JOINT EUROPEAN X-RAY MONITOR
JEM-X

SOFTWARE SPECIFICATION DOCUMENT FOR THE DFEE ON-BOARD-SOFTWARE

IN-SD-JEM-0001

Issue 2, Rev. 0

Prepared by: Ib Lundgaard Rasmussen Date: December 3, 2001

Reviewed by: _____ Date:_____
Søren Brandt
Test Manager

Approved by: _____ Date:_____
Kurt Omø
Project Manager

Approved by: _____ Date:_____
Niels Lund
Principal Investigator

Danish Space Research Institute
Juliane Maries Vej 30
DK-2100 Copenhagen
+45 3532 5830

REVISION NOTICE

Document Revision History		
Revision	**Date**	**Changes**
Issue 1, Rev 0	September 25, 1999	Initial Release
Issue 2, Rev 0	December 3, 2001	General Revision

1. Introduction

1.1 Purpose of the Documentation

This document describes the analysis and design of the DFEE On-Board Software. It is not a historically correct description of the software development process. As it is often the case when developing software for a set of hardware under concurrent development, the requirements and specifications as well as the environment undergo changes throughout the software development period. New requirements appear even in the middle of the coding and testing of the software. This evolution was especially pronounced in the event analysis segment of the software where the complicated correlations of the data needed the presense of the actual detector to allow the final analysis of the position determination algorithme.

This document will however present the software in the framework of a standard top down structured analysis and design.

1.2 Definitions, Acronyms and Abbrivations

ADC	Analog to Digital Converter
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DFD	Data Flow Diagrams
DFEE	Digital Front End Electronics
DPE	Data Processing Electronics
FIFO	First In First Out Memory
FPGA	Field Programmable Gate Array
HK	House Keeping
HSL	High Speed Line
HV	High Voltage
HW	HardWare
JEM-X	Joint European X-ray Monitor
LSL	Low Speed Line
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
OBT	On Board Time
RAM	Random Access Memory
ROM	Read Only Memory
USART	Universal Synchronous Asynchronous Receiver Transmitter

1.3 References

[1] ESA, BSSC(96)2 Issue 1, May 1996, Guide to applying the ESA software engineering standards to small software projects

[2] ESA, PSS-05-0 Issue 1, January 1987, ESA software engineering standards

[3] ESA, Issue 1, Rev. 7, September 1999, Integral EID-A

[4] DSRI, Issue 1, Revision 0, February 1996, On-Board Software User Requirements

[5] DSRI, Issue 1, Revision 0, January 1997, Software requirements Document (On-Board Software)

[6] Ward and Mellor, Structured Development for Real-Time Systems, Yourdon Press, 1985 and 1986

[7] T. De Marco, Structured Analysis and System Specification, Yourdon Press, 1978

[8] G. J. Meyer, The Art of Software Testing, Wiley 1979

[9] MIL-STD-1750A(USAF)

[10] JEM-X CPU Board function description and register definition IN-DD-JEM-0001, November 1999

1.4 Overview of the documentation

2. Model Description

The software will be described using a Yourdon Data Flow Diagram method based upon the books by Ward and Mellor [6].

The logical model of the program is shown in a set of Data Flow Diagrams (DFDs) [7]. The following remarks are intended to guide the reader through the diagrams.

The DFEE software provides the interface between the JEM-X hardware and the DPE. All interfaces to the outside world is handled through the FPGA unit. All the low level interface routines are placed in the FPGA. This means that all input and output operations are performed by reading and writing to FPGA ports [10].

The program will be loaded from the ROM memory and have all interactions through the FPGA as illustrated in figure 1.

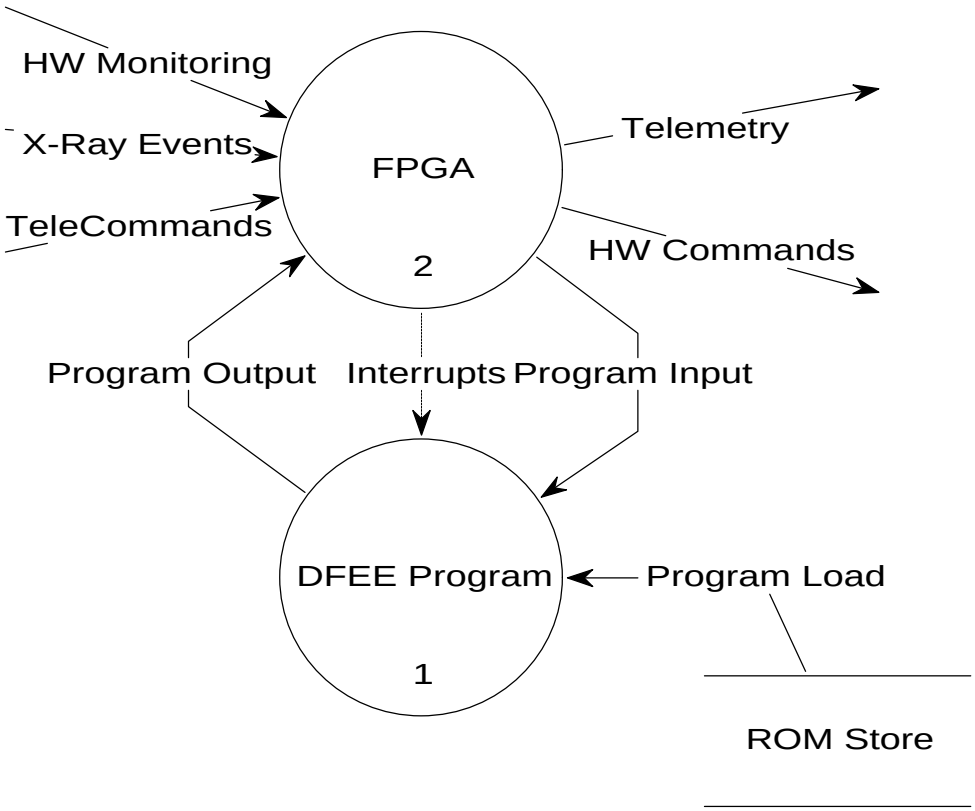

Figure 2.1. DFEE Software overview.

2.1 Overall design considerations

The main purpose of including a CPU in the JEM-X instrument comes from the fact, that the amount of X-ray data generated by the instrument exceeds by orders of magnitude the telemetry capacity provided for JEM-X by the INTEGRAL satellite. This fact has been the driving design consideration.

X-ray events arrieve randomly in time. This points to a interrupt driven design of the software. The given design of the DFEE-DPE interface (a half-duplex protocol) also points in this direction.

It was thus decided to create the program around an Idle loop. This loop check a sequence of flags that signal if an event has occurred. Events are here to be taken in the programming sense with X-ray event being only one type of events. The occurrance of an event generates an interrupt. Each interrupt is handled by a dedicated interrupt handler. The handler collects all volatile data, stores it and sets the corresponding interrupt flag. When the flag is detected further processing is performed and the flag is cleared.

This design philosophy has the advantage that the different segments of the program are almost completely independent. This has been a great advantage in the development process as it is an enormous help in locating errors in the software. The independence means that errors introduced by new code harly ever propagate to other parts of the software.

Having a CPU closely associated with the JEM-X hardware led to the inclusion of a lot of other functions.

The main functions of the software are to:

Control the Hardware
Receive messages from and transmit messages to the DPE via the Low Speed Line
Get X-ray event data from the Hardware
Determine the x-ray position in the detector
Transmit X-ray event data to the DPE via the High Speed Line
Collect X-ray calibration spectra
Collect House Keeping Data

The main loop is structured as a skip chain checking of event flags. If no event flags are set the main loop has only two activities: To reset the Watchdog function and to check that the X-ray event rate is below the HV Cutoff limit.. If an event flag is present a specific set of activities is performed. The event flags are set by the interrupt handlers. After the required activities are performed the flags are cleared . This means that each function of the software is checked once for each passage of the Idle Loop.

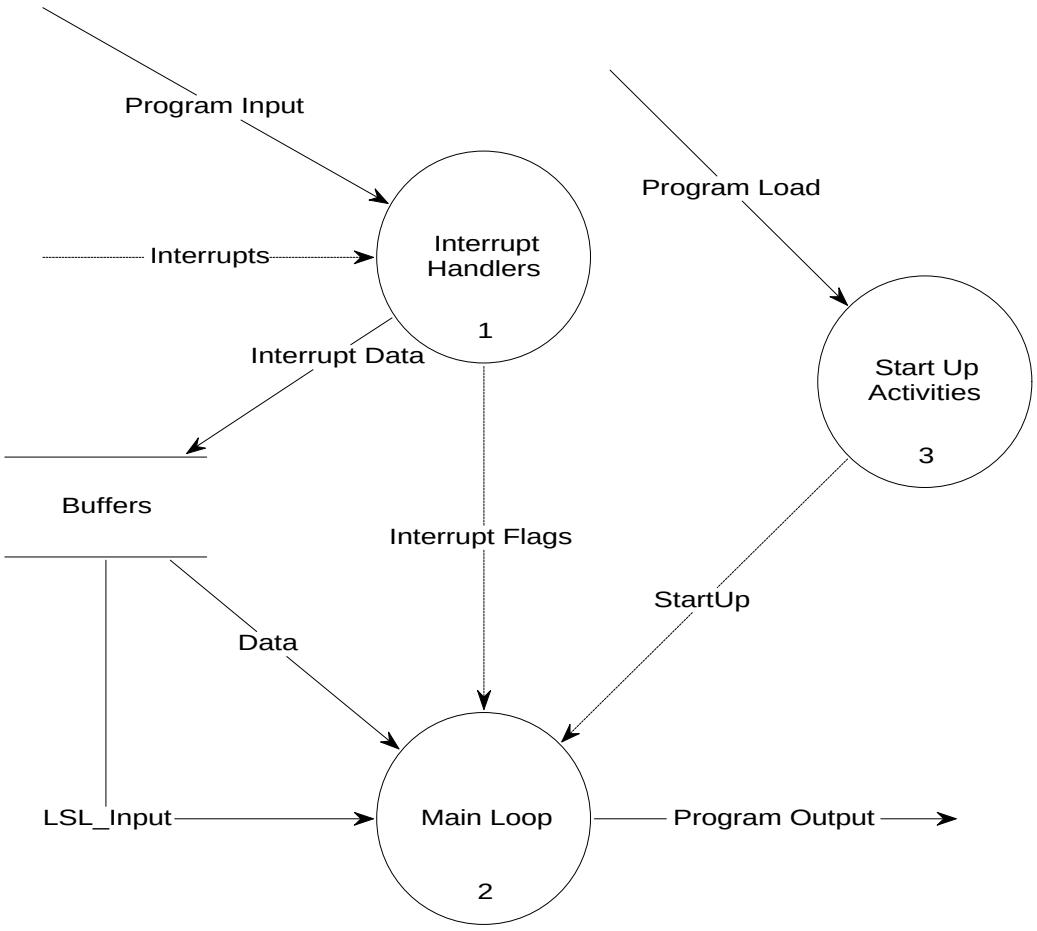

Figure 2.2. Overall structure of the DFEE software

The interrupts will each activate an interrupt handler. The interrupt handlers are:

USART Interrupt
OBT Interrupt
X-ray Event Interupt
Timer A Interrupt
Timer B Interrupt
High Voltage Interrupt

While an interrupt handler is active the interrupt system is disabled. Therefore the handlers are designed to be as fast as possible. In the case of an X-ray event interrupt all the data needs to be read before the next X-ray event is allowed to enter. A main consideration in the design of this handler has been to ensure that no other interrupt becomes too old during this activity. This was especially important with respect to the USART operations as these are controlled by the Low Speed Line Clock.

The initialization segment of the program will setup the interrupt system and load the variables used to control the operations. A large part of these variables will be loaded from the DPE. The initialization sequence also activates a basic set of the interrupts and transfers the control to the main loop.

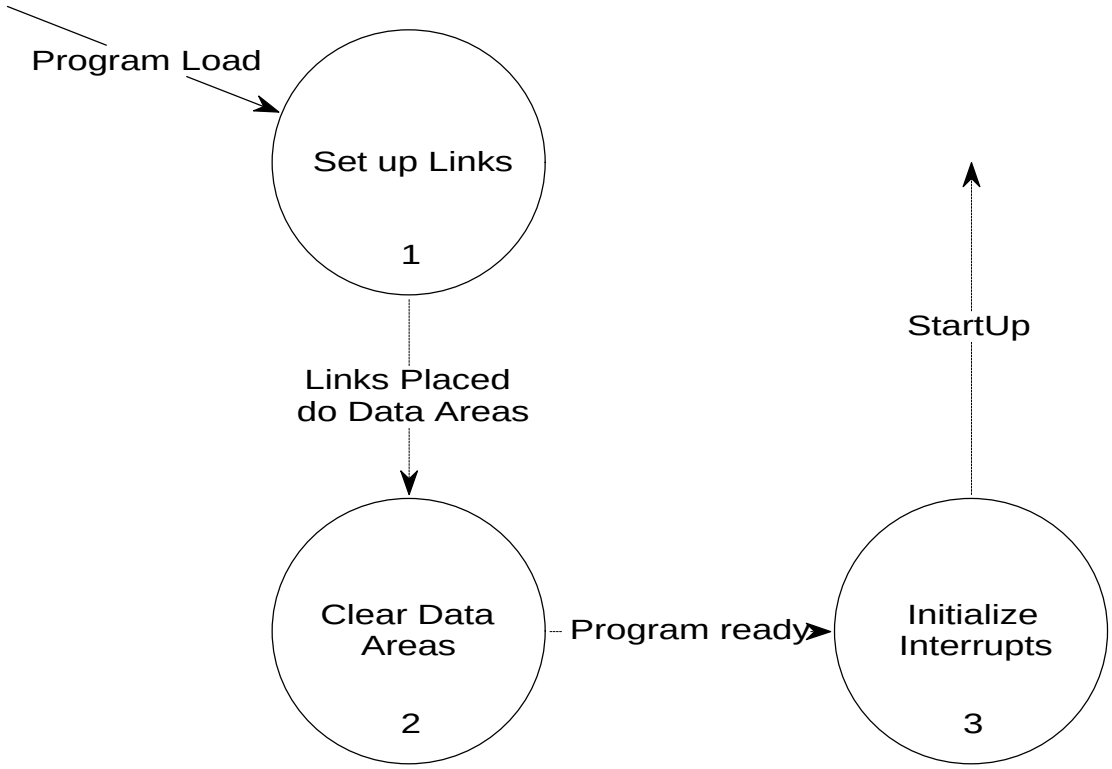

Figure 2.3. Startup Activities

When the main segment of the program is activated it executes the IDLE LOOP. This is a skip chain of flag inspections. Each flag signals if an activity needs to be executed. The flags are tested in the following sequence:

USART_INPUT_FLAG
USART_OUTPUT_FLAG
INPUT_READY_FLAG
VALID_MESSAGE_FLAG
X-RAY_EVENT_FLAG
FIFO_FLUSH_FLAG
TIMER_A_FLAG
TIMER_B_FLAG
EXTERNAL_HV_OFF_FLAG

If one of the event flags in the chain is set the corresponding activity is performed. At the end of each passage through the loop two activities are performed: The WatchDog is resat and the number of X-ray events collected in the present 8 second period is compared to a limit and if it is too high the HV will be switched off and the DFEE will be placed in the SAFE state.

2.2 Event processing

The main activity of the program with respect to processing time is the processing of the X-ray events. Each event has to be read, analysed and if accepted transmitted to the DPE on the High Speed Line. The event data will be the only load on the HSL.

The number of events detected by the instrument is much larger than the number of events that can be transmitted. To reduce the amount of event data is the main reason for the DFEE CPU. This reduction can occur in two different ways:

The event can be rejected
The event can be discarded

The event can be rejected if it fails one of the rejection criteria.This process requires a lot of calculations. The rejection criteria are designed to remove background particle events and only let real X-ray events get through. In normal operating conditions we expect to have approx. 2000 particle events/second and between 10 to 1000 good events. The higher number occurring when observing a strong source or during a burst period. Most sources will yield below 100 events/second. This illustrayes the importance of the rejection. Even acceptance of just 5% of the particle events will more than double the transmission load. The rejection will be described in more detail later.

The events can be discarded when the system accepts more events than it can handle. These throughput problems can occur for a number of reasons:

The DPE can not transmit the number of accepted events
The HSL can not transmit all accepted events
The DFEE can not process all the detected events

The various causes of throughput problems are solved in different ways. If the DPE can not handle all event a Grey Filter mechanisme is activated. This discards a selected percentage of the incoming events. If HSL problems or processing problem occur all events are rejected until the problem is removed.

The fact that throughput problem is a major design consideration has lead to adoption of the following design consideration:

> To process a large amount of X-ray events it is important to spend as little time as possible on events that will later be rejected. This means that particle background events must be identified rapidly and that events that will anyway be rejected due to lack of TM capacity (by the Grey Filter mechanism) should be eliminated without being read.

The second method of reducing the transmission load is to perform a omboard analysis of each X-ray event. From the 34 words of 16 bit each the data is reduced to 4 16 bit words. These 64 bits can be further reduced in the DPE thus ensuring that we always get some

information about the target under observation.

If the event is not discarded before the reading of the event data, the data is read one analog board at a time, each board containing 8 data channels. The second board read contains the anode data and the HV monitoring. Simple checks on the anode data can lead to the rejection of the event. This mechanism mainly reduces the amount of particle background events brought forward for detailed analysis. The tests performed are:

Check Fast and Slow anode for overflow or underflow
Check Slow anode < maimum limit
Check that 2*Fast > Slow anode

If any of these tests fails the event is rejected and the event data are discarded.

If the tests are passed the rest of the event data are read and the full event data stored in a slot in a ring buffer. This is the end of the event interrupt handling. The events are later extracted from this buffer and analysed in detail.

During this event analysis several other rejection tests are performed:

Ratio of Slow/Fast Anode signals in range
Ratio of Slow Anode/Veto signals in range
Ratio of Slow Anode/Total Backplane signals in range
Ratio of Slow Anode/Total Cathode signals in range

Some of these tests require detailed corrections of the data before application.

All the different ways of either rejecting or discarding an event are tallied in a set of counters reproted in the HK data. The counters should allow the check on the correct performance of the event analysis:

#triggers = #good events + #discarded events + #rejected events + #calibration events.

The result of the event analysis is:

Event identifier	16bits	
Event arrival time	32bits	1/8192 sec resolution
Event energy	12bits	
Event position		
x position	12bits	1/4 mm resolution
y position	12bits	1/4 mm resolution

the 12 bit values are stored in a 16bit word so each event requires 6 words transmitted to the DPE. This is done via a FIFO.

One type of events are accepted but not transmitted on the HSL this are the calibration

events.There are 4 calibration sources placed in the detector collimator. The events from these sources are identified from their arrival position in the detector. The energy of these events are used to generate spectra that are transmitted in the HK block, one spectrum for each source. The spectra are collected in a double buffer system, one spectrum is collected while the previous is transmitted. A new set of calibration spectra is transmitted every 256 seconds.

2.3 Hardware Control

The DFEE software controls the JEM-X hardware configuration. The following aspects of the configuration can be controlled:

Anode configuration
Low level discriminator
High Voltage

The active area of the JEM-X detector is divided into 4 anode sections, 11 cathode strips in paralllel and 20 backplane strips perpendicular to the anodes. The anodes are used to trigger the instrument as well as measuring the energy of the x-rays. The anode configuration controls the inclusion or exclusion of each individual anode section in the trigger condition. The low level discriminator is used to set a minimum level of the trigger pulse. The discriminator is set to exclude as much as possible of the electronic noise without cutting into the interesting x-ray events.

The most complicated hardware control function is the adjustment of the High Voltage. As the high voltage is the basis for the operation of the detector the detector can only be affected by the radiation in space when the HV is on. Therefore the turn on of the HV is considered a hazardous command. This command is therefore designed as a Arm-Fire sequence. The arming command is HVREADY. When this command is executed the instrument is placed in a special state where only the commands are accepted: The HK request and the HVON command. The instrument remains in this state until the HVON command is received or until 1 minute has passed. If a HVON command is not received before the timeout the Arm-Fire sequence has to be restarted.

Once the HV is on the two voltages are commanded to a minimum value corresponding to a setting of 12 Hex. This is done to ensure that the voltages are above the bias setting and thus under active program control.

Adjustment of the level of the HV is one of the more complicated operations the DFEE performs.This is caused by the fact that a direct hardware setting of the HV to full operating level due to an overshoot problem will cause an overvoltage with possible damage of the detector as a consequence. Therefore the program will generate a series of hardware commands that will slowly and stepwise approach the desired voltage. This means that adjustment of the HV is one of the two operations (the other being the Electronic Calibration) with a timed duration. Each of the functions is controlled by one of the two timers in the CPU, Calibration by TIMER A and HV by TIMER B.

3. Specific Requirements

3.1 Functional Requirements

3.2 Performance Requirements

3.3 Interface Requirements

3.3.1 The protocol on the Low Speed Line should be in the form of a request-response exchange.

3.3.2 All requests should come from the DPE.

3.3.3 When a request does not generate data as a response, the DFEE software should send a request acknowledge to terminate the transaction. This request acknowledge should contain an execution code giving the result of the request.

3.3.4 The DFEE software should handle the following request types:

State changing requests
Memory Patch and Dump requests
Parameter change requests
Hardware setting requests
Grey filter change requests
HK Data Request

3.3.5 The Requests shall have the format:

Request Code
Request parameters
CRC word.

The request code (in Hex) is:
The corresponding Answer Code (in Hex) is:

1234		Change State
	F00F	Request Acknowledge
1608		Set CPU speed
	F00F	Request Acknowledge
2323		DFEE Status
	0F0F	Status Block
2468		Change Grey Filter
	F00F	Request Acknowledge

4644		Load first part of Energy Table
	F00F	Request Acknowledge
4645		Load second part of Energy Table
	F00F	Request Acknowledge
4646		Create Energy Table
	F00F	Request Acknowledge
4711		Request HK Data Block
	7777	HK DataBlock
7F7F		Calculate Memory CRC
	F7F7	Memory CRC
8642		Change Software Integer Parameter
	F00F	Request Acknowledge
8765		Change Software Floating Point Parameter
	F00F	Request Acknowledge
AAAA	Change HW Setting	
	F00F	Request Acknowledge
ABCD	Load Memory	
	F00F	Request Acknowledge
BBBB		Dump Parameter Block
	BC01	Integer Block
	BCEF	Floating Point Block
CADB	Dump Memory	
	0FF0	Memory Dump
FDB9		Value of Integer Software Parameter
	1357	Parameter Value
FEDC		Value of Floating Point Software Parameter
	147A	Parameter Value

The Request Acknowledge has the form:

F00F
Code
CRC

The code has the following values:

0000	Operation OK
0001	CRC error in the request
0002	Unknown request
0003	Request not valid
0004	Requested parameter# not valid
0005	Requested parameter value not valid
0006	DFEE not ready

The DFEE not ready is a condition that occurs when a HV change is requested and the DFEE is still processing a previous HV request.

3.3.6 The High Speed Line shall be used only as a FIFO transfer line from DFEE to DPE.

3.3.7 When 4K words of data are ready to be transmitted a Half-Full-FIFO-Flag shall be raised on an analog line.

3.3.8 When data collection is terminated 4K of filler words are placed in the FIFO to ensure the Half-Full flag will be raised.

3.4 Operational Requirements

3.4.1 The modification of the software (i.e. uplinking of memory patches) shall only take place when the software is not activily accepting X-ray data.

3.5 Ressource Requirements

3.5.1 The DFEE software shall not occupy more than 8K in ROM.

3.6 Verification Requirements

3.6.1 The software shall be verified by extended test runs against the instrument hardware.

3.6.2 The proper function of the On-Board X-ray event analysis shall be verified through inspection of Diagnostic Dump data.

3.6.3 The proper function of the On-Board data compression scheme shall be verified through testing at high count rates.

3.7 Acceptance Testing Requirements

3.7.1 The Acceptance Testing (A T) will exercise all States of the DFEE Software.

3.7.2 The A T will verify the proper responce to all types of Low Speed Line messages.

3.7.3 LSL Messages with parameters should be tested in a set of representative cases.

3.8 Documentation Requirements

3.8.1 Functional aspects of the DFEE software impacting the use of the JEM-X instruments shall be documented in the user manual.

3.9 Security Requirements

This imbedded software has no security requirements. Protection against misuse rests upon the MOC security arrangements.

3.10 Portability Requirements

The software is not portable.

3.11 Quality Requirements

3.11.1 The analysis and design of the software shall use top down structured methods.

3.12 Reliability Requirements

3.12.1 The MTTF shall be more than one orbit.

3.12.2 The MTTR shall be less than one day.

3.13 Maintainability Requirements

3.13.1 The DFEE software shall be placed in ROM memory. When activated the software shall be moved from ROM to RAM memory prior to execution. A selftest shall be performed.

3.13.2 It shall be possible to modify the software by a memory patch mechanism.

3.13.3 It shall be possible to dump any part of the memory to ground.

3.14 Safety Requirements

3.14.1 A Watchdog function in the DFEE software shall ensure that the High Voltage is switched off in case of software failure.

3.14.2 An Arm-Fire command sequence should be used to switch on the High Voltage.

4. System Design

4.1 Design Method

The design of the DFEE software was heavily influenced by the given environment. The hardware platform was given, the outside interfaces defined, the size of the program constrained by the ROM requirement. The nature of the tasks to be performed by the software led to a decision to construct the software as a collection of event driven routines. The routines are only loosely connected. This means that the design chosen has reduced the task to the construction of a suite of small independent routines.

The events driving the software comes mostly from interrupts generated by the FPGA hardware. To each interrupt is associated a interrupt handler. This handler restores the interrupt, secures all volatile data and if necessary raises a flag indicating that the event needs further processing.

The central part of the software is a flag checking skip chain: The idle loop. If no flag is set the only activity is to reset the Watchdog function. If a flag is set the corresponding service routine is activated. At the end of the activity the flag is cleared.

This approach meant that it was possible to build and test a simple program consisting of the Idle Loop, the USART interrupt, its interrupt handler and the input and output service routines. The prototype only gave request acknowledge returns on all messages on the Low Speed Line. Then the State Changing function was added. After this part of the structure of the program was checked out the functionality of the States could be added.

The independence of the different routines meant that trouble shooting was localized, errors where always in the new routines. The addition of new routines did not create errors in already tested parts of the software.

4.2 Decomposition Description

The top-level diagram shows the initialization part of the DFEE software which is the first part to be executed after the software has been loaded from the ROM. After this initialization the software reaches the IDLE LOOP where it stays until an outside event requires attention.

The IDLE LOOP is a skip chain that checks a series of flags. If one of the flags has been set the corresponding activity will be executed. The flags are set in response to an interrupt which has been handled by one of the Interrupt handlers. These are shown detached from the flow of the program.

Once a flag has been set the program moves to the corresponding service routine:

Interrupt Analysis
LSL Message
Event Analysis
External HV OFF
FIFO flush

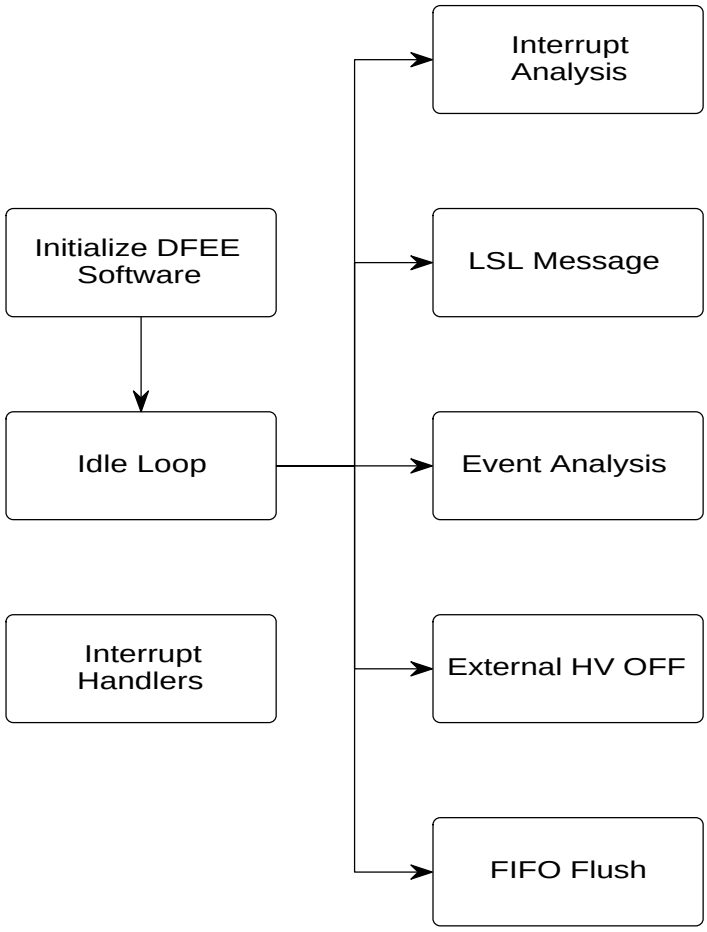

Fig 4.1 Top Level Diagram

The initialization processing is shown in figure 4.2. The clearing of the memory is now redundant as it turned out that it was better to clear all the memory before loading the DFEE software. This was needed in order to generate consistent CRC values after a restart of the DFEE.

An added reason for the retainement was also that the clearing was interspersed with the setup of the LSL where some milliseconds are needed between the setup commands.

The expansion of the rejection limits is performed because they could then be stored as one word integers in stead of two word floating point values.

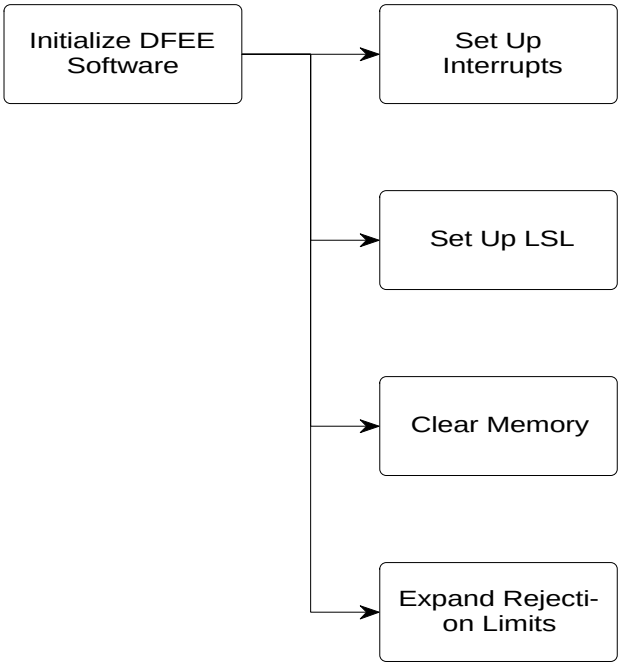

Fig. 4.2 Initialization

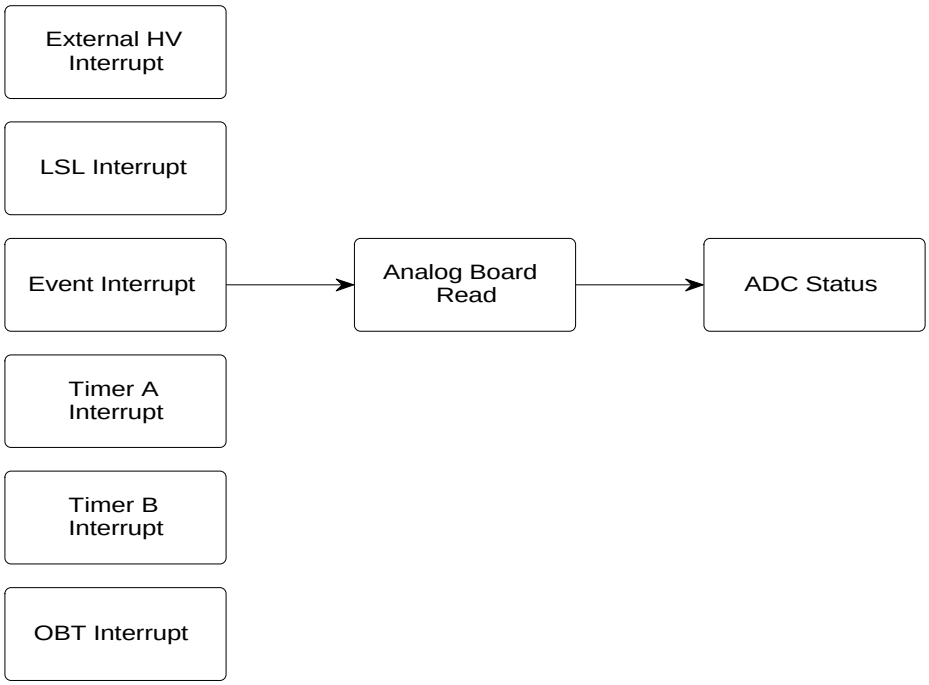

Fig. 4.3 Interrupt Handlers

The Interrupt handlers are shown in figure 4.3.

The handlers all have the same basic structure:

Save all volatile data
Reset the interrupt
Set the Interrupt Service flag

In the Event interrupt handler things are more complicated, it is necessary to check if the data can be collected or should be collected (refer to section 3.3). The 5 analog boards have to be read each containing 8 analog channels to be read by the ADCs. Then some control of the data is performed and the accepted data stored for further processing.

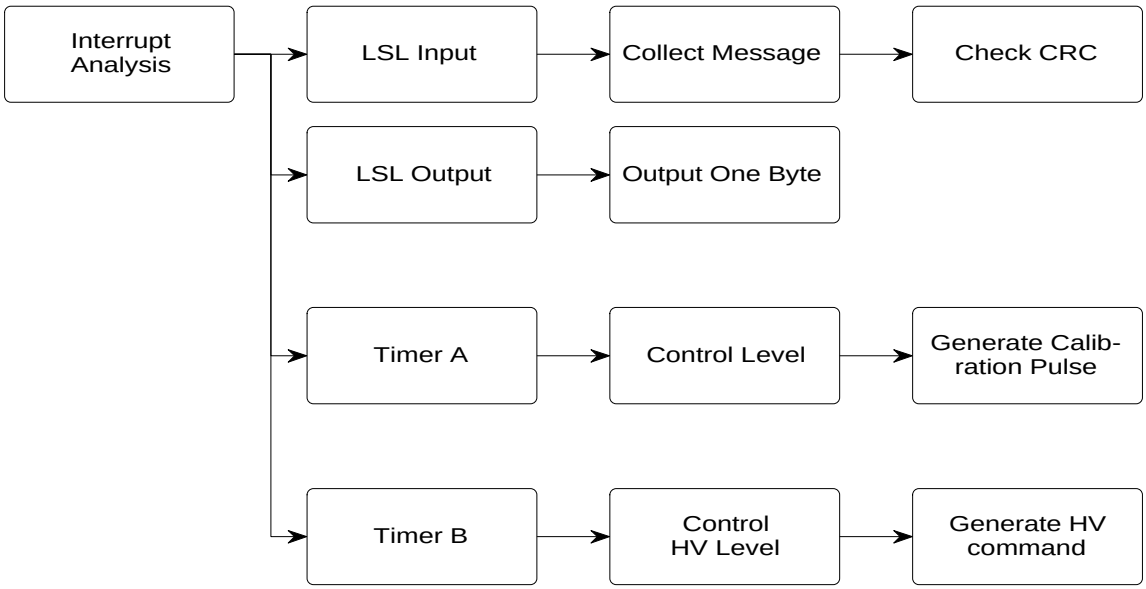

Fig. 4.4 Interrupt Analysis

Figure 4.4 shows the collection of the LSL messages. Each byte transmitted on the LSL generates an interrupt, it is therefore necessary to keep count of the output and to determine the end of the input to allow further processing. The transmission of data from the DPE to the DFEE is involving a start and a stop code around the message so the data will look as:

DDDD DDDD Message 3333 6666

with two hexwords before and after.

Once the messages has been collected they are analysed to determine the action requested.

The possibilities are shown in figure 4.5.

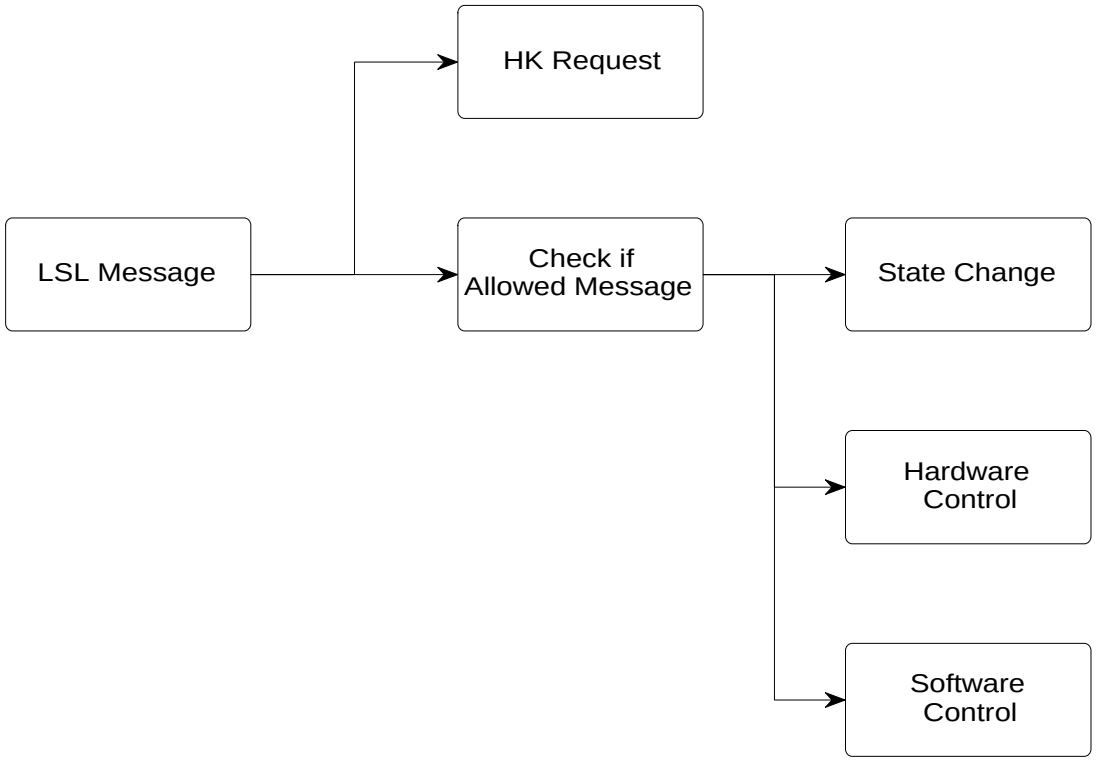

Fig. 4.5 LSL Messages

The first possibility to be checked is the HK request. There are two reasons for this:

The request is always allowed
It is the most frequent request

Actually most of the time it is the only input appearing on the LSL.

If the request is not for HK then it is checked to see if it is allowed in the current state of the software.

If the message is allowed it is then determined which of the three possible types it is.

The software control is shown in figure 4.6.

The software starts in the SAFE state. This ensures that the HV is off. From this state one can get to the MEMORY state in which possible corrections to the software can be uploaded. Upon leaving this state the expansion of the rejection limits is invoked. This process is carried out even if the rejection limits have not been modified.

Going from the SAFE state to the SETUP state allows the energy table to be loaded and expanded. It also allows the software parameters to be modified or dumped to the DPE. This is shown in figure 4.7. The final software modification takes place when the is changed from SETUP to the DATA TAKING state. This transition initialises the Grey filter to fully open. In the DATA TAKING state the grey filter can be modified to any of 31 states allowing a fraction of the event data to be rejected if the count rate is high.

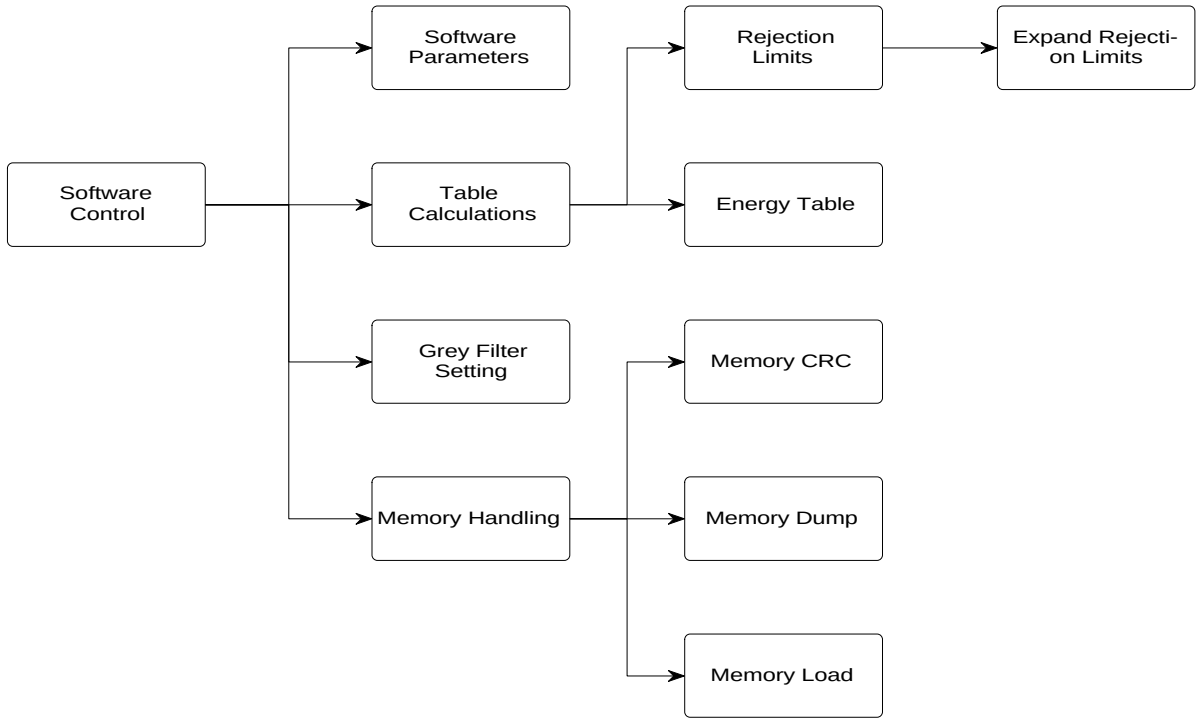

Fig. 4.6 Software Control

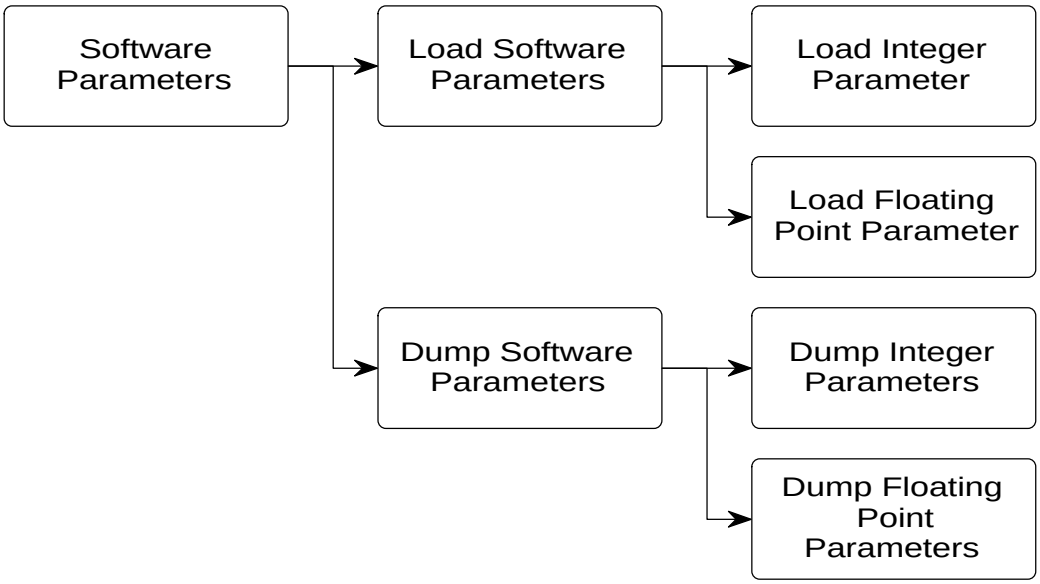

Fig. 4.7 Software Parameter Control

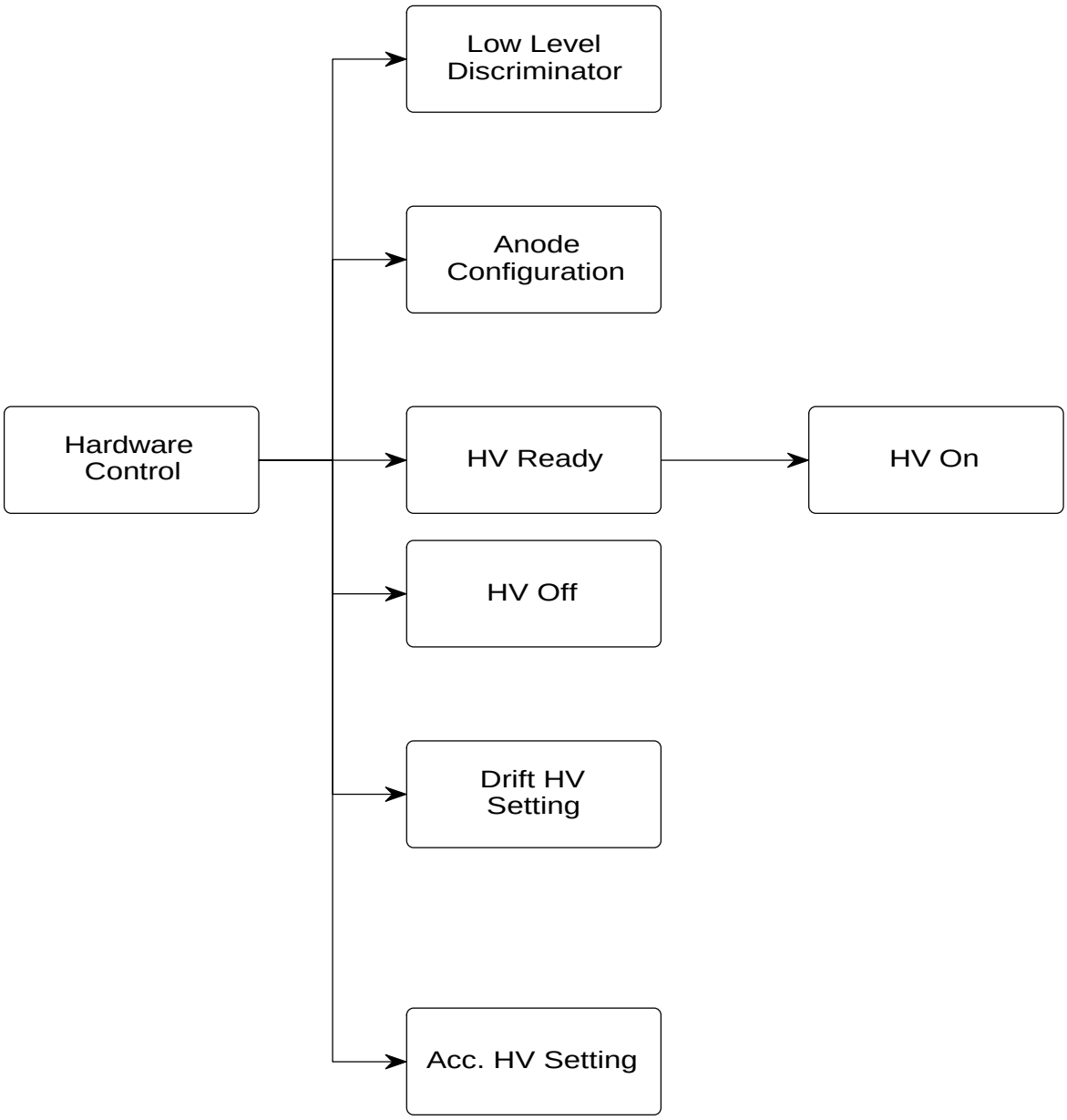

Fig 4.8 Hardware Control

Before starting the instrument for good one must also exercise the hardware control. This is shown in figure 4.8.

The hardware control takes care of the active anode segments, the low level discriminator and the High Voltage. The only complicated activity is the HV control. The switch on of the Hv is the only critical activity of the DFEE software. Therefore the HV is switched on by an arm-fire command sequence, this involves a special state HVREADY where the only accepted command is the HVON command. If this command is not executed less than one minute after the HV READY command then the sequence is terminated.

Changing the level of the HV requires the use of the timer B. This is necessary because the voltage has to be increased slowly to avoid overshoot damage to the detector.

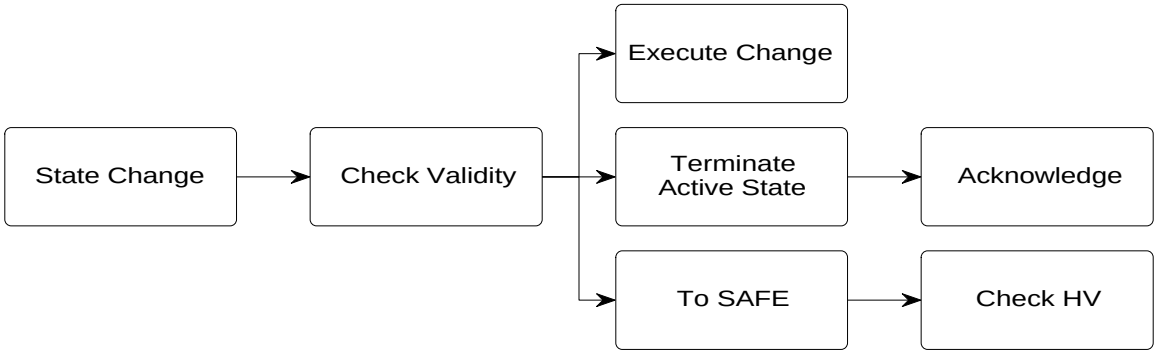

Fig. 4.9 State Changes

When the software and hardware has been properly configured the data taking operations can be initiated. The DFEE is commanded into one of the active states: DATA TAKING, CALIBRATION or DIAGNOSTIC DUMP. This activates a few special features of the state changing routine as shown in figure 4.9. With the HV on it is important that the instrument can reach a safe configuration quickly. If a GO TO SAFE command is issued the instrument will move directly from the active state to the SAFE state bypassing the SETUP state. Part of this state change is to ensure that the HV is off.

Two of the active states can be terminated either by command or automatically when a given number of events has been collected. In the latter case no acknowledge message should be generated as this would violate the LSL protocol requiering the DPE to initiate all transfers.

In the active states the event interrupt is active allowing x-ray events to be collected.

In the case of the CALIBRATION state the calibration pulses are generated using Timer A because the pulser needs time to regenerate the required voltage level (refer to figure 4.4).

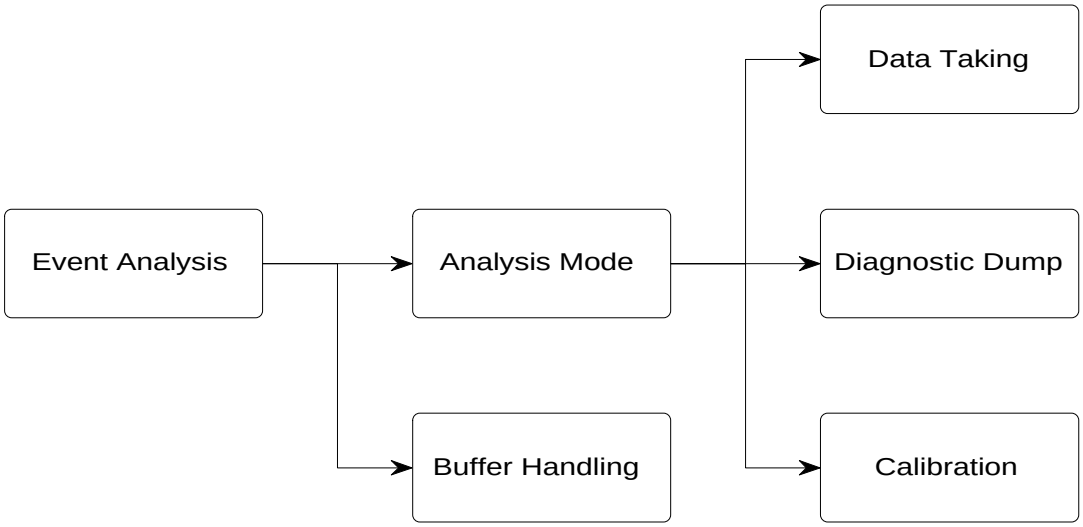

Fig. 4.10 Event Analysis.

When an event has been read as shown in figure 4.3 it is stored in a circular buffer of size 5 events and the event flag is set. The processing of the event is shown in figure 4.10. The buffer handling controls the circular buffer.

Only the DATA TAKING mode will be considered in the paper. The processing in the CALIBRATION mode consists of getting the event and putting it out on the HSL without any processing. The DIAGNOSTIC DUMP mode is completely similar to the DATA TAKING mode in the processing except when the event fails one of the tests. In DATA TAKING the event is then rejected whereas in DIAGNOSTIC DUMP it is written to the HSL with a code describing which test it failed.

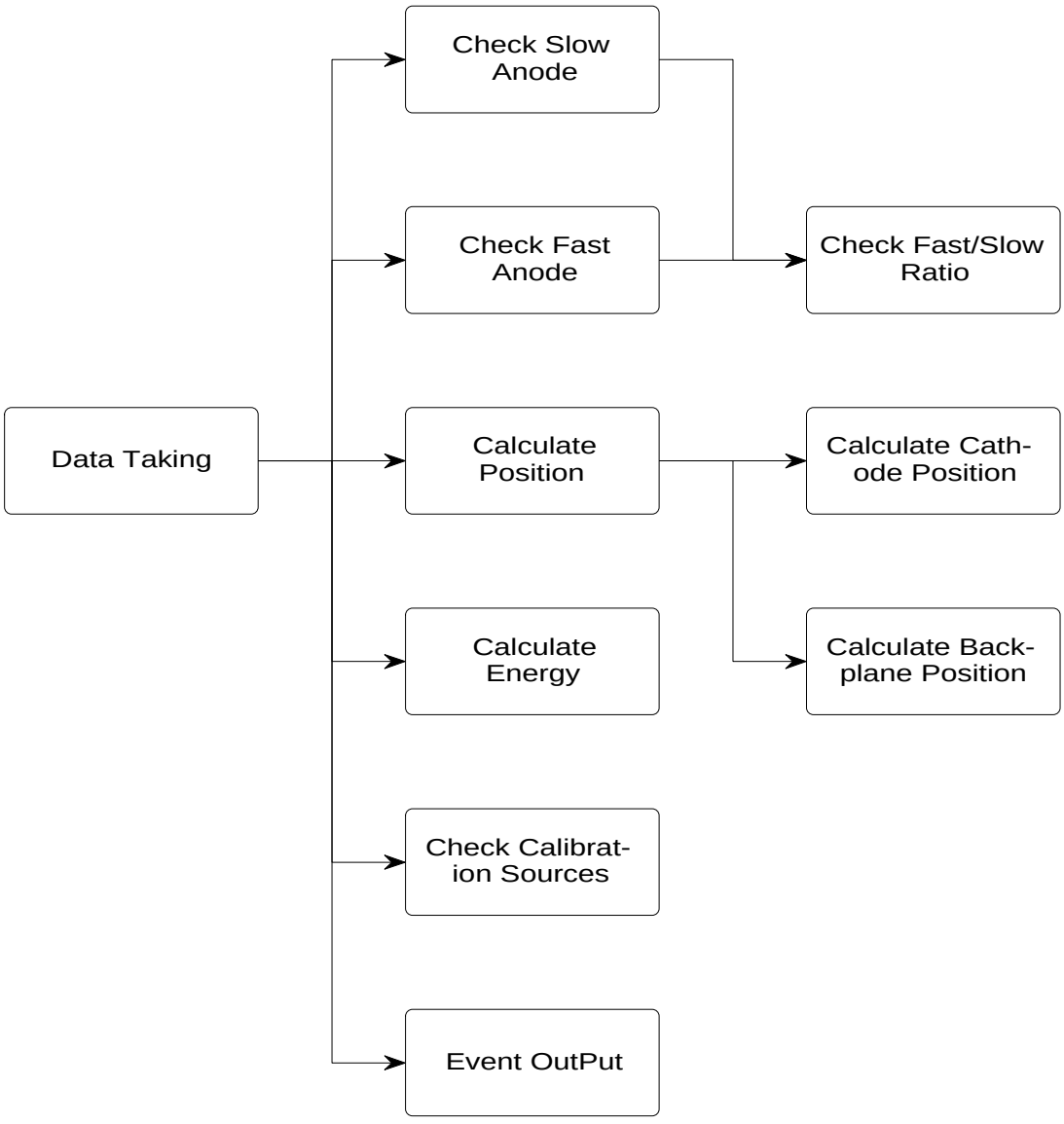

Fig. 4.11 Data Taking Event Analysis

The data taking event analysis is shown in figure 4.11. First the anode data is corrected for

offset and gain and then checked. Events with too high values are rejected as well as events with the fast/slow ratio outside the rejection limits. Then the position is calculated as shown in figure 4.12.

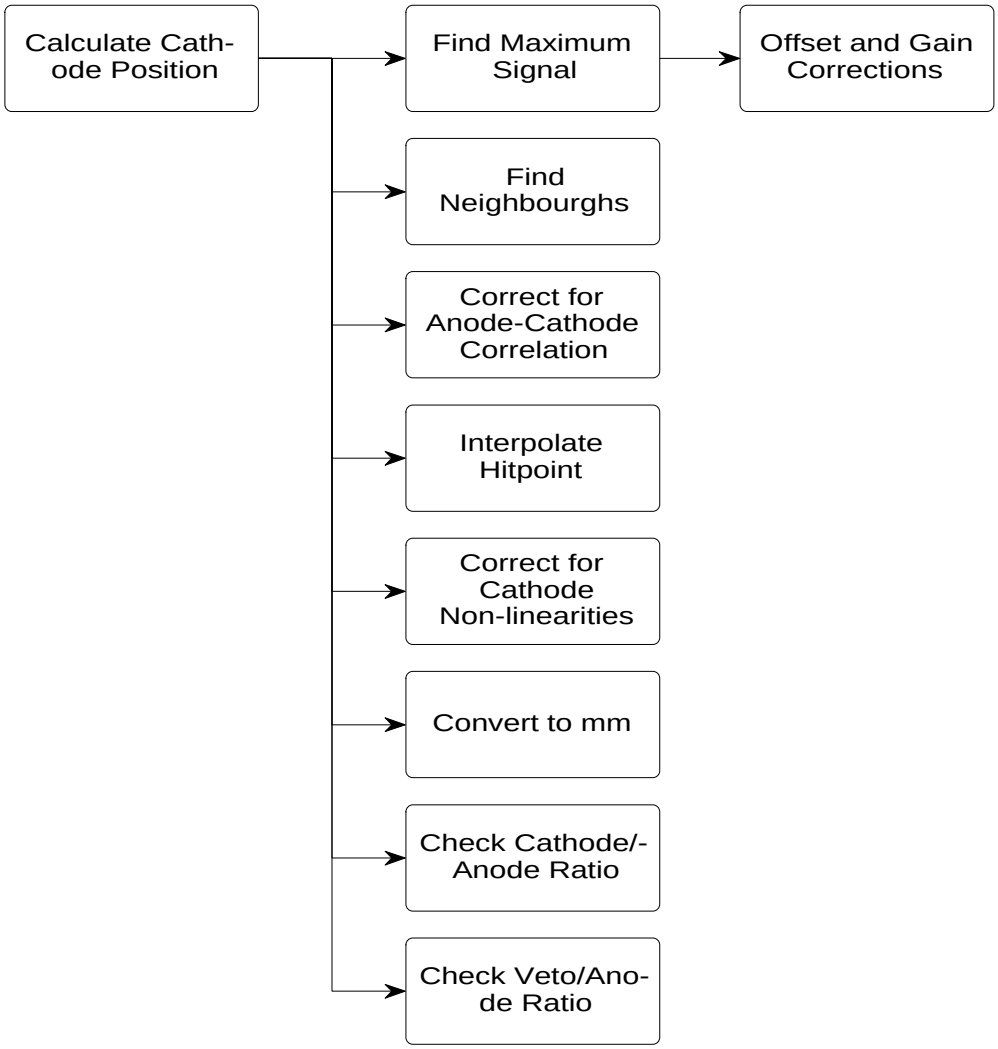

Fig. 4.12 Position calculation

The processing necessary to determine the Cathode position is shown in figure 4.12. The processing of the Backplane position determination is very similar and is therefore not shown here.

To find the maximum pulseheight it is necessary to look at all the readouts from the Cathode, itis therefore a useful place to include the corrections for Offsets and Gains.

It is obvious from figure 4.12 that the position determination involves a lot of corrections for non-linear effects. This is one area where the backplane calculation differs from the Cathode calculation. The correlations are more complicated in the Cathode. The anode-Cathode

correlations not only affect the cathode signals but also the anode signal. Thus the energy signal also has to be corrected both for correlations and for area variations. The energy calculation is shown in figure 4.13.

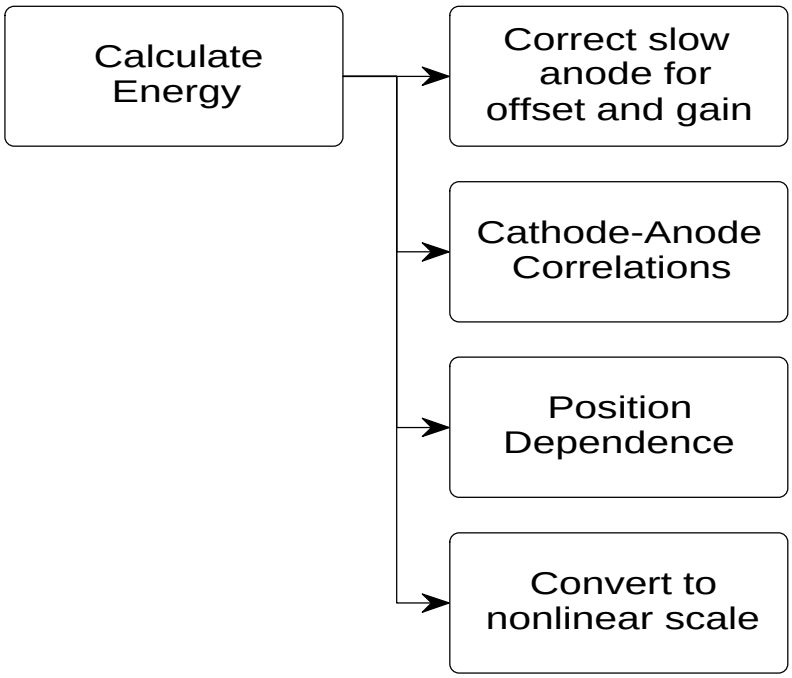

Fig. 4.13 Energy Calculation

The last part of the energy calculation is a conversion from a 12bit scale to an 8bit scale with greather resoliution at low energies than at high.

After the position has been determined both in the Cathode and in the Backplane the result is used to test whether the event is in one of the 4 calibration areas. ‘These areas can only be reached by x-rays coming from 4 radioactive sources placed in the detector collimator.

If an event is identified as a calibration source event it is not transmitted via the HSL but instead collcted in a calibration spectrum (one of four), and transmitted via the HK format.

5. Component Description

The following decomposition follows the outline in section 4. except that the interrupt handlers are included in the beginning.

The description is slightly different compared with the Template in [1]. The DFEE Software is written in assembler and fits within one memory block of 32K words. This means that any component of the software has access to all varibles in the memory. The Interface described here is therefore the FPGA I/O. This is the only place where the components import and export data and control information.

The Data section gives a list of the data used by the component whereas details about the data has been incorporated in the processing description where needed.

Other component calling or called by the component under consideration is listed under dependencies and ressources respectively.

5.1	Component Identifier:	USART Interrupt Handler
5.1.1	Type:	Interrupt Handler The Interrupt is always active.
5.1.2	Function:	Controls the Transmission and reception of Characters on the Low Speed Line
5.1.3	Interfaces:	FPGA USART Status Register FPGA USART Data Register Interrupt Mask
5.1.4	Dependencies:	Activated by the USART Interrupt
5.1.5	Processing:	Get USART Status Register Test for Reception If true Get character Store in Circular Temporary Buffer Set Input Flag Test for Transmission If true Set Output Flag Reset Interrupt
5.1.6	Data:	Input Flag Output Flag Temporary Buffer
5.1.7	Resources	
5.1.8	Memory location:	9115 - 9153

5.2	Component Identifier:	Timer A Interrupt Handler
5.2.1	Type:	Interrupt Handler. The Interrupt is only active in Calibration State.
5.2.2	Function:	Signals when a Timer A Interrupt has occurred
5.2.3	Interfaces:	Interrupt Mask
5.2.4	Dependencies:	Activated by the Timer A Interrupt.
5.2.5	Processing:	Halt the Timer Set the Timer A Flag
5.2.6	Data:	Timer A Flag
5.2.7	Resources	
5.2.8	Memory Location:	9154 - 9161

5.3	Component Identifier:	Timer B Interrupt Handler
5.3.1	Type:	Interrupt Handler The Interrupt is only active when adjusting the HV settings
5.3.2	Function:	Signals when a Timer B Interrupt has occurred
5.3.3	Interfaces:	Interrupt Mask
5.3.4	Dependencies:	Activated by the Timer B Interrupt
5.3.5	Processing:	Halt the Timer Set the Timer B Flag
5.3.6	Data:	Timer B Flag
5.3.7	Ressources	
5.3.8	Memory Location:	9162 - 916F

5.4	Component Identifier:	OBT Interrupt Handler
5.4.1	Type:	Interrupt Handler. The Interrupt is always active.
5.4.2	Function:	Counts the number of 8 second pulses.
5.4.3	Interfaces:	Interrupt Mask.
5.4.4	Dependencies:	Activated by the OBT Interrupt.
5.4.5	Processing:	Increment the HK Cycle Counter. Reset the Interrupt.
5.4.6	Data:	The HK Cycle Counter
5.4.7	Resources	
5.4.8	Memory Location:	9170 - 9185

5.5	Component Identifier:	HV Interrupt Handler
5.5.1	Type:	Interrupt Handler
5.5.2	Function:	Activated by external switch off of the HV Active when HV is ON
5.5.3	Interfaces:	Interrupt Mask
5.5.4	Dependancies:	Activated by Ext. HV Interrupt
5.5.5	Processing:	Set Ext. HV Flag Reset Interrupt
5.5.6	Data:	Ext. HV Flag
5.5.7	Resources	
5.5.8	Memory Location:	9186 - 919D

5.6	Component Identifier:	Event Interrupt Handler
5.6.1	Type:	Interrupt Handler Interrupt is only active in the Data Taking State, Calibration State and Diagnostic Dump State.
5.6.2	Function:	Determine if the event should be read. If so then read it in.
5.6.3	Interfaces:	FPGA ADC Registers FPGA Status Register FPGA Clear Register Interrupt Mask
5.6.4	Dependencies:	Activated by the Event Interrupt.
5.6.5	Processing:	Time Stamp the Event Check the Data Mode If Data Taking Check The Grey Filter Check the Circular Buffer Check the FIFO If Check Passed Read Ana Board 1 Read Anode Board If Data Taking Check Anode signals If Check Passed Read Ana Board 2 Read Ana Board 3 Read Ana Board 4 Move Data into Circular Buffer Update Buffer Pointers Set Event Flag Reset Interrupt
5.6.6	Data:	Event Flag Event Buffer Event Counters
5.6.7	Resources:	brd_read Routine
5.6.8	Memory Location:	919E - 9286

5.7	Component Identifier:	brd_read
5.7.1	Type:	ADC read-in routine
5.7.2	Function:	Reading of 8 ADC Channels
5.7.3	Interfaces:	FPGA ADC Registers
5.7.4	Dependencies:	Called by Event Interrupt Handler
5.7.5	Processing:	Wait for ADCs Ready Read the 2 ADCs Wait for ADCs Ready Read the 2 ADCs Wait for ADCs Ready Read the 2 ADCs Wait for ADCs Ready Read the 2 ADCs
5.7.6	Data:	8 ADC values
5.7.7	Resources:	io_stat Routine
5.7.8	Memory Location:	97F6 - 980E

5.8	Component Identifier:	io_stat
5.8.1	Type:	ADC control Routine
5.8.2	Function:	Test ADCs Ready
5.8.3	Interfaces:	FPGA Status Register
5.8.4	Dependencies:	Called by brd_read
5.8.5	Processing:	Get ADC Status Repeat Until Ready More than 10 tries Signal Error
5.8.6	Data:	Error code
5.8.7	Resources	
5.8.8	Memory Location:	980F - 981A

5.9	Component Identifier:	InitDFEE
5.9.1	Type:	Initialization program
5.9.2	Function:	Prepare Hardware and Software
5.9.3	Interfaces:	INT_MASK MULTIPLEX HOLD_CON_0 HOLD_CON_! HOLD_CON_2 USART_CONTROL ANODE_CON LOW_LEVEL
5.9.4	Dependencies:	
5.9.5	Processing:	Set up the Interrupt Addresses Activate USART and OBT Interrupts Set up ADC Multiplexer Set Event Hold Times Set up Low Speed Line Set Hardware Configuration Expand Event Rejection parameters
5.9.6	Data:	
5.9.7	Resources:	parclear par_unfold
5.9.8	Memory Location:	9000 - 9097

5.10	Component Identifier:	parclear
5.10.1	Type:	Service Routine
5.10.2	Function:	Clear Memory
5.10.3	Interfaces:	
5.10.4	Dependencies:	
5.10.5	Processing:	Clear specified area
5.10.6	Data:	
5.10.7	Resources	
5.10.8	Memory Location:	9098 - 909E

5.11	Component Identifier:	par_unfold
5.11.1	Type:	parameter preparation routine
5.11.2	Function:	Calculates the parameters needed to control the Validity of the x-ray events
5.11.3	Interfaces:	
5.11.4	Dependencies:	initialization
5.11.5	Processing:	Controls the calculation of the rejection parameters Each set of parameters are changed from fixed to Floating point and their differences are calculated and Stored. Set up source and target pointers for one type of Parameters Do the calculation Repeat until all parameters are calculated
5.11.6	Data:	
5.11.7	Resources:	expand
5.11.8	Memory Location:	909F - 90E6

5.12	Component Identifier:	expand
5.12.1	Type:	parameter calculation routine
5.12.2	Function:	Performs the calculations of one set of rejection Parameters
5.12.3	Interfaces:	
5.12.4	Dependencies:	par_unfold
5.12.5	Processing:	set up target and source pointers Convert one number from integer to Floating point Repeat until done Calculate and store difference between two consecutive numbers Repeat until done
5.12.6	Data:	
5.12.7	Resources:	par_unfold
5.12.8	Memory Location:	90E7 - 9114

5.13	Component Identifier:	Idle Loop
5.13.1	Type:	Main routine
5.13.2	Function:	Check the Event Flags and reset Watchdog
5.13.3	Interfaces:	FPGA Reset Register
5.13.4	Dependencies:	
5.13.5	Processing:	Test for Input Character If Flag call Handler Test for Output Character If Flag call Handler Test for Input Message If Flag call Handler Test for Output Message If Flag call Handler Test for X-ray Events If Flag call Handler Test for FIFO Flushing If Flag call Handler Test for Timer A If Flag call Handler Test for Timer B If Flag call Handler Test for HV off If Flag call Handler Reset Watchdog Repeat from Start
5.13.6	Data:	Event Flags
5.13.7	Resources:	indata Routine outdata Routine crcck Routine ls_in Routine event_input Routine FIFOflu Routine timhand Routine Hvhandle Routine
5.13.8.	Memory Location:	9287 - 9315

5.14	Component Identifier:	indata
5.14.1	Type:	USART Interrupt Service Routine
5.14.2	Function:	Analyse the input Characters and build a Input Message
5.14.3	Interfaces:	
5.14.4	Dependencies:	Called from Idle Loop
5.14.5	Processing:	Get Character from Circular Buffer Check Input Level If Outside Message Check for Opening Character No then Exit Yes Level+1 then Exit If Inside Message Add Character to Message Check For Closing Character No then Exit Yes Level+1 If Message End Adjust Length Reset Level Set Input Ready Flag
5.14.6	Data:	Input Ready Flag Character Buffer Input Buffer
5.14.7	Resources	
5.14.8	Memory Location:	9667 - 96E6

5.15	Component Identifier:	outdata
5.15.1	Type:	USART Interrupt Service Routine
5.15.2	Function:	Output single Characters from a Buffer
5.15.3	Interfaces:	FPGA USART Data Register
5.15.4	Dependencies:	Called from Idle Loop
5.15.5	Processing:	Check if more Characters If Yes Write to USART Exit
5.15.6	Data:	Output Buffer
5.15.7	Resources	
5.15.8	Memory Location:	96E7 - 9707

5.16	Component Identifier:	crcck
5.16.1	Type:	CRC calculation
5.16.2	Function:	Verify the CRC of the Input Messages
5.16.3	Interfaces	
5.16.4	Dependencies:	Called when a Input Message is ready
5.16.5	Processing:	Calculate CRC of Message Compare with Transmitted CRC If same set Message Valid Flag If different set Error Code = 1 Call answer_back
5.16.6	Data:	Input Buffer Message Valid Flag Error Code
5.16.7	Resources:	answer_back Routine crc_calc
5.16.8	Memory Location:	9708 - 971E

5.17	Component Identifier:	crc_calc
5.17.1	Type:	Service Routine
5.17.2	Function:	Calculate the CRC value for the input string
5.17.3	Interfaces:	
5.17.4	Dependencies:	crcck
5.17.5	Processing:	get first word Get first bit Check bit If more bits get next bit If more words get next word Return crc value
5.17.6	Data:	
5.17.7	Resources	
5.17.8	Memory Location:	9756 - 9772

5.18	Component Identifier:	answer_back
5.18.1	Type:	Request Acknowlegde Routine
5.18.2	Function:	Output Request Acknowledge
5.18.3	Interfaces:	FPGA USART Data Register
5.18.4	Dependencies:	Called from different routines when an Input Message do not generate other responses.
5.18.5	Processing:	Get Answer Code Setup Output Message Send first Character to USART
5.18.6	Data:	Answer Code Output Buffer
5.18.7	Resources	
5.18.8	Memory Location:	9BB5 - 9BDF

5.19 Component Identifier: ls_in

5.19.1 Type: Messages Interpreter

5.19.2 Function: Determines the Actions Requested by the LSL Message

5.19.3 Interfaces:

5.19.4 Dependencies: Activated by the Message Valid Flag

5.19.5 Processing:

Get Message Type Word
Check if HK Request
    Call HK
Get Current State
Call Appropriate Service Routine

State	Routine
1	safe
2	memory
5	setup
6	Hvon
10 (A Hex)	datain
20 (14 Hex)	calib
40 (28 Hex)	ddump

Exit

5.19.6 Data: Message Type Word from Input Buffer
Current State

5.19.7 Resources: safe Routine
memory Routine
setup Routine
Hvon Routine
datain Routine
calib Routine
ddump Routine
answer_back Routine

5.19.8 Memory Location: 9316 - 9343

5.20	Component Identifier:	safe
5.20.1	Type:	Message Control
5.20.2	Function:	Test the message to be allowed in State Safe
5.20.3	Interfaces:	
5.20.4	Dependencies:	Called from ls_in
5.20.5	Processing:	Check Type of Message If State Change (1234 Hex) Call fromsafe If not set Error Code = 3 call answer_back
5.20.6	Data:	Input buffer Error Code
5.20.7	Resources:	answer_back Routine fromsafe Routine
5.20.8	Memory Location:	93CB - 93D5

5.21	Component Identifier:	fromsafe
5.21.1	Type:	State Change Control Routine
5.21.2	Function:	Check and perform the State Change from Safe State. If the Change is Safe to Safe it is possible to execute a FPGA patch area
5.21.3	Interfaces	
5.21.4	Dependencies:	called from safe
5.21.5	Processing:	Get Target State Is it Setup, Memory or Safe = 1,2 or 5 Then Change State Else set Error Code = 3 and call answer_back Set Error Code = 0 call answer_back
5.21.6	Data:	Input Buffer Error Code
5.21.7	Resources:	answer_back Routine
5.21.8	Memory Location:	9441 - 9462

5.22	Component Identifier:	memory
5.22.1	Type:	Message Validity Check
5.22.2	Function:	Verify the validity of the LS Message
5.22.3	Interfaces:	
5.22.4	Dependencies:	Called from ls_in
5.22.5	Processing:	Get Message Type If State Change (1234 Hex) Get Target State If Safe (= 1) Call memtosafe Else set Error Flag = 4 Call answer_back If Memory load (ABCD Hex) Call memup If Memory Dump (CADB Hex) Call memdump If Memory CRC (7F7F Hex) Call memcrc Else set Error Code = 3 Call answer_back
5.22.6	Data:	Input Buffer Error Flag
5.22.7	Resources:	memtosafe Routine memup Routine memdump Routine memcrc Routine answer_back Routine
5.22.8	Memory Location:	93D6 - 93F0

5.23	Component Identifier:	memtosafe
5.23.1	Type:	State changing routine
5.23.2	Function:	Changes the State from Memory to Safe
5.23.3	Interfaces:	
5.23.4	Dependencies:	Called from memory
5.23.5	Processing:	Change Current State Set Error Code = 0 call answer_back
5.23.6	Data:	currentstate
5.23.7	Resources:	answer_back Routine
5.23.8	Memory Location:	9463 - 9470

5.24	Component Identifier:	memup
5.24.1	Type:	Memory Patch Routine
5.24.2	Function:	Patches indicated Memory area
5.24.3	Interfaces	
5.24.4	Dependencies:	Called from memory
5.24.5	Processing:	Get start addres of Patch Get length of Patch Check for Wraparound If wrap Set Error Flag = 5 call answer_back Check for conflict with primary version of memup If conflict use secondary version Load memory Set Error Code = 0 call answer_back
5.24.6	Data:	Input Buffer
5.24.7	Resources:	answer_back Routine
5.24.8	Memory Location:	95FC - 9625

5.25	Component Identifier:	memup2
5.25.1	Type:	Memory Patch Routine Alternate
5.25.2	Function:	Patches indicated Memory Area
5.25.3	Interfaces:	
5.25.4	Dependencies:	memup
5.25.5	Processing:	Load memory
5.25.6	Data:	
5.25.7	Resources	
5.25.8	Memory Location:	9943 - 9950

5.26	Component Identifier:	memdump
5.26.1	Type:	Memory Dump Routine
5.26.2	Function:	Dumps selected parts of memory
5.26.3	Interfaces	
5.26.4	Dependencies:	Called from memory
5.26.5	Processing:	Get starting address of dump Get length of dump Check validity If error set Error Code = 5 and call answer_back Transfer memory contents
5.26.6	Data:	Input Buffer Output Buffer Error Code
5.26.7	Resources:	answer_back Routine
5.26.8	Memory Location:	9626 - 9666

5.27	Component Identifier:	memcrc
5.27.1	Type:	Memory Routine
5.27.2	Function:	Calculate CRC for selected part of memory
5.27.3	Interfaces	
5.27.4	Dependencies:	Called from memory
5.27.5	Processing:	Get starting address of area Get length of area Check for wrap around If error set Error Code = 5 and call answer_back Calculate CRC Transmit Result
5.27.6	Data:	Input Buffer Output Buffer Error Code
5.27.7	Resources:	answer_back Routine
5.27.8	Memory Location:	991B - 9942

5.28	Component Identifier:	hvon
5.28.1	Type:	Hardware Control Routine
5.28.2	Function:	Turns on the High Voltage
5.28.3	Interfaces	
5.28.4	Dependencies:	Called from ls_in
5.28.5	Processing:	Get Message Type (AAAA Hex) Check it is HV On (Param1 = 0 Param2 = 5) If not Set Error Code = 3 call answer_back Check currentstate is 6 If not set Error Code = 3 call answer_back Call hwset1 Set Error Code = 0 call answer_back
5.28.6	Data:	Input Buffer Error Code
5.28.7	Resources:	answer_back Routine Hwset1
5.28.8	Memory Location:	93B4 - 93CA

5.29	Component Identifier:	hwset1
5.29.1	Type:	Hardware Control Routine
5.29.2	Function:	Performs the HV turn On
5.29.3	Interfaces:	
5.29.4	Dependencies:	Called from hvon
5.29.5	Processing:	Turn On HV Set currentstate to 5
5.29.6	Data:	
5.29.7	Resources:	answer_back
5.29.8	Memory Location:	95CC - 95FB

5.30 Component Identifier: setup
5.30.1 Type: Input service Routine
5.30.2 Function:: Check type of message
5.30.3 Interfaces
5.30.4 Dependencies: Called from ls_in
5.30.5 Processing: Get message type
Check validity
If not valid set Error code = 3 and call answer_back
Call message handler

Message Type	Routine
AAAA	hwset
8642	swup
8765	swup
FDB9	swdump
FEDC	swdump
1234	chstat
4644	enerlo
4645	enerhi
4646	enercon
1608	cpucon

5.30.6 Data: Input Buffer
Error Code
5.30.7 Resources: hwset Routine
swup Routine
swdump Routine
chstat Routine
answer_back Routine
enerlo Routine
enerhi Routine
enercon Routine
cpucon Routine
5.30.8 Memory Location: 9367 - 93B3

5.31 Component Identifier: hwset

5.31.1 Type: Message handler

5.31.2 Function: Check type of HW message

5.31.3 Interfaces

5.31.4 Dependencies: Called from setup

5.31.5 Processing:
Get message type
Check validity
    If not valid set Error Code = 4
    call answer_back
Call message execution Routine

Type	Routine
0	hvstat
1	anode
2	discri
3	moddv
4	modvc

5.31.6 Data: Input buffer
Error Code

5.31.7 Resources: hvstat Routine
modMS Routine
modGEM Routine
anode Routine
discri Routine
answer_back Routine

5.31.8 Memory Location: 9961 - 997C

5.32	Component Identifier:	anode
5.32.1	Type:	Hardware Control Routine
5.32.2	Function:	Controls the Anode setting
5.32.3	Interfaces	
5.32.4	Dependencies:	Called from hwset
5.32.5	Processing:	Get setting Mask setting Upload setting Save in Hw monitor word Set Error Code = 0 call answer_back
5.32.6	Data:	Input Buffer HW monitor word
5.32.7	Resources:	answer_back Routine
5.32.8	Memory Location:	9AB8 - 9AC1

5.33	Component Identifier:	discri
5.33.1	Type:	Hardware Control Routine
5.33.2	Function:	Controls the Low Level Discriminator Setting
5.33.3	Interfaces	
5.33.4	Dependencies:	Called from hwset
5.33.5	Processing:	Get setting Mask setting Upload setting Save in HW monitor word Set Error Code = 0 call answer_back
5.33.6	Data:	Input Buffer HW monitor word
5.33.7	Resources:	answer_back Routine
5.33.8	Memory Location:	9AC2 - 9ACC

5.34	Component Identifier:	hvstat
5.34.1	Type:	HV status Control Routine
5.34.2	Function:	Controls the HV operations
5.34.3	Interfaces	
5.34.4	Dependencies:	Called from hwset
5.34.5	Processing:	Get hv state request Check validity If not valid set Error Code = 3 and call answer_back Change hv status If status = Hvready set currentstate to 6 Save status Set Error Code = 0 call answer_back
5.34.6	Data:	Hvstatus Input Buffer Error Code
5.34.7	Resources:	answer_back Routine
5.34.8	Memory Location:	997D - 99BC

5.35	Component Identifier:	modMS
5.35.1	Type:	Hv setting Control Routine
5.35.2	Function:	Controls the setting of the MS High voltage
5.35.3	Interfaces	
5.35.4	Dependencies:	Called from hwset
5.35.5	Processing:	Get new setting Check validity If not valid set Error Code = 4 call answer_back Calculate interpolated settings Upload next setting Verify setting reached Repeat until final setting done Save setting Set Error Code = 0 call answer_back
5.35.6	Data:	HV setting Input Buffer Error Code
5.35.7	Resources:	answer_back Routine comfill comcheck HV_SET Comstep
5.35.8	Memory Location:	99BD - 9A12

5.36	Component Identifier:	modGEM
5.36.1	Type:	HV setting Control Routine
5.36.2	Function:	Controls the setting of the GEM high voltage
5.36.3	Interfaces	
5.36.4	Dependencies:	Called from hwset
5.36.5	Processing:	Get new setting Check validity If not valid set Error Code = 4 call answer_back Calculate interpolated settings Upload next setting Verify setting reached Repeat until last setting done Save setting Set Error Code = 0 call answer_back
5.36.6	Data:	HV setting Input Buffer Error Code
5.36.7	Resources:	answer_back Routine comfill comcheck HV_SET Comstep
5.36.8	Memory Location:	9A13 - 9A6C

5.37	Component Identifier:	comfill
5.37.1	Type:	HV service routine
5.37.2	Function:	Calculate interpolated settings
5.37.3	Interfaces:	
5.37.4	Dependencies:	Called from modMS and modGEM
5.37.5	Processing:	Calculate up to 10 steps to new value
5.37.6	Data:	
5.37.7	Resources	
5.37.8	Memory Location:	9A6D - 9A7D

5.38	Component Identifier:	comcheck
5.38.1	Type:	HV service routine
5.38.2	Function:	Select first higher interpolated settings
5.38.3	Interfaces:	
5.38.4	Dependencies:	Called from modMS and modGEM
5.38.5	Processing:	Loop through setting table until found
5.38.6	Data:	
5.38.7	Resources	
5.38.8	Memory Location:	9A7E - 9A8A

5.39	Component Identifier:	comstep
5.39.1	Type:	HV service routine
5.39.2	Function:	Go to next interpolated setting
5.39.3	Interfaces:	
5.39.4	Dependencies:	Called from modMS and modGEM
5.39.5	Processing:	Check if more settings If not call HVclean Get next setting
5.39.6	Data:	
5.39.7	Resources	Timer B HVclean
5.39.8	Memory Location:	9A9C - 9AB7

5.40	Component Identifier:	HVclean
5.40.1	Type:	HV service routine
5.40.2	Function:	Terminate HV command processing
5.40.3	Interfaces:	
5.40.4	Dependencies:	Called from comstep
5.40.5	Processing:	Resets Timer B
5.40.6	Data:	
5.40.7	Resources	Timer B
5.40.8	Memory Location:	9A8B - 9A9B

5.41	Component Identifier:	swint
5.41.1	Type:	Software Parameter Update Routine
5.41.2	Function:	Change the Value of one of the Integer Software Parameters
5.41.3	Interfaces	
5.41.4	Dependencies:	Called from swup
5.41.5	Processing:	Get Parameter Number Check Validity If not valid Set Error Code = 5 and call answer_back Update Parameter Set Error Code = 0 call answer_back
5.41.6	Data:	Input Buffer Integer Parameter Table Error Code
5.41.7	Resources:	answer_back Routine
5.41.8	Memory Location:	0942 - 9855

5.42	Component Identifier:	swftp
5.42.1	Type:	Software Parameter Update Routine
5.42.2	Function:	Change the Value of one of the Software Floating Point Parameters
5.42.3	Interfaces	
5.42.4	Dependencies:	Called from swup
5.42.5	Processing:	Get Parameter Number Check Validity If not valid Set Error Code = 5 and call answer_back Update Parameter Set Error Code = 0 call answer_back
5.42.6	Data:	Input Buffer Floating Point Parameter Table Error Code
5.42.7	Resources:	answer_back Routine
5.42.8	Memory Location:	9856 - 986D

5.43	Component Identifier:	intdump
5.43.1	Type:	Software Control Routine
5.43.2	Function:	Obtains the value of a selected Software Integer Parameter
5.43.3	Interfaces	
5.43.4	Dependencies:	Called from swdump
5.43.5	Processing:	Get Parameter Number Check Validity If not Valid Set Error Code = 5 call answer_back Get Parameter Value Build message in Output Buffer Start Transmission
5.43.6	Data:	Input Buffer Error Code Output buffer
5.43.7	Resources:	answer_back Routine
5.43.8	Memory Location:	987B - 98AD

5.44	Component Identifier:	ftpdump
5.44.1	Type:	Software Control Routine
5.44.2	Function:	Obtains the value of a selected Software Floating Point Parameter
5.44.3	Interfaces	
5.44.4	Dependencies:	Called from swdump
5.44.5	Processing:	Get Parameter Number Check Validity If not Valid Set Error Code = 5 call answer_back Get Parameter Value Build message in Output Buffer Start Transmission
5.44.6	Data:	Input Buffer Error Code Output buffer
5.44.7	Resources:	answer_back Routine
5.44.8	Memory Location:	98AE - 98D5

5.45	Component Identifier:	chchk1
5.45.1	Type:	State Control Routine
5.45.2	Function:	Identifies the Target state and call the appropriate Routine
5.45.3	Interfaces	
5.45.4	Dependencies:	Called from setup
5.45.5	Processing:	Get Target State If State = Setup Call setsetup If State = Data Taking Call setodat If State = Calibration Call setocal If State = Diagnostic Dump Call setodum If State = Safe Call setosafe Else Set Error Code = 3 call answer_back
5.45.6	Data:	Input Buffer Error Code
5.45.7	Resources:	setsetup Routine setodat Routine setocal Routine setodum Routine setosafe Routine answer_back Routine
5.45.8	Memory Location:	95AE - 95CB

5.46	Component Identifier:	setsetup
5.46.1	Type:	State Changing Routine
5.46.2	Function:	Changes currentstate tosetup
5.46.3	Interfaces	
5.46.4	Dependencies:	Called from chstat
5.46.5	Processing:	Set Error Code = 0 call answer_back
5.46.6	Data	
5.46.7	Resources:	answer_back Routine
5.46.8	Memory Location:	93F1 - 93F4

5.47	Component Identifier:	setodat
5.47.1	Type:	State Changing Routine
5.47.2	Function:	Changes currentstate to data_taking and start the Data Taking
5.47.3	Interfaces:	Interrupt Mask
5.47.4	Dependencies:	Called from chstat
5.47.5	Processing:	Set up the 100% Grey Filter Reset Event Buffer Set Data Mode = 0 Change currentstate to A (Hex) Activate Event Interrupt Set Error Code = 0 call answer_back
5.47.6	Data:	Input Buffer Grey Filters Event Buffer currentstate
5.47.7	Resources:	answer_back Routine
5.47.8	Memory Location:	9490 - 94CA

5.48	Component Identifier:	setocal
5.48.1	Type:	State Changing Routine
5.48.2	Function:	Change currentstate to Calibration and start the Calibration
5.48.3	Interfaces:	Interrupt Mask FPGA Reset Register FPGA Control Register
5.48.4	Dependencies	Called from chstat
5.48.5	Processing:	Reset Event Buffer Get Number of Events/Level Verify range If not valid Set Error Code = 5 call answer_back Save Change currentstate to 14 (Hex) Load first Calibration Level Transmit to FPGA Set Data Mode = 1 Activate Event Interrupt Fire Calibration Pulser Set Error Code = 0 call answer_back
5.48.6	Data:	Input Buffer Error Code
5.48.7	Resources:	answer_back Routine
5.48.8	Memory Location:	94CB - 9510

5.49	Component Identifier:	setodum
5.49.1	Type:	State Changing Routine
5.49.2	Function:	Change currentstate to dump
5.49.3	Interfaces:	Interrupt Mask
5.49.4	Dependencies:	Called from chstat
5.49.5	Processing:	Reset Event Buffer Get Number of Dump Events Verify range If not valid Set Error Code = 5 call answer_back Save Change currentstate to 28 (Hex) Set Data Mode = 2 Activate Event Interrupt Set Error Code = 0 call answer_back
5.49.6	Data:	Input Buffer Error Code
5.49.7	Resources:	answer_back Routine
5.49.8	Memory Location:	9511 - 9552

5.50	Component Identifier:	setosafe
5.50.1	Type:	State Changing Routine
5.50.2	Function:	Change currentstate to Safe
5.50.3	Interfaces	
5.50.4	Dependencies:	Called from chstat
5.50.5	Processing:	Get HV Status If HV ON Set HV Off Set currentstate to safe Set Error Code = 0 call answer_back
5.50.6	Data:	HV Status
5.50.7	Resources:	answer_back Routine
5.50.8	Memory Location:	9471 - 948F

5.51	Component Identifier:	hkcoll
5.51.1	Type:	House Keeping Data Collection Routine
5.51.2	Function:	Gets the House Keeping data and transmit it to the DPE
5.51.3	Interfaces:	FPGA ADC Registers USART Data Register
5.51.4	Dependencies:	Called from ls_in
5.51.5	Processing:	Set up connection to Output Buffer Get and Store HK Cycle Number Check HV Status If HV Ready Increment counter Check ready Duration If Time ran out currentstate = 5 Set HV Off Store currentstate Get and Store Low Level and Anode Check State If active get HV values from Event If not Read from ADC Store HV values Get, Store and Reset Counters Call speccon Start HK output
5.51.6	Data:	Output Buffer Currentstate HW status HV monitor values Event Counters
5.51.7	Resources:	speccon Routine
5.51.8	Memory Location:	9ACD - 9B46

5.52	Component Identifier:	speccon
5.52.1	Type:	Calibration Spectra Collection Routine
5.52.2	Function:	Controls the Calibration Spectra Duble Buffer
5.52.3	Interfaces	
5.52.4	Dependencies:	Called from HK
5.52.5	Processing:	Get Number of Segment to be Transferred If Segment Number = 32 Switch pointer to other Buffer Clean old Buffer Set Collection Pointer to Old Buffer Set Number = 0 Get and Store a Segment of 8 channels from Spectrum 1 Get and Store a Segment of 8 channels from Spectrum 2 Get and Store a Segment of 8 channels from Spectrum 3 Get and Store a Segment of 8 channels from Spectrum 4
5.52.6	Data:	Output Buffer Calibration Spectra Area Segment Pointer
5.52.7	Resources	
5.52.8	Memory Location:	9B47 - 9BB4

5.53	Component Identifier:	datain
5.53.1	Type:	State Control Routine
5.53.2	Function:	Checks messages when in state Data Taking
5.53.3	Interfaces	
5.53.4	Dependencies:	Called from ls_in
5.53.5	Processing:	Get Message Type If State Change Check target = setup If not Set Error Code = 3 call answer_back If yes Change State If Grey Filter Get Filter Number Check Range of Number If not Valid Set Error Code = 5 Call answer_back Call greyflt
5.53.6	Data:	Input Buffer Grey Filters Filter Control
5.53.7	Resources:	greyflt Routine dtosetup Routine answer_back Routine
5.53.8	Memory Location:	9344 - 9352

5.54	Component Identifier:	gfilt
5.54.1	Type:	Grey Filter Control Routine
5.54.2	Function:	Get the Bit combination of a Grey Filter and creates the Filter Parameters
5.54.3	Interfaces	
5.54.4	Dependencies:	Called from datain
5.54.5	Processing:	Get 32 bit Filter Code Check each bit If bit Set Parameter = 1 If bit clear Parameter = 0
5.54.6	Data:	Filter Codes Filter Parameters
5.54.7	Resources	
5.54.8	Memory Location:	9553 - 9595

5.55	Component Identifier:	dtosetup
5.55.1	Type:	State Change Control Routine
5.55.2	Function	Termination of one of the Active States and return to setup
5.55.3	Interfaces:	Interrupt Mask
5.55.4	Dependencies:	Called from datain Called from calib Called from ddump
5.55.5	Processing:	Reset Event Interrupt Setup FIFO Flush Change State to setup Check if State Change is Forced If Forced set Error Code = 0 call answer_back
5.55.6	Data:	currentstate Output Buffer
5.55.7	Resources:	answer_back Routine
5.55.8	Memory Location:	93F5 - 9440

5.56	Component Identifier:	calib
5.56.1	Type:	State Control Routine
5.56.2	Function:	Checks the Messages in Calibration State
5.56.3	Interfaces	
5.56.4	Dependencies:	Called from ls_in
5.56.5	Processing:	Get Message Type If State Change Get Target State Check if setup If setup call dtosetup Else Set Error Code = 3 call answer_back Else Set Error Code = 0 call answer_back
5.56.6	Data:	Input Buffer Currentstate Output Buffer
5.56.7	Resources:	dtosetup Routine answer_back Routine
5.56.8	Memory Location:	9353 - 935D

5.57	Component Identifier:	ddump
5.57.1	Type:	State Control Routine
5.57.2	Function:	Checks the Messages in Diagnostic Dump State
5.57.3	Interfaces	
5.57.4	Dependencies:	Called from ls_in
5.57.5	Processing:	Get Message Type If State Change Get Target State Check if setup If setup call dtosetup Else Set Error Code = 3 call answer_back Else Set Error Code = 0 call answer_back
5.57.6	Data:	Input Buffer Currentstate Output Buffer
5.57.7	Resources:	dtosetup Routine answer_back Routine
5.57.8	Memory Location:	935E - 9366

5.58	Component Identifier:	timhand
5.58.1	Type:	Interrupt Service Routine
5.58.2	Function:	Controls the Timer during Calibration
5.58.3	Interfaces:	FPGA Clear Register FPGA DAC Register
5.58.4	Dependencies:	Called from Idle Loop
5.58.5	Processing:	Reset Calibration Pulser Check number of Pulses If Level End thenCheck for More Levels If more Change Level If no More terminate Call dtosetup Restart Timer Fire a Pulse
5.58.6	Data:	Calcount Callevel Calmore
5.58.7	Resources:	dtosetup Routine
5.58.8	Memory Location:	9773 - 9791

5.59	Component Identifier:	FIFOflu
5.59.1	Type:	FIFO Flush Handling
5.59.2	Function:	Fills the FIFO with Dummy Events to Trigger a Half Full Signal to The DPE
5.59.3	Interfaces:	FPGA FIFO Register
5.59.4	Dependencies:	Called from Idle Loop
5.59.5	Processing:	Get Circular Event Buffer Status If not Empty Terminate Get Data Mode Build Dummy Event to fit the Data Mode Transmit the necessary number of Events to FIFO Reset FIFO Flush Flag
5.59.6	Data:	Event Buffer status Data Mode FIFO Flush Flag
5.59.7	Resources:	
5.59.8	Memory Location:	971F - 9755

5.60	Component Identifier:	calmore
5.60.1	Type:	timer support routine
5.60.2	Function:	Initializes one calibration pulse
5.60.3	Interfaces:	INT_MASK
5.60.4	Dependencies:	Called from timhand
5.60.5	Processing:	Reset Interrupt Start timer A Fire calibration pulse
5.60.6	Data:	
5.60.7	Resources	
5.60.8	Memory Location:	9792 - 97A6

5.61	Component Identifier:	calend
5.61.1	Type:	timer A support routine
5.61.2	Function:	Auto termination of calibration
5.61.3	Interfaces:	CAL_LEVEL INT_MASK
5.61.4	Dependencies:	Called from timhand
5.61.5	Processing:	Rearm trigger Call dtosetup Reset calibration voltage level
5.61.6	Data:	
5.61.7	Resources	
5.61.8	Memory Location:	97A7 - 97BF

5.62	Component Identifier:	timhand2
5.62.1	Type:	timer B handler
5.62.2	Function:	controls the actions during HV changes
5.62.3	Interfaces:	Timer B
5.62.4	Dependencies:	Called from IdleLoop
5.62.5	Processing:	Read the HV Monitoring value using ADC Check if it is still rising If yes restart Timer B If no transfer control to comstep
5.62.6	Data:	
5.62.7	Resources	comstep readADC
5.62.8	Memory Location:	97C0 - 97D9

5.63	Component Identifier:	readADC
5.63.1	Type:	Support routine
5.63.2	Function:	Direct read of HV Monitoring values
5.63.3	Interfaces:	CHANNEL_OUT USART_STATUS ADC1_PORT ADC2_PORT CLEAR_REG
5.63.4	Dependencies:	Called from tim2hand and hkcoll
5.63.5	Processing:	Set ADC multiplexer to HV monitoring Read ADCs Reset ADCs
5.63.6	Data:	
5.63.7	Resources	
5.63.8	Memory Location:	97DA - 97F5

5.64	Component Identifier:	enerlo / enerhi
5.64.1	Type:	Energy Conversion Control
5.64.2	Function:	Build Energy Conversion Table
5.64.3	Interfaces	
5.64.4	Dependencies:	Called from setup
5.64.5	Processing:	Get Message Type Load starting address Move table segment
5.64.6	Data:	Input Buffer Segment Table
5.64.7	Resources:	answer_back Routine
5.64.8	Memory Location:	9BFB - 9C07

5.65	Component Identifier:	enercon
5.65.1	Type:	Energy Conversion Table
5.65.2	Function:	Expands the Energy Conversion Table
5.65.3	Interfaces	
5.65.4	Dependencies:	Called from setup
5.65.5	Processing:	Create table of Energy values for all 4096 Possible Slow Anode ADC values Set Error Code = 0 call answer_back
5.65.6	Data:	Energy Parameter Table Energy Value Table Error Code
5.65.7	Resources:	answer_back Routine
5.65.8	Memory Location:	9C08 - 9C32

5.66	Component Identifier:	hvhandle
5.66.1	Type:	Interrupt Handler
5.66.2	Function:	Activated by external switch off of the HV Active when HV is ON
5.66.3	Interfaces:	Interrupt Mask
5.66.4	Dependancies	
5.66.5	Processing:	Set State to Safe Activate FIFOflu if active State Reset Interrupt
5.66.6	Data:	Interrupt Mask Currentstate FIFO Flag
5.66.7	Resources	
5.66.8	Memory Location:	9C33 - 9C63

5.67	Component Identifier:	ev_ana
5.49.1	Type:	Event Flow Control Routine
5.49.2	Function:	Determines the Data Mode and Calls the Correct Routine
5.49.3	Interfaces	
5.49.4	Dependencies:	Called from Idle Loop
5.49.5	Processing:	Set up Circular Event Buffer Pointer and flags Get datmode If datmode = 0 Call anadat If datmode = 1 Call caldat If datmode = 2 Call dumdat
5.49.6	Data:	datmode
5.67.7	Resources	analyse Routine datadump Routine calibrate Routine
5.67.8	Memory Location:	B000 - B01B

5.68	Component Identifier:	analyse
5.68.1	Type:	X-ray Event Analysis Routine
5.68.2	Function:	Analyses one X-ray Event in Data Taking State. Determines if the Event should be rejected, accepted or classisfied as Calibration Event.
5.68.3	Interfaces:	FPGA FIFO Register
5.68.4	Dependencies:	Called from event_input
5.68.5	Processing:	Correct and check Slow Anode If fail then Increment counter and cleanup Correct and check Fast Anode If fail then Increment counter and cleanup Correct Veto value Calculate and check fast/slow ratio If fail then Increment counter and cleanup Find cathode maximum Correct for anode/cathode coupling Calculate cathode position and convert to mm Calculate and check cathode/anode ratio If fail then Increment counter and cleanup Check Veto/anode ratio If fail then Increment counter and cleanup Find backplane maximum Correct for cathode backplane coupling Calculate backplane position and convert to mm Calculate and check backplane /anode ratio If fail then Increment counter and cleanup Check if Calibration source event If calib Add Event to Spectrum and cleanup If not call output
5.68.6	Data:	Event Buffer Event Buffer Pointer Event Data Rejection Value Event Result HK Cycle Timer Value Energy X Position Y position Calib Value

5.68.7	Resources:	slowanode Routine clean_up Routine fastanode Routine vetoval Routine startpos Routine f_max Routine cal_neigh Routine anodcoup Routine chit Routine cflexpos Routine polcorr Routine hitresultc Routine catindex Routine catcheck Routine cat_back Routine bhit Routine hitresultb Routine backindex Routine backcheck Routine xcalib Routine getout Routine
5.68.8	Memory Location:	B01C - B0E3

5.69 Component Identifier: datadump
5.69.1 Type: X-ray Event Analysis Routine
5.69.2 Function: Analyses one X-ray Event in Diagnostic Dump State.
Determines if the Event should be rejected, accepted or classisfied as Calibration Event.
All evernt will be written to the FIFO
5.69.3 Interfaces: FPGA FIFO Register
5.69.4 Dependencies: Called from event_input
5.69.5 Processing:
Correct and check Slow Anode
If fail then
    Increment counter and dump all data
Correct and check Fast Anode
If fail then
    Increment counter and dump all data
Correct Veto value
Calculate and check fast/slow ratio
If fail then
    Increment counter and dump all data
Find cathode maximum
Correct for anode/cathode coupling
Calculate cathode position and convert to mm
Calculate and check cathode/anode ratio
If fail then
    Increment counter and dump all data
Check Veto/anode ratio
If fail then
    Increment counter and dump all data
Find backplane maximum
Correct for cathode backplane coupling
Calculate backplane position and convert to mm
Calculate and check backplane /anode ratio
If fail then
    Increment counter and dump all data
Check if Calibration source event
If calib Add Event to Spectrum
Dump everything
5.69.6 Data:
Event Buffer
Event Buffer Pointer
Event Data
Rejection Value
Event Result
    HK Cycle
    Timer Value
    Energy
    X Position
    Y position

		Calib Value
5.69.7	Resources:	dtosetup Routine
		slowanode Routine
		clean_up Routine
		fastanode Routine
		vetoval Routine
		startpos Routine
		f_max Routine
		cal_neigh Routine
		anodcoup Routine
		chit Routine
		cflexpos Routine
		polcorr Routine
		hitresultc Routine
		catindex Routine
		catcheck Routine
		cat_back Routine
		bhit Routine
		hitresultb Routine
		backindex Routine
		backcheck Routine
		xcalib Routine
		full_dump Routine
5.69.8	Memory Location:	B11F - B27D

5.70	Component Identifier:	calibrate
5.70.1	Type:	Calibration Event Routine
5.70.2	Function:	Transmits the Event Data via FIFO
5.70.3	Interfaces:	FPGA FIFO Register
5.70.4	Dependencies:	Called from event_input
5.70.5	Processing:	Get Event Data
		Write Event Data to FIFO (36 words)
5.70.6	Data:	Event Data
5.70.7	Resources	full_dump Routine
5.70.8	Memory Location:	B27E - B28B

5.71	Component Identifier:	getout		
5.71.1	Type:	FIFO writing Routine		
5.71.2	Function:	Writes the results from data taking and increment the good event counter		
5.71.3	Interfaces:	FIFO_OUT		
5.71.4	Dependencies:	Called from analyse		
5.71.5	Processing:	Write	event ID	F000
			Hkcycles	
			eventtime	
			energy	
			x-position	
			y-position	
5.71.6	Data:			
5.71.7	Resources	clean_up		
5.71.8	Memory Location:	B0E4 - B103		

5.72	Component Identifier:	clean_up
5.72.1	Type:	event output clean up
5.72.2	Function:	resets the buffer and other event parameters
5.72.3	Interfaces:	
5.72.4	Dependencies:	Called from getout
5.72.5	Processing:	reset buffer flags Check if events are waiting If not Clear event flag
5.72.6	Data:	
5.72.7	Resources	
5.72.8	Memory Location:	B104 - B11E

5.73	Component Identifier:	full_dump
5.73.1	Type:	Event output routine
5.73.2	Function:	writes all raw event data to the FIFO
5.73.3	Interfaces:	FIFO_OUT
5.73.4	Dependencies:	Called from calibrate Called from datadump
5.73.5	Processing:	Move raw data to FIFO Check mode If datadump move status and source number to FIFO Call clean_up
5.73.6	Data:	
5.73.7	Resources	clean_up
5.73.8	Memory Location:	B28C - B2C0

5.74	Component Identifier:	f_max
5.74.1	Type:	Event analysis Routine
5.74.2	Function:	Find the maximum value in an integer array
5.74.3	Interfaces:	
5.74.4	Dependencies:	Called from analyse
		Called from datadump
5.74.5	Processing:	initialize maximum
		Call convert Routine
		Save in FtpTemp
		Compare with maximum
		Replace if greather
		Repeat until end
5.74.6	Data:	Start of array
		Length of array
		FtpTemp
5.74.7	Resources	convert Routine
5.74.8	Memory Location:	B2C1 - B309

5.75	Component Identifier:	convert
5.75.1	Type:	Event analysis Routine
5.75.2	Function:	Convert for offset and gain
5.75.3	Interfaces:	
5.75.4	Dependencies:	Called from f_max
5.75.5	Processing:	Correct for offset Convert the integer to floating point Correct for gain
5.75.6	Data:	
5.75.7	Resources	
5.75.8	Memory Location:	B30A - B311

5.76	Component Identifier:	cal_neigh
5.76.1	Type:	Event analysis Routine
5.76.2	Function:	Calculate neighbourgh positions
5.76.3	Interfaces:	
5.76.4	Dependencies:	Called from analyse Called from datadump
5.76.5	Processing:	Find right neighbourgh Find farright neighbourgh Find left neighbourgh Find farleft neighbourgh Correct for array wraparound Save neighbourgh addresses
5.76.6	Data:	
5.76.7	Resources	
5.76.8	Memory Location:	B323 - B33F

5.77	Component Identifier:	xcalib
5.77.1	Type:	Event analysis Routine
5.77.2	Function:	Check if an event is from a calibration Source
5.77.3	Interfaces:	
5.77.4	Dependencies:	Called from analysis Called from datadump
5.77.5	Processing:	Check if low source Check if right source Check if left source Check if high source Check if right source Check if left source If source Increment counter Add to spectrum If not source disregard
5.77.6	Data:	
5.77.7	Resources	
5.77.8	Memory Location:	B365 - B3C1

5.78	Component Identifier:	chit
5.78.1	Type:	Event analysis Routine
5.78.2	Function:	Calculate the cathode hit point
5.78.3	Interfaces:	
5.78.4	Dependencies:	Called from analyse Called from datadump
5.78.5	Processing:	Calculate right - left Divide with sum of left, center and right
5.78.6	Data:	
5.78.7	Resources	
5.78.8	Memory Location:	B3C2 - B3D2

5.79	Component Identifier:	bhit
5.79.1	Type:	Event analysis Routine
5.79.2	Function:	Calculate backplane hit point
5.79.3	Interfaces:	
5.79.4	Dependencies:	Called from analyse
		Called from datadump
5.79.5	Processing:	Calculate right - left
		Divide with sum of left, center and right
5.79.6	Data:	
5.79.7	Resources	
5.79.8	Memory Location:	B3D3 - B3E0

5.80	Component Identifier:	slowanod
5.80.1	Type:	Event analysis Routine
5.80.2	Function:	Calculates the slow anode value
5.80.3	Interfaces:	
5.80.4	Dependencies:	Called from analyse Called from datadump
5.80.5	Processing:	Set up pointers Call massage
5.80.6	Data:	
5.80.7	Resources	massage Routine
5.80.8	Memory Location:	B3E1 - B3F5

5.81	Component Identifier:	massage
5.81.1	Type:	Event analysis Routine
5.81.2	Function:	Correct for gain and offset
5.81.3	Interfaces:	
5.81.4	Dependencies:	Called from slowanod
		Called from fastanod
		Called from vetoval
5.81.5	Processing:	Mask raw data
		Subtract offset
		Convert to floating point
		Multiply with gain
5.81.6	Data:	
5.81.7	Resources	
5.81.8	Memory Location:	B3F6 - B403

5.82	Component Identifier:	fastanod
5.82.1	Type:	Event analysis Routine
5.82.2	Function:	Corrects the fast anode value
5.82.3	Interfaces:	
5.82.4	Dependencies:	Called from analyse
		Called from datadump
5.82.5	Processing:	Set up addresses
		Call massage
5.82.6	Data:	
5.82.7	Resources	massage Routine
5.82.8	Memory Location:	B404 - B415

5.83	Component Identifier:	vetoval
5.83.1	Type:	Event analysis Routine
5.83.2	Function:	Corrects the veto value
5.83.3	Interfaces:	
5.83.4	Dependencies:	Called from analyse
		Called from datadump
5.83.5	Processing:	Set up addresses
		Call massage
5.83.6	Data:	
5.83.7	Resources	massage Routine
5.83.8	Memory Location:	B416 - B427

5.84	Component Identifier:	cat_back
5.84.1	Type:	Event analysis Routine
5.84.2	Function:	Calculate the backplane correction
5.84.3	Interfaces:	
5.84.4	Dependencies:	Called from analyse Called from datadump
5.84.5	Processing:	Get correction Weight with anodevalue Correct the three positions
5.84.6	Data:	
5.84.7	Resources	
5.84.8	Memory Location:	B428 - B44F

5.85	Component Identifier:	fastslow
5.85.1	Type:	Event analysis Routine
5.85.2	Function:	Calculate and check the fast/slow ratio
5.85.3	Interfaces:	
5.85.4	Dependencies:	Called from analyse
		Called from datadump
5.85.5	Processing:	Caculate the ratio
		Determine interpolation range
		Calculate upper limit
		Calculate lower limit
		Compare with limits
		Flag result
5.85.6	Data:	
5.85.7	Resources	calindex Routine
		calinter Routine
5.85.8	Memory Location:	B450 - B494

5.86	Component Identifier:	calindex
5.86.1	Type:	Event analysis Routine
5.86.2	Function:	Calculate the interpolation point
5.86.3	Interfaces:	
5.86.4	Dependencies:	Called from fastslow
5.86.5	Processing:	Divide energy with interval Calculate index Calculate residual
5.86.6	Data:	
5.86.7	Resources	
5.86.8	Memory Location:	B495 - B49E

5.87	Component Identifier:	calinter
5.87.1	Type:	Event analysis Routine
5.87.2	Function:	Interpolation Routine
5.87.3	Interfaces:	
5.87.4	Dependencies:	Called from fastslow
		Called from catcheck
		Called from vetochecck
		Called from backcheck
5.87.5	Processing:	Get start position
		Get start value
		Get interpolation weight
		Multiply with residual
		Add to start value
5.87.6	Data:	
5.87.7	Resources	
5.87.8	Memory Location:	B49F - B4AC

5.88	Component Identifier:	cflexpos
5.88.1	Type:	Event analysis Routine
5.88.2	Function:	Correct cathode non linearity
5.88.3	Interfaces:	
5.88.4	Dependencies:	Called from analyse Called from datadump
5.88.5	Processing:	Correct for sign If close to readout expand scale If far from readout contract scale
5.88.6	Data:	
5.88.7	Resources	
5.88.8	Memory Location:	B4AF - B4C9

5.89	Component Identifier:	polcorr
5.89.1	Type:	Event analysis Routine
5.89.2	Function:	Calculate energy and backplane corrections
5.89.3	Interfaces:	
5.89.4	Dependencies:	Called from analyse Called from datadump
5.89.5	Processing:	Get fractional cathode hit point Calculate polynomium fit to correction
5.89.6	Data:	
5.89.7	Resources	
5.89.8	Memory Location:	B4CA - B4E1

5.90	Component Identifier:	ancatwgt
5.90.1	Type:	Event analysis Routine
5.90.2	Function:	Calculate anode-cathode correction
5.90.3	Interfaces:	
5.90.4	Dependencies:	Called from anodcoup
5.90.5	Processing:	Calculate left right difference Divide with anode value Multiply with 25 Scale between 0 and 1
5.90.6	Data:	
5.90.7	Resources	
5.90.8	Memory Location:	B4E2 - B4FA

5.91	Component Identifier:	anodcoup
5.91.1	Type:	Event analysis Routine
5.91.2	Function:	Calculate anode cathode coupling
5.91.3	Interfaces:	
5.91.4	Dependencies:	Called from analyse
		Called from datadump
5.91.5	Processing:	Get farleft and farright values
		Calculate weight
		Correct three center cathode values
5.91.6	Data:	
5.91.7	Resources	ancatwgt Routine
		calcorr Routine
5.91.8	Memory Location:	B4FB - B52A

5.92	Component Identifier:	calcorr
5.92.1	Type:	Event analysis routine
5.92.2	Function:	Correct one cathode value
5.92.3	Interfaces:	
5.92.4	Dependencies:	Called from anodcoup
5.92.5	Processing:	Calculate correction factor Calculate left correcction Calculate right correction Sum the corrections
5.92.6	Data:	
5.92.7	Resources	
5.92.8	Memory Location:	B52B - B536

5.93	Component Identifier:	hitresultc
5.93.1	Type:	Event analysis Routine
5.93.2	Function:	Calculate cathode positon in mm*4
5.93.3	Interfaces:	
5.93.4	Dependencies:	Called from analyse Called from datadump
5.93.5	Processing:	Get hit position Convert to strips Convert strips to mm*4 Check and flag range
5.93.6	Data:	
5.93.7	Resources	
5.93.8	Memory Location:	B537 - B563

5.94	Component Identifier:	hitresultb
5.94.1	Type:	Event analysis Routine
5.94.2	Function:	Calculate backplane position in mm*4
5.94.3	Interfaces:	
5.94.4	Dependencies:	Called from analyse Called from datadump
5.94.5	Processing:	Get hit position Convert to strips Convert strips to mm*4 Check and flag range
5.94.6	Data:	
5.94.7	Resources	
5.94.8	Memory Location:	B564 - B58E

5.95	Component Identifier:	catindex
5.95.1	Type:	Event analysis Routine
5.95.2	Function:	Calculate cathode index
5.95.3	Interfaces:	
5.95.4	Dependencies:	Called from analyse Called from datadump
5.95.5	Processing:	Get hit position Multiply with 5 Keep in range
5.95.6	Data:	
5.95.7	Resources	
5.95.8	Memory Location:	B58F - B5A4

5.96	Component Identifier:	backindex
5.96.1	Type:	Event analysis Routine
5.96.2	Function:	Calculate backplane index
5.96.3	Interfaces:	
5.96.4	Dependencies:	Called from analyse
		Called from datadump
5.96.5	Processing:	Get hit position
		Multiply with 2
		Keep in range
5.96.6	Data:	
5.96.7	Resources	
5.96.8	Memory Location:	B5A5 - B5BA

5.97	Component Identifier:	catcheck
5.97.1	Type:	Event analysis Routine
5.97.2	Function:	Check cathode anode ratio
5.97.3	Interfaces:	
5.97.4	Dependencies:	Called from analyse
		Called from datadump
5.97.5	Processing:	Calculate interpolation index
		Calculate upper limit
		Calculate lower limit
		Check and flag range
5.97.6	Data:	
5.97.7	Resources	calinter Routine
5.97.8	Memory Location:	B5BB - B5DC

5.98	Component Identifier:	vetocheck
5.98.1	Type:	Event analysis routine
5.98.2	Function:	Check veto anode ratio
5.98.3	Interfaces:	
5.98.4	Dependencies:	Called from analyse
		Called from datadump
5.98.5	Processing:	Calculate interpolation index
		Calculate upper limit
		Calculate lower limit
		Check and flag range
5.98.6	Data:	
5.98.7	Resources	calinter Routine
5.98.8	Memory Location:	B5DD - B5FE

5.99	Component Identifier:	backcheck
5.99.1	Type:	Event analysis Routine
5.99.2	Function:	Check backplane anode ratio
5.99.3	Interfaces:	
5.99.4	Dependencies:	Called from analyse
		Called from datadump
5.99.5	Processing:	Calculate interpolation index
		Calculate upper limit
		Calculate lower limit
		Check and flag range
5.99.6	Data:	
5.99.7	Resources	calinter Routine
5.99.8	Memory Location:	B5FF - B620