are NOT part of the ISP daisy chain, and therefore their program can't be changed.

The 8 bits from 4 chips(32 bits) are then sent to the **CSUM_MUX** chip shown in schematics 12 and 13. This chip is also an Altera EPLD and named CSUMMUX.GDF. This chip is non–trivial. The purpose of this chip is to pick and choose(hence the name multiplexer) which of the triggers get passed on down the line. On the left side of the schematic are a set of latches that perform in two ways. During data taking they are made transparent and merely pass the data through to the rest of the chip. During diagnostics, it is possible to strobe an arbitrary pattern into the latches. This allows testing the internal logic of the **CSUM_MUX** chip and testing the backplane and inputs to CS. The next bit of logic on the left side converts all the 2–bit trigger summaries into 1–bit summaries, 0 towers or >0 towers passed the trigger. It is important to keep in mind what is actually passed to CS. The 16 bits that go to CS are 10 1–bit triggers and 3 2–bit triggers. Thus the 1–bit compaction is needed. Next in line are the actual multiplexers. There are 8 2–1 multiplexers. These determine the first 8 1–bit triggers. They simply choose between the Jet or EM 1–bit triggers. How to set which is chosen is discussed later. The multiplexers to choose the last 2 1–bit triggers are 8-1 versions. The first allows the user to choose one of the 1st 4 EM or 4 Jet triggers. The second allows a choice of one of the last 4 EM or 4 Jet triggers. Then there are the 3 2–bit triggers. Dirac requires the 2–bit triggers that are passed to CS to be the 3 last trigger bits. Therefore, if you want to specify a di–electron trigger, it must occupy trigger 6, 7 or 8 of the 8 EM triggers. The multiplexer for the 2–bit triggers is then simply a series of 2–1 devices. The first 2–bit trigger is a choice of the 6th EM or Jet triggers, etc.... The flip–flops at the bottom of the schematic are used to store an internal address. This address is used to determine which triggers are passed to CS. The decoding is fairly simple now that you understand how this chip works. The table in the general Dirac document describes this mapping.[1] Finally the buffers on the lower right are used to read the state of the final trigger summaries back out to VME. The state of the internal register can not be directly read back, but can be inferred with a properly clever test routine.

Now we have a 16–bit word to send to CS. The buffers on the right side of sheet 12 perform this task. The enable comes from the Brain and is controlled by the **CSUM_EN** bit in the **Control_1** register. This allows a user to turn off the backplane drivers to CS. The **TRSUM** bus is then passed back to the **Custom_P3** module previously discussed. The last feature are the data transceivers located on the lower left of sheet 12. These can read out the data stored in the **CSUM** chips and write data to the **CSUM_MUX** chip. The choice of data to write to the **CSUM_MUX** chip is controlled by the Brain. There are 2 clocks entering the **CSUM_MUX** chip, **VME_CLK** and **REG_CLK**. **VME_CLK** writes to the input latches and **REG_CLK** writes to the internal register. The **TRIG_WRITE** signal is used to read the trigger outputs, **CSUMOUT**, by VME.

The last thing of note is the pipeline timing for **CardSum**. The end of the last pipeline section for this data is in the **CSUM** chips. Therefore data must get through the **CSUM_MUX** chip, through the backplane buffers, across the backplane and clocked into CS in 132ns. I have noted on the schematic that there are 49ns to traverse the backplane and get clocked into CS. If a chip should fail such that it runs much slower, it could cause many problems that would be very hard to determine. Keep this section in mind when debugging bizarre problems related to timing.

4 The Circulatory System

The Heart is definitely the center of the circulatory system. The timing signals emanating from the Heart chip are the arteries and as you will see it even has a pace maker. The Heart is located on schematic sheet 6 and 9 and the actual schematic for the Heart design is called HEART.GDF. Of course, it is an Altera EPLD. On the right side of the Heart (on sheet 9) are the clocks that drive Dirac. On the left side are the signals from the Brain that control the operation. The top has signals from the global trigger system and from a digital delay line that allows Dirac to clock any section with a 25ns granularity.

The Heart is a pretty complicated chip, arguably the most complicated on Dirac. The overall design consists of a register that stores the delays for the various clocks, a bit of logic for the startup sequence(how Dirac syncs with the rest of the trigger), and the clock generation logic. The clock logic contains a set of shift registers, a set of multiplexers and another set of shift registers. The first set of shift registers(SRs) count how many clocks have passed since the start of a run. The set of multiplexers use the bits stored in the delay registers to select off a signal from the first SR. This signal enables the second set of SRs. The final SR starts the actual clock generation sequence; this SR is the final enable for the clocks.

