Digital Gamma Finder (DGF)

Differences Between
DGF-4 Rev. E, DGF-4C Rev. F and Pixie-4

(For Programmers)

Version 1.2, July 2009

XIA LLC

31057 Genstar Road
Hayward, CA 94544 USA

Phone: (510) 401-5760; Fax: (510) 401-5761
http://www.xia.com

1 Overview

The Digital Gamma Finder (DGF) family of digital pulse processors includes several instruments: Standalone single-channel units (Polaris, Gamma200), multi-channel CAMAC modules (DGF-4C Rev. C, D, E, F), and multi-channel PXI modules (Pixie-4 and Pixie-16). They follow the same basic architecture and operate with similar software. Historically, DGF-4C modules have been developed first, branched out to the single channel models, and have later been redesigned as PXI modules to take advantage of the modern, high speed interface. The most recent development is the DGF-4C Rev. F, in which essentially the Pixie-4 design is adapted back to a CAMAC form factor, with an additional USB interface for high speed readout.

This document describes the differences between DGF Rev. E, Rev. F and the Pixie-4. The emphasis is placed on describing changes in the DGF Rev. F compared to the other models. A full manual for each model is provided separately.

Changes to the original (2007) version made in 2008 are highlighted in gray
Changes made in July 2009 are highlighted yellow

2 Hardware

2.1 Summary

The board hardware of the DGF Rev. F consists of the same basic building blocks as the DGF Rev. E and the Pixie-4: An analog section to adjust gain and offset, a high speed ADC to digitize the signal, an FPGA for triggering and filtering, a DSP to reconstruct the pulse height, and a "System FPGA" to control external MCA memory and act as interface to a host computer. The differences are listed in table 1:

	Pixie-4	**DGF Rev. E**	**DGF Rev. F**
Input impedance	50 Ohm or 5k Ohm (jumper)	50Ω, 250Ω and 1kΩ (jumpers)	50 Ohm or 5k Ohm (jumper)
Input attenuation	1:7.5 (for 50 Ohm), 1:1	1:21, 1:5 and 1:1	1:7.5 (for 50 Ohm), 1:1
Offset	-2.5V … +2.5V	-3V … +3V	-2.5V … +2.5V
Gain	0.965 … 11.25 (2^6 steps by relays, 10% digital adjustment of computed energy for gain matching)	1 … 100 x [0.16.] (2^{16} steps by DAC controlled variable gain amplifier)	0.965 … 11.25 (2^6 steps by relays, 10% digital adjustment of computed energy for gain matching)
ADC	14 bits, 75 MHz	14 bits, 40 MHz	14 bits, 80 MHz
FPGA	Spartan 2	Spartan XL	Spartan 2
DSP	ADSP2185M	ADSP2183	ADSP2185N
MCA memory	128k x 32 bit	128k x 24 bit	128k x 32 bit
List mode memory	128k x 32 bit	--	128k x 32 bit
Host interface	PCI	CAMAC	CAMAC, USB
Clock, trigger distribution	PXI backplane	CAMAC auxiliary connectors (header and FW)	CAMAC auxiliary connectors (header and FW)
Run synchronization	PXI backplane	Front panel LEMO connectors	Front panel LEMO connectors
Digital I/O	2 MMCX connectors (GFLT, Status)	7 LEMO connectors (GFLT, GSLT, Multiplicity or Sum, Busy/Sync, Trigger)	11 LEMO connectors (GFLT, GSLT, Multiplicity, Busy/Sync, Trigger, 4 channel Gates)

Table 1: Hardware differences

Notes:

- In the DGF Rev. E, the GSLT input is connected to a DSP interrupt which records a timestamp. In the DGF Rev. F, it is connected to the System FPGA with no defined function (yet), it can not create interrupts.
- In the DGF Rev. E, the Multiplicity Output can be connected to either the sum of multiplicity signals (fast triggers) or the analog sum of the channel inputs. This second option is not available in the Rev. F.
- The DSP on all three instruments chips are code compatible, and the DSP code for data acquisition follows the same architecture. User plug in code should be compatible as long as no external memory is accessed ("io" command).