4.1 Delays Register

The upper left corner of the Heart schematic contains the internal delays register and tri–state buffers, the **DELAYS** register in the Dirac memory map. The buffers are for reading/writing data from/to the Heart from the Brain. The delay register contains 3 bits of offset and 2 bits of phase for both the calorimeter and tracking sections. The offset is used to select how far into the pipeline the system proceeds before the data arrives at Dirac. The phase bits select a 25ns interval within a single clock. There are 2 clocks used by Dirac. The normal data–taking clock is the **CDF_CLK** with a period of 132ns. The diagnostic clock is a doubled **SYSCLK** generated by VME with a period of ≈120ns. The user can define the time to start clocking Dirac with a resolution of 25ns using either clock. The **CTRLB** bus contains the outputs of the delays register.

4.2 Startup and Shift Registers

So how are the clock generated? Next to the delays register is a set of 4 5–bit SRs. These are used to keep track of how many clocks the Dirac board is into the run. Once the clocks start going, one is generated every **CLK**[4]; the only time this SR bank is important is for the initial startup. See technical note 5. The serial input to the SR is just VCC. The SR is reset by a **_RESET** signal from the Brain. That leaves the clock for the SR. The clock is generated by the logic in the middle left of the schematic. It truly is ugly, but that is the beloved CDF startup sequence.[9] The simple part is the lower section that selects which **CLK** to utilize. The **VMEclk** is doubled and a multiplexer picks either the **VMEclk*2** or the **CDF_CLK** based on the state of the **CDF/VME** bit from the Brain. The **CDF_CLK** is turned off if a **_HALT** is received from the TSI. The **CDF_CLK** is also killed if the proper startup sequence has not been issued. The proper sequence has a **_HALT** followed by a **_RESET** then the de–assertion of **_HALT**. Then the system is ready to be synced by the **_B0** signal. In logic, the **_HALT** signal clocks VCC into a flip–flop that turns off all **CDF_CLK**s through an OR gate. The CLRN input to the flip–flop is disabled while the **_HALT** signal is asserted through another OR gate. After the **_HALT** signal is de–asserted, the next **_B0** will clear the flip–flop enabling the **CDF_CLK** to proceed to the multiplexer. The **_RESET** can come at any point; Dirac does not actually insure it occurs while the **_HALT** signal is asserted. This is the startup sequence. The occurance of the **_B0** after the end of **_HALT** syncs all the trigger boards to a common signal.

The **CLK** signal that comes out the multiplexer is used to clock the first SR. This signal also leaves the Heart chip to pass through a 10–tap digital delay line called **Data Timing Taps** on schematic sheet 5. The clock is echoed 10 times each delayed by 25ns from the previous signal and passed back to the Heart. The Heart does not use the 10th delay. These **TAPS** are used by the Heart for the fine granularity discussed earlier. After the SR are 3 banks of multiplexers. The first bank determines which signals clock the

[4]For simplicity, I will refer to **CLK** to mean either the **CDF_CLK** or the VME **SYSCLK**

calorimetry data. This refers to the **TOWLAT**, **AL_FIFO** and **EtSum** sections. The second bank selects the clock for the **Track** FIFO. The last set picks the clocks after both calorimetry and tracking data are aligned. This occurs for the **Decision**, **CSUM** and **DAQ** sections. The bits that actually select the clocks are from the **Delays** register. What is selected are taps from the first SR. This means that we are selecting a delay at the start of a run in units of **CLK**. After the clock turns on the selected tap, the SR is always ON unless a **_RESET** is issued. Therefore the clocks will continue to flow out of the Heart every **CLK**. The other part of the multiplexer banks selects the phase delay for each set of data. The phase delay comes from the **TAPS** bus. Two **TAPS** are used in each section, one delayed by 2 **TAPS**. The first tap clocks the second bank of SRs. The second tap actually generates the clocks that leave the Heart. Two taps are required to synchronize all the signals to a common edge without worrying about race conditions between the second SRs and the tap used to generate the clock signal. I chose 2 taps(50ns) because it was longer than the delays for the SRs.

The final bank of SRs is clocked by the phase selected signal and the input is the selected output of the first SR bank. The SRs are cleared by a **_RESET**. The outputs of the SR enable the actual clocks. Thus, at the start of run, the first clock is enabled first. One clock later, the next clock is also enabled. Note: the first clock stays enabled. And so on.

4.3 Clock Generation

The actual generation of the clock is through the final AND gates on the far right of the schematic. Each clock being generated is an AND of the second SR output and the 2–clock delayed tap. In some cases there are other signals involved.

Timer0 is the clock for the **TOWLAT** latches. During diagnostics we want to turn this clock off when using the VME **SYSCLK** to strobe Dirac. The SR output for this clock is ANDed with the **CDF/VME** signal, turning it off when in VME mode. **Timer6**(L1 FIFO WCLK) and **Timer7**(CSUM Latches) are really the same clock, but the lines on the PCB are routed separately in case there is a future desire to de–couple the clocks. There are 3 special signals sent to the Heart, **_AL_EMPTY**, **_L1_EMPTY** and **_TR_EMPTY**. Each of these signals indicates that a set of FIFOs on Dirac is empty. The **ERROR** chip generates the first 2 signals and the 3rd comes directly from the tracking FIFO. (There is only one FIFO) When any FIFO is empty, after the first WCLK is sent, the next RCLK only updates the status flags for the FIFOs. The data that was written into the FIFO is not actually sent the outputs, that requires a second RCLK. Thereafter, the RCLKs work as expected. The **_EMPTY** signals are used to generate this extra flag update clock. Actually the Heart uses these signals to block future generation of the clocks; after the FIFOs are "not empty", the appropriate **_EMPTY** signal is de–asserted. Note how **Timer4** (Alignment Fifo Read CLK) and **Timer5** (Tracking input CLK) are related. When tracking info is 1st available, it's latched in, propagated through Tracking Memories and available at the Decision memories along with the Calorimeter info stored in AL_FIFO and just read by a coincident clock, **Timer4**. The _AL_empty info coming in to form **Timer4** allows for the PREVIOUS generation of the extra read clock of the fifo, before the first real read cycle, coincident with Track Latch-In.

Timer8 and **Timer9** are the RCLK for the L1 FIFOs and the GW* clock for the L2 buffers. When there is a L1 decision, accept or reject, a RCLK is issued to the L1 FIFOs. When a **CDF_L1A** occurs, there needs to be a GW* signal to the L2 buffers. The RCLK is generated in the same fashion as the previously discussed RCLKs for FIFOs. However, the clocks after that are generated only off **_L1A** or **_L1R** signals sent directly to the Heart from the TSI. For timing reasons **timer8** is enabled by **TAPS0** and **Timer9** is a 50ns wide pulse generated by **TAPS1** and **TAPS3**. Data is strobed out of the L1 FIFOs in phase with **CLK** and written into the L2 buffers 75ns after the **CLK**. **Timer9** can also be faked by the Brain. This is determined by the state of **run/halt**. When Dirac is in **RUN** mode, the TSI must issue all **_L1A**s. When performing diagnostics(**LOAD** mode) the Brain must issue all L1 accepts via **_L1A_VME**.

4.4 Fake Clock Generation

The **Strobe** register on Dirac allows the user to generate a fake clock. All clocks can be generated in this fashion except the GW* for the L2 buffers. GW* must be generated by writing to the **L1_ACC** bit of **Control_1**. There is a simple reason why the **_L1A** is not faked in the same fashion. The **Timer9** signal is inverted from the others. It, of course, would be possible, but was implemented separately instead. The logic in the upper right corner of the Heart schematic generates the fake clocks. This circuit is a simple multiplexer that chooses between the timing signals generated by the Heart, **Timer[8..0]**, and the data lines from the Brain enabled by **Step**. The choice is made based on the state of **run/halt**. When in **RUN** mode, the Heart generated clocks are always used. Conversely, when in **LOAD** mode, the fake clocks from the Brain are used.

4.5 Important hardware note.

The timing output signals of the clock, Timer0-9, are pins 63, 64, 65, 67, 68, 69, 70, 73, 74, 75 and HAVE to be specified as Slow slew-rate output in the Altera software [5]. Otherwise, we found that on the pre-production and production boards, there were timing problems causing reflections that were ABSENT on the prototype.

You are done with the Heart. Fun chip, huh?

5 The Five Senses

The VME interface is Dirac's link to the outside world. It is how it feels, listens, sees, touches and smells? Well four senses in any case. You should read portions of the *VME*

[5] Do Assign->Logic options from thge Altera 9.21 main window. Check the .rpt file after compiling...

Handbook so you understand the VME protocol.[10] These schematics are numbered 1–4 and have a separate reference number. This is because **VME_Interface** is an independent design, easily portable between designs. Sheet 2 contains the interface, the other three sheets contain the connectors.

5.1 The Interface

On the left side of sheet 2 are the connectors. Dirac is addressed via 5 geographical addressing(GA) pins. This means that there are 5 dedicated lines on the backplane that count from slot 0–20.(There are only 21 slots in a standard VME crate) The lines are terminated such that slot 0 is 11111, slot 1 is 11110, etc....(Inverse logic) The set of inverters below the **Address Decoder** section on the left of sheet 2 converts the GA to normal logic. The upper comparator in **Address Decoder** compares the **LWORD*** to ground to insure we are using long word transfers(32 Bit). The upper comparator also looks at the address modifier(AM) bits and sees if the VMEbus master is using an extended supervisory or non–privileged transfer. If both these conditions are true, the upper comparator enables the lower comparator. The lower comparator compares the GA to the upper 5 address bits being broadcast by a VMEbus master. It also compares address bits 26:24 to 000, as specified in the Dirac memory map. If the upper 8 bits are ok, then the **VME2000** chip receives an **ADEN***, address enable, signal.

The **VME2000** chip is the central controller of the VME interface. When the **ADEN*** and **AS*** signals are asserted, the **VME2000** asserts **MODSEL***, module select. As far as Dirac is concerned, the **DS0*** and **DS1*** signals are identical and the **VME2000** ORs them to produce **DS***. Dirac ignores the **D0EN*** and **D1EN*** signals produced by the **VME200** chip. As discussed previously, the Brain returns an **ACK*** to indicate completion of a VME cycle. This signal passes through the **VME2000** and is driven onto the VMEbus as **DTACK***, data acknowledge. There is also a VME bus error, **BERR***, signal controlled by **ERR***. **ERR*** is driven by the Brain, but currently there are no VME errors defined, so the Brain keeps the signal disabled. Dirac does not recognize any interrupts.

The **Data Transceivers** on the right of schematic 2 isolate the VMEbus data lines from the internal Dirac data bus. Data is enabled by the assertion of **MODSEL***. The direction the data flows is controlled by the **WRITE*** signal. During VME read cycles, data is driven on to the VMEbus backplane by Dirac. In a VME write cycle, data is driven onto the internal Dirac data bus. The previous references throughout this paper to the VME data bus really refer to the internal data bus.

The last piece of the VMEbus interface is the **Address Buffer/Counter** section located in the lower center of sheet 2. This latches the address that the VMEbus master broadcasts and also controls incrementing the address during block transfers. Two strobes are generated by the delay lines in the center of the schematic. One strobe is generated by the **MODSEL*** signal and pre–sets the data in the counters to the address on the backplane. The other strobe is generated from the end of the **DS*** signal. After the current cycle is completed, this strobe increments the address counters. In block transfer, this would prepare Dirac for the next data transfer. In a single–word transfer, the strobe is ignored. There are 2 counters that increment the lower 6 address bits. VMEbus block transfers are limited to 256 bytes, or 64 long words. Thus only 6 address bits need to be controlled by the counters. The other 16 address bits are stored in normal flip–flops, clocked by the **MODSEL*** signal. The final address latch clocks AM bit 1. This bit is set if there is a block transfer(BLT) occurring. This bit then enables the clock on the counters.

5.2 The World

The backplane **VMEBUS_P0**, **VMEBUS_P1** and **VMEBUS_P2** connectors are Dirac's link to the external world. Each connector conforms to the VME-P 64 standard.[11] **VMEBUS_P0** is located on sheet 4. P0 contains only power pins as far as Dirac is concerned. Dirac receives about 50% of it 5V power from P0. **VMEBUS_P1** is on sheet 3. P1 contains some +5V power for Dirac and many VMEbus signals. The VMEbus signals on these connectors are all standard, please refer to the VMEbus documents for details.[11] In column D are the geographical address pins with their pull–up resistors. **VMEBUS_P2** is located on schematic 5. The center column of P2 contains VMEbus signals. Columns A and C contain CDF defined signals. The signals relavent to Dirac were listed in the general Dirac document; this is where they originate. The only thing of note is the open–collector driver for the **CDF_ERROR*** signal. This allows multiple chips to drive the **CDF_ERROR*** line and the result is a wire–or of all the signals.

6 Miscellaneous Internal Organs

You are on the home stretch, this section describes all the remaining miscellaneous parts of Dirac. Most pieces are on sheet 6, the **Control** section, but some have their own schematics.

6.1 The Liver

The **ERROR** chip on schematic sheet 6 is Dirac's liver. The schematic for the **ERROR** chip is called ERROR.GDF. The **ERROR** chip receives all the EF* and FF* from the FIFOs on Dirac. At the top of the schematic is the logic for the FF*s. **/L1FF[6..0]** are the FF*s from the L1 FIFOs and **/ALFF[6..0]** are from the alignment FIFOs. In each **ffmacro**, the FF*s are ANDed together and the final answer from each block is ANDed to produce the **/ERROR** signal that is sent to the Brain. This signal indicates that one of the FIFOs has filled up with data, which should never happen during regular data taking. **/L1EF[6..0]** is received by a **efmacro** block. This receives the L1 FIFO EF*s, ANDs them together and passes **/L1EMPTY** on to the Heart. The final signal indicates that one of the L1 FIFOs is empty and the Heart needs to send the extra clock

Table 4: Write Mapping of the **ERROR** Data

Register	Mapping
L1 Full Flags	/L1FF[6..0]
AL Full Flags	/TRFF, /ALFF[6..0]
L1 Empty Flags	/L1EF[6..0]
AL Empty Flags	/TREF, /ALEF[6..0]

to update the FIFOs flags. The last block is similar but handles the alignment FIFOs and sends **/ALEMPTY** on to the Heart.

Both **efmacro** and **ffmacro** contain logic to mask off a bad channel. By writing to the appropriate **ERROR** chip registers, one can turn off a single FIFOs contribution to the **ERROR** chip. This is used in case a single chip is broken, but CDF does not want to stop running to replace the board. The offending channel can be masked and the run can continue. Note: you absolutely MUST mask off the tower energy as well as the **ERROR** chip flags. It is possible that only the flags are bad, but that is highly unlikely. There are 7 FIFOs in each set, so how do they map to the FIFOs on Dirac? The first 6 in each set correspond to the regular trigger tower FIFOs. The 7th alignment FIFO handles the **LSBHAD** data and the 7th L1 FIFO is the **Bunch_Counter**'s FIFO. Now it should be clear what the error chip does and how to fix a problem, but what about reading back the state of the flags?

The logic on the lower part of the schematic takes care of that problem. In each **efmacro** there are actually 2 paths the data takes, a fast and a slow path. Since an error is not time critical on the **CDF_CLK** scale, all the FF*s are slow. The fast path ANDs the EF*s and passes the result to the Heart ASAP. If this were the only path, we would have no method to read back the state and see what was broken. The slow path solves that, the flags(EF* and FF*) are all latched on the edge of the **CDF_CLK**. Therefore, if an error occurs because an EF* is acting up, the data is persistant and can be read out via VME. The data mapping for reading is shown in table 5. The mapping is pretty straightforward, except for the last register. It contains 4 EF*s and the results of the **efmacro**s and **/ERROR**. The mapping for writing is shown in table 4. The 8th of the **AL** registers packs in the data for the **Track** FIFO. Note: there is one bit that can not be read out, the **/TRFF**. It can be inferred from the state of the final **/ERROR** signal and the other individual FF*s. This was omitted because I ran out of data space and did not want to extend the space for 1 bit.

6.2 The Pace Maker

On schematic sheet 6, right next to the heart is the **Bunch_Counter**, Dirac's pacemaker. Sheet 7 contains the **Bunch_Counter** logic. On the right side of the sheet is the standard

Table 5: Read Mapping of the **ERROR** Data

Register	Mapping
L1 Full Flags	/ALFF[0],/L1FF[6..0]
AL Full Flags	/L1EF[1..0],/ALFF[6..1]
L1 Empty Flags	/ALEF[2..0],/L1EF[6..2]
AL Empty Flags	/TREF, /ERROR, AL /EMPTY, L1 /EMPTY,/ALEF[6..3]

DAQ L1 FIFO and L2 Buffer for the **BUNCH** data. **BUNCH** counts the number of clocks from an interaction for the event. As data is clocked into the L1 FIFOs around Dirac, the **BUNCH** chip is also clocking data into its L1 FIFO. In this way, if there is a clock distribution problem anywhere in the system, we can quickly diagnose and solve the problem. In TRIGMON, the online trigger monitoring software, the data from all trigger boards will be checked to insure that they all have the same bunch number. Pretty simple!

The schematic for the **BUNCH** chip is called BC.GDF. In essence, the **BUNCH** chip is a counter with a shift register as the on/off switch. The counter is located on the far right of the schematic and the shift register on the far left. The shift register is reset by the a **_B0** signal. The **_B0** pulse indicates the occurance of a special bunch, logically called number 0. Equally logically, we reset the counter on this pulse, suitably delayed by our depth in the pipeline. The SR is clocked by the L1 FIFO WCLK signal, **L1SI**. The **_RESET** signal pre–sets the SR to prepare it for a **_B0**. Only 8 taps from the SR are used, the others are just place holders. So prior to actual data taking, the approximate postion of Dirac in the pipeline must be known, 8 clocks is a lot of lee–way. **BUNCH** has a 3–bit register called **Bunch Offset** in the memory map. This register picks which of the 8 SR taps are used to clear the counter. The 8–1 multiplexer in the center of the schematic has the 3–bits from the register as an address and the 8 SR taps from which to pick. Watch the next step, it's a lu–lu.

The logic to actually clear the counter is difficult. During a startup sequence, a **_RESET** must come while **_halt** is asserted. On **BUNCH**, these signals are ORed to clear a flip–flop. When the system starts running the flip–flop(FF) is initally cleared. The D input is GND and the clock is the output of the SR. Thus, when the correct tap is turned on, the FF will be clocked to GND. The key is the pre–set for the FF. This includes a little feedback logic. First, on a proper startup sequence reset, another FF is pre–set. The output of this FF drives the pre–set of the initial FF. So at startup, the first FF is low and the second is high. The output of the first FF feeds back to pre–set the second FF if the first one goes high. If the SR is low, then the clock to the second FF is blocked. The transition of the SR from low to high triggers the actual reset of the counter. The first effect is to clock the first FF which resets the counter. There is a slight danger at this point; if a **L1SI** rising edge occurs during the transition of the first FF and clearing of the counter, then a race condition occurs. This could cause a failure to reset the counter. Be aware. Now the rising edge of the next **L1SI** will turn on the first FF, allowing the counter to start. The falling edge of the same **L1SI** counts to 1. Read technical note 6. And we are off to the races.

As per CDF specification, the first word of DAQ readout contains the bunch counter number, board ID and serial numbers.[12] In the lower middle of schematic sheet 6 are a series of jumpers that set the board ID [6] (S2) and serial numbers (S1). These are placed on the board during stuffing of the components. The actual selection of these bits during DAQ readout is part of the memory map. A reported [1] feature of the switches S1 is apparent on the schematics: the switch counting goes as 1 2 3 4 5 6 9 7 8 10, instead of 1 2 3 4 5 6 7 8 9 10. Therefore: to set bit 6 (staring counting at 0) must set switch number 9 (starting counting at 1). To set bit 7, must set switch number 7; to set bit 8, switch number 8.

6.3 Bells and Whistles

There are just a few more assorted things to mention. In the upper middle of schematic sheet 6 is the logic to receive the CDF specific signals. The chip on the left is a PECL differential receiver; it receives the **CDF_CLK** and passes the single ended version on to Dirac. The chip on the right is a fast flip–flop. It clocks the other CDF signals(labeled on the schematic) using the rising edge of the **CDF_CLK**. The **CDF_CLK** is guaranteed to arrive in the crate 10ns before the other CDF signals. The CDF signals are valid for only 10ns after the **CDF_CLK**. To insure that these signals are valid for the entire **CDF_CLK** period, they are latched and will not change until the next rising edge of **CDF_CLK**.

The ability to reprogram many of the EPLDs on Dirac is built in through the Altera Bit–Blaster (also Byte-Blaster) interface.[13] This is discussed in the general document pretty completely.[1] Schematic sheet 23 contains the actual connector for Bit–Blaster compatability. There is nothing new to add here.

Every board needs lights on the front panel. Dirac is no exception. **Front_Panel** contains Dirac's LEDs on schematic 19. Table 6 lists the lights. If you look at a dirac in the crate then the left column of table 6 are the lights on the left. The power light is on if Dirac is getting +5V. Clock is on if the Heart is driving a clock to its timing taps. It can be either the **CDF_CLK** or the VME **SYSCLK**. The error light echos the error register in the Brain. L1 decision is the L1 FIFO WCLK. Halt* is the **CDF_HALT*** signal after being latched off the backplane. Run and load are mutually exclusive. They indicate what mode dirac is in based on the state of the **LOAD** bit of **Control_0**. If both these lights are on, something is rotten in Denmark. Finally, and I am as relieved as you are, the VME Active light indicates that Dirac is responding to a VME read or write cycle.

[6] Dirac board ID is 005 (decimal)

Table 6: Dirac Lights

Left	Right
Power(+5V)	Clock
Error	L1 Decision
Halt*	RUN
VME Active	LOAD

7 SURGERY

7.1 Reprogramming: Try not to kill your patient

As you probably know by now, in the front-panel of every Dirac there is an input to insert an ALTERA's Bit Blaster or Byte Blaster (the last one is better as it is MUCH faster) so as to reprogramm the daisy chain of 15 reprogrammable Altera chips.

In case you are forced to, either because you want to get a PhD, or you need to avoid starvation, you might need to reprogram a Dirac board. This section gives you some guidelines, that if obeyed, will increase your probability of success from 0% to about 1 or 2%.

The first thing you need to know is that reprogramming a board can seriously damage it. And it did happened several times, on several boards. How ?

- Bugs in Altera code. Therefore *ALWAYS* use Altera version 9.21.
- Programming a chip with the wrong code, namely wrong pin assignment. If the board is routed in such a way that expects a given pin to sink current, and instead it's giving you current, or even worst if the routing expects something to be a ground, and when you recompile the code, you made it to be Vcc.
- Programming chip A with the (right) code for chip B. Therefore you must respect the *ORDER* of the chips in the chain. See below.
- Programming a given chip (or even a full chain) several (3 or 4 times) in a row, consecutevely. Therefore between each reprogramming, always allow at least 5 minutes. Reprogramming heats up the components, and possibly it also takes some time for the charges inside the chips's P-N gates to be properly localized.

You will find all the Altera code at
http://fozzie.uchicago.edu/cdf/level1/dirac/dirac.html
as a tar and zipped [7] file dirac_Altera_code.tar.gz. Get it, unzip and untar it, and put it in an Windows NT. This was the platform used to develop and program all these

[7] This was tared and zipped in a UNIX hp, so use Unix or Linux first to untar and unzip, not Windows.

boards, so if you try something else, you're on your own. *Use Altera V9.21* ! Over the years, using the old 9.21 might be difficult, but again, I wish you good luck if you use other versions.

In your directory, you should have now a directory called "copied to test slew". Why this name? Because. Inside this directory, you should find several other directories, each one corresponding to one different kind of chip, as described before. Namely:

- Bunch - This directory contains the Bunch counter code, chip U22. Program is bunch.gdf
- Control - This directory is now used ONLY to contain the .jcf files that allow you to program the all chain, or given chips in the chain.
- Csum - This directory contains the Csum programs in U180-U183. Program is csum311.gdf. These chips CANNOT be reprogrammed, since it is NOT in the chain. (The chips themselves are reprogrammable, though).
- Csum_mux - This directory contains the CSUM_MUX code, chip U21. Program is csum_mux.gdf
- Error - This directory contains the Error code, chip U20. Program is error.gdf
- Etout - This directory contains the EtOut code, chip U32. Program is etout.gdf. This chip CANNOT be reprogrammed, since it is NOT in the chain. (The chips themselves are reprogrammable, though).
- Heart - This directory contains the Heart code, chip U23. Program is heart.gdf
- Inp - This directory contains the EtInp code, chips U31, U33, U34. Program is bigadder.gdf (not bigadderpedro.gdf).
- NewControl3 - This directory contains the Brain code, chip U19. Program is brain.tdf
- TowMask - This directory contains the TowMask code, chips U25-U30. Program is towmask.gdf
- VME_interface - This directory contains the Heart code, chip U23. Program is VME_interface.gdf

Now if you want to reprogram a board with the code already compiled, then:

1. Run Altera 9.21.
2. File-> Open-> newcontrol3-> control.tdf
3. File-> Project-> SetProject to Current File

4. Max+plusII-> Programmer

5. Go to Options-> Hardware Setup, and check you have the right hardware setup

6. Go to JTAG and check Multi-Device JTAG chain

7. Click on Multi-Device JTAG chain, on the Programmer Menu.

8. Hit Restore jcf. Then navegate until you find

   copied to test slew/control/full_download.jcf. Hit OK.

9. Now you should have the list of programming File Names with the devices. Most probably you won't have a directory called e:/users/cdf/toolatedirac/dirac so you will need to change that according to your path. So you select one by one, edit the programming File Name, hit change, then go to the next programming file name.

   The list MUST be /inp/bigadder.pof, /inp/bigadder.pof, /inp/bigadder.pof, /csum_mux/csum_mux.pof, /towmask/towmask.pof, /towmask/towmask.pof, /towmask/towmask.pof, /towmask/towmask.pof, /towmask/towmask.pof,/towmask/towmask.pof, /newcontrol3/control.pof, /error/error.pof, /bunch/bunch.pof, /heart/heart.pof, /vme_interface/vme_interface.pof.

   At the end Save JCF... This is a very important step, as if you get the order wrong, or the bad code for one given chip, then you can kiss your board goodbye. See now why I didn't want to change the name of the top directory from "copied to test slew" to something more sensible, like "will I ever graduate" ?...

10. Put your byte blaster in the board and detect the chain. The board must be powered ON, of course. You should detect 15 devices. If not, you're already in somewhat BIG problems... Turn the power off, and run to the nearest pub...

11. Hit OK on the Multi-Device JTAG Chain Setup menu.

12. Go to Options->Programming Options, and take your pick. I suggest NOT using any of them.

13. If you can, plan in advance to monitor the current drawn by the board and check the temperature (with your fingies) of the chips.

14. Then hit Program on the Programmer menu.

15. If and when it fails a given chip, power off IMMEDIATELY !!! If you're very, very lucky your chips might still be alive...

16. If all goes well, congratulations !!!

What about if someone points a gun at you and makes you to do a change in the code itself ? If so, this tells you that you're probably in the wrong HEP collaboration. You should know at least that when you recompile, you absolutely MUST:

- Before anything else, make 10 copies of the original .rpt file for the chip you are about to destroy, err... reprogram...
- Assign->Back Annotate Project->Chip, pin & Device
- Should also compile with 9.21
- Check the pin assignments in the new .rpt file with the previous one. If they differ, well done: you just put yourself into a LOT of trouble !!!

THE END

APPENDIX

A Glossary of Terms

AUX Auxiliary Card. A card that plugs into the back of the crate. Usually used to receive signals from another crate.

TT Tracking and Threshold. Refers to the Decision section of Dirac where tracking and calorimetry thresholds are applied to make a L1 decision.

EtInp Transverse Energy summing Input Chips.

EtOut Transverse Energy summing Output Chip.

CSUM MUX Trigger Summary(CardSum) Multiplexer Chip.

CSUM, Csum Trigger Summary. Part of Dirac that performs the summary of the trigger decision for the board.

SRAM Static Random Access Memory. A type of Memory chip.

EF* FIFO Empty Flag. Inverse logic signal telling whether a FIFO is empty.

FF* FIFO Full Flag. Inverse logic signal telling whether a FIFO is full.

VRC VME Readout Controller. The master in every crate that control VME access to all cards.

TSI Trigger Supervisor Interface. The controller and coordinator of the trigger.

SR Shift Register.

DFF D type Flip Flop.

DFFE D type Flip Flop needing an enable.

B Technical Notes

1. You may notice that some of the internal nodes are NODE definitions while others are LCELL, DFF or DFFE. The DFF and DFFE are internal D flip-flops without and with an enable switch. LCELL is short for logic cell. This is a fundamental division in the architecture of an Altera EPLD. When a node is defined as an LCELL the logic used to generate the internal node is isolated from any other logic that it might feed. This has two effects. First, it preserves the existence of the signal which lets the user view that signal in a simulation. If the logic is "synthesized" away(I prefer to use the technical term, "logiced" away!) by the logic compiler, then the node essentially disappears. This is done by using any number of Logic theorems, such as DeMorgan's Rule. Second, it reduces the amount of feedback logic in some cases. The compiler is usually smarter than you, MUCH smarter in fact, so it is usually good to let it do what it wants. However, sometimes the compiler will try to feed TOO many signals to a single logic cell.(the architecture limits the number of Fan–In signals) The LCELL node allows you(the user) to combine logic into little blocks and then use the resulting blocks to form the final answer. Thus reducing the total number of signals feeding any given logic. This is a complicated concept, but is extremely valuable if there are problems in compiling these programs.

2. You might think, just use a couple more data lines from VME to the TOWLAT chip and the data bit packing becomes much more simple. That was how the chip was originally designed and it was much simpler. We then decided to add ISP to Dirac. This requires 4 pins from an Altera EPLD. Unfortunately, there were not 4 free pins on the TOWLAT, so the solution was to multiplex the data even further and free up a few pins. . .which brings you to the present.

3. Ok, why not 200 Ohm resistors? The 100 Ohm resistors were much easier to find than 200 Ohm. They come in a 16 pin package with 8 resistors per chip, so there is one resistor pack per receiver.

4. The consequences are notable. The FIFO has a 2 clock latency before data appears on the outputs. Therefore the fact that both paths have FIFOs will cause, at a minimum, a 2 FIFO clock delay. If one path had a latch, the delay would be only the fall–through time of the latch. The clock on the FIFO can easily run at 25ns, so there is at least a 50ns delay.

5. We decided to treat every **CDF_CLK** as a real interaction. This implies that the TSI generates a **CDF_L1R** for all crossings that do not contain an actual interaction. There are 2 occasions when this is true. First, we are running with collisions every 396ns.(this is the initial plan) In this case only every 3rd **CDF_CLK** actually contains a collision. Second, there are abort gaps in the beam structure.(So named because the accelerator physicist like to have a nice region where they can abort the beam without spraying us particle physicists with our lifetime radiation

dose! Bless them. Either that or government safety regulations. Hmm.) During the abort gaps some systems will perform calibrations, but the TSI will send **CDF_L1R** to all other systems.

6. What happened to bunch number 0? Well the counter gets reset, right? At the next rising edge of **L1SI**(the WCLK for the L1 FIFOs), 0 is written into the FIFO. This is the same time that the first FF is being turned on to allow the counter to start. Then on the falling edge the counter increments. The next rising edge clocks in 1 and so forth. Now it is clear why the falling edge of **L1SI** is used to increment the counter.

References

[1] Wahl, John E. and Amaral, Pedro *Dirac — The FINAL Specificaton.* CDF/DOC/TRIGGER/CDFR/4225.

[2] Shaw, Teresa M. *Specification for ADC/Memory Module.* Particle Physics Division. CDF Upgrade Project. 6/11/97.

[3] Freeman, Jim, et al. *The Run II eXtrememly Fast Tracker (XFT)* http://cdfsga.fnal.gov/upgrades/daq_trig/xft/xft_home.html.

[4] Toback, Dave. *Crate Sum - Upgrade for Run II.* CDF Trigger Note 91.

[5] The P3 Auxilliary(AUX) card was designed and built by Peter Wilson.

[6] Sarah Truitt is designing the Digital Compare and Sum (DCAS) boards for the Level 2 trigger.

[7] CDF Group, Yale University. *Trigger Supervisor Protocols for Run II* August 18, 1996, Version 1.2.

[8] CDF DAQ Group. *The CDF DAQ Manager's Handbook.* August 18, 1996, Draft #3.

[9] Wahl, John and Wilson, Peter J. *Digital Receive And Compare (DIRAC) Board Startup Timing Diagram.* June 11, 1995, CDF Trigger Note 95.

[10] Peterson, Wade D. *The VMEbus Handbook.* Third Edition. VITA. Scottsdale, AZ, 1993.

[11] VIPA *VME-P Document.* NIM/VME-P 9612.

[12] Ohl, K, Wahl, J, and Wilson, P. *Specification of DAQ/Trigger Synchronization Checking Protocol and VME board Header word.* Version 2.0. CDF/DOC/TRIGGER/CDFR/3145.

[13] Altera Corporation. *Bit Blaster Serial Download Cable Data Sheet.* Altera Corporation Product Data Sheet.