2.2 Clock Distribution

The clock distribution on DGF Rev. F and Rev. E is identical; both kinds of modules use an auxiliary header on the rear to distribute 40 MHz clock signals from slot to slot (or alternatively use a 4-pin Firewire (FW) connector as input). However, in the DGF Rev. F the distributed clock is doubled in the FPGAs and DSP on each board to obtain the 80 MHz sampling and processing frequency. Clock mode and termination jumpers have different PCB references, but the same function. Jumpers should be set according to module function and position as listed in table 2.

In the Pixie-4, 37.5 MHz clocks are distributed over the PXI backplane. Differential clock signals can be brought out via a PXI PDM module, but the signal standard is LVDS, not directly compatible with the PECL signals of the DGF.

Module	**E: JP1, JP2** **F: JP404, JP405**	**E: JP3, JP4** **F: JP406, JP407**	**E : JP5** **F: JP403**	**Crate Position**
Clock Master	Yes	No	“board clock”	Most right
Clock Repeater	No	Yes	“external”	Middle
Clock Terminator	No	Yes	“external”	Most left
Standalone	Yes	Yes	“board clock”	Any
FW input	No	Yes	“external”	Any

Table 2: DGF clock jumper settings

2.3 Trigger Distribution

The trigger distribution on DGF Rev. F and Rev. E is identical; both kinds of modules use the auxiliary header on the rear to distribute trigger signals (PECL standard) from slot to slot. “Left” and “right” pins on the header are connected on the board so that all modules in the chain are connected to the same signals. However, Rev. F modules pass the trigger signals through the System FPGA, so that a module can be disconnected from backplane triggers by setting a control bit in ModCSRA.

2.4 Power

The DGF Rev. F uses approximately 4.1A on +6V and 0.3A on -6V.

2.5 Front Panel Connections

From top to bottom, the DGF F front panel has the following connections:

- LED connected to CAMAC N line
- Analog input for channel 3
- Analog input for channel 2
- Analog input for channel 1
- Analog input for channel 0
- Gate input for channel 3 (NIM logic level)
- Gate input for channel 2 (NIM logic level)
- Gate input for channel 1 (NIM logic level)
- Gate input for channel 0 (NIM logic level)
- “Trigout” output (NIM logic level)
- “Busy” output (NIM logic level)
- “GFLT” input (NIM logic level)
- “GSLT” input (NIM logic level)
- “Sync” input (NIM logic level)
- Multiplicity output (analog)
- Multiplicity input (analog)

2.6 Other Jumpers

Additional jumpers on the models have the following functions and equivalences:

Function	Pixie-4	DGF Rev. E	DGF Rev. F	Notes
Analog input termination and attenuation	**JPx01**: remove for 1:7.5 attenuation (if JPx02 is set) **JPx02**: set for 50 Ohm input impedance, else 5k Ohm	**JPx00**: remove for attenuation (1:2,1:5, or 1:21 if JPx01,2,3 is set) **JPx01**: set for 1kΩ **JPx02**: set for 250Ω **JPx03**: set for 50Ω (remove others)	**JPx01**: remove for 1:7.5 attenuation (if JPx02 is set) **JPx02**: set for 50 Ohm input impedance, else 5k Ohm	x = 1..4
Multiplicity or analog sum on front panel MULT OUT	No analog multiplicity output	Connect **JP10-13** if channel 0-3 should contribute to analog sum Connect **JP14-17** if channel 0-3 should contribute to multiplicity	Always connected to multiplicity (can still enable/disable contribution in software)	
Compare multiplicity or analog sum to SUMDAC and issue GFLT	No analog multiplicity output	Set JP18	Set **JP408**	
USB/System clock selection			JP410	Must be set to "SYS"
Testpoints only			**JP409, JP427**	

Table 3: Other Jumpers

2.7 USB connection

The USB interface on the DGF Rev. F is implemented in a Cypress EZ-USB FX2LP chip (P/N CY7C68013A-100AXC). It is programmed by XIA to identify itself as Vendor ID 0x10E9 (=XIA) and product ID 0x0600 (=DGF Rev. F).

To read out the module through the USB interface, connect the "Mini-USB" connector on the rear to a USB 2.0 port of the PC. This connection is hot-pluggable, i.e the connection can be made or unmade any time when the CAMAC chassis is powered. However, at least when using the XIA Igor interface, the connection must be present when booting the module.

3 DSP Code and Parameters

Since the DSP chips in all models are from the same code-compatible family, the DSP code is largely identical. The Pixie-4 and the DGF Rev. F use the same code (with a few compiler switches for model specific functions, e.g. where frequency is important), which was originally derived from the DGF Rev. E code. A list of DSP variables is given in Tables 4-7. Differences between DGF Rev. E and DGF Rev. F are highlighted, also changes in the DGF F/Pixie-4 DSP code version 3.9 and higher compared to previous versions of Pixie-4 code. Variables are listed in order of DSP memory, but as always software should refer to the .var file to find the location of a variable to be compatible with future changes. After each table, those variables requiring further explanation are described in detail.

3.1 Module Input Variables

DGF Rev. E	**DGF Rev. F/Pixie-4 (version 3.9)**	**Notes**
MODNUM	MODNUM	
MODCSRA	MODCSRA	Additional bits used Bit 0 unused in version 3.9+
MODCSRB	MODCSRB	
MODFORMAT	MODFORMAT	
SUMDAC	SUMDAC	Not present in Pixie-4 code version 3.4 Never modified by Rev. E Clib or Igor
RUNTASK	RUNTASK	No more fast list mode runs 0x20n
CONTROLTASK	CONTROLTASK	Tasks 9-21 not supported. New tasks 22, 25, 26.
MAXEVENTS	MAXEVENTS	
COINCPATTERN	COINCPATTERN	
COINCWAIT	COINCWAIT	
SYNCHWAIT	SYNCHWAIT	
INSYNCH	INSYNCH	
HOSTIO	HOSTIO	
	RESUME	Used instead of CSR to indicate new/resumed run Not set by Igor
	FILTERRANGE	Input parameter to select decimation
	MODULEPATTERN	Pixie-4 only: module coincidence
	NNSHAREPATTERN	Pixie-4 only: module coincidence
	CHANNUM	Channel number for some control tasks Not set by Igor
	MODCSRC	Reserved for future use
	DBLBUFCSR	For double (ping pong) buffer control
U00	U00	
XDATLENGTH	XDATLENGTH	
USERIN	USERIN	

Table 4. Module input variables for DSP code

<u>Details:</u>

MODCSRA0:

Bit	Function	Notes
0	Local Time Stamp	If set, record each channel's time stamp in group trigger mode, else set all channels to common time stamp. Useful to retain trigger time differences when not recoding traces. No longer used. Moved to CHANCSRA bit 13 as a channel option
1	LM data to external memory	If set, whenever an 8k DSP buffer is full, data is transferred to external memory. The acquisition halts after 32 buffers until host reads and resumes. **Must be set/cleared for all modules in the system. If set, clear bit 0 of DBLBUFCSR**
2	Backplane connect	If set, group triggers are shared over backplane or aux. connector, else only within module
3	Clover addback	In events with more than one channel, sum energy is binned into sum spectrum. MCA memory is rearranged into four 16K channel bocks and one 16K sum block.
4	Clover addonly	Only exclusive singles are binned into channel spectra.
5	Front input to GFLT	Pixie-4's front panel input puts GFLT signal on backplane
6	NN transmit to right	Unused Formerly Pixie-4 only: pass on module coincidence signals to right neighbor
7	NN transmit to left	Unused Formerly Pixie-4 only: pass on module coincidence signals to left neighbor
8	Enable module coincidence	Unused Formerly Pixie-4 only: share coincidence information with other module over backplane
9	MC left	Pixie-4 only: send module coincidence signals to PDM
10	MC right/ Switchbus 2	Pixie-4 unused. Formerly : send module coincidence signals to right neighbor DGF only: Switchbus 2 for trigger termination
11	MC status	Pixie-4 unused. Formerly: send module concidence signals to status line
12	MC token	Pixie-4 only: send coincidence signals to token line
13	MC star/ Switchbus5	Pixie-4 only: send coincidence signals to star trigger line DGF only: Switchbus 5 for trigger termination
14	Front input to status	Pixie-4's front panel input puts signal on status line on backplane
15	NN bus	Pixie-4 only Enable NN triggering across PXI segment boundaries (with bit 2)

Notes:

- In DGF Rev. F, the switchbus bits are applied by the DSP as part of the "ProgramFippi" controltask. No direct write to the ICSR is required as in the DGF Rev. E.

- In the DGF E C library, the whole of MODCSRA was written to the ICSR to set the switchbus bits. However, ICRS's lower bits control FPGA programming. Luckily, MODCSRA lower bits were always zero, so no accidental rebooting occurred. In Rev. F, lower bits can be 1, so make sure to only write (MODCSRA & 0x2400) to ICSR, but as per previous comment, no ICSR write is required in Rev. F to set the switchbus bits.
- Module coincidence control for the Pixie-4 is planned to be revised/improved in the future

RESUME:
Prior to runstart, set this variable to 0 to resume a data run; otherwise, set it to 1 to start a new run. Set to 2 before stopping a list mode run prematurely.
At the end of a run, it is set to zero by the DSP to start the following run as a "resume run" by default.

FILTERRANGE:
The energy filter range downloaded from the host to the DSP. It sets the number of ADC samples (2^FILTERRANGE) to be averaged before entering the filtering logic. The currently supported filter range in the signal processing FPGA is 1 - 6.

CHANNUM:
The chosen channel number. May be modified internally for tasks looping over all 4 channels, or to pass on current channel to user code. Should be set by host before starting controltask 4 and 6 to indicate which channel to operate on. (Previously HOSTIO was used in controltask 4). We recommend to always change CHANNUM when changing the channel that is addressed in the user interface.

DBLBUFCSR:
A register containing several bits to control the double buffer (ping pong) mode to read out external memory. In the future, these control bits may be moved to the CSR register in the System FPGA.

Bit	Function	Notes
0	Enable double buffer	If this bit is set, transfer list mode data to external memory in double buffer mode. **Must be set/cleared for all modules in the system. If set, clear bit 1 of MODCSRA** Set by host, read by DSP
1	Host read	Host sets this bit after reading a block from external memory to indicate DSP can write into it again. Set by host, read and cleared by DSP
2	reserved	
3	Read_128K_first	If run halted because host did not read fast enough and both blocks in external memory are filled, DSP will set this bit to indicate host to first read from block 1 (staring at address128K) or from block 2. Set by DSP, read by host. Cleared by DSP at runstart or resume

3.2 Channel Input Variables

DGF Rev. E	**DGF Rev. F/Pixie-4 (version 3.9)**	**Notes**
CHANCSRA0	CHANCSRA0	Additional bits used Bit 13: local timestamp
CHANCSRB0	CHANCSRB0	
GAINDAC0	GAINDAC0	Ignored in Rev. F/Pixie-4
TRACKDAC0	TRACKDAC0	
	SGA0	Bit pattern for gain relays
	DIGGAIN0	Multiplier for energy
UNUSEDA0	UNUSEDA0	
SLOWLENGTH0	SLOWLENGTH0	New limit: SL+SG <= 127
SLOWGAP0	SLOWGAP0	New limit: SL+SG <= 127
FASTLENGTH0	FASTLENGTH0	New limit: SL+SG <= 63
FASTGAP0	FASTGAP0	New limit: SL+SG <= 63
PEAKSAMPLE0	PEAKSAMPLE0	New dependency
PEAKSEP0	PEAKSEP0	New dependency
FASTADCTHR0	USERDELAY	Replaces unused parameter FASTADCTHR
FASTTHRESH0	FASTTHRESH0	
MINWIDTH0	MINWIDTH0	
MAXWIDTH0	MAXWIDTH0	
PAFLENGTH0	PAFLENGTH0	New dependency
TRIGGERDELAY0	TRIGGERDELAY0	New dependency
RESETDELAY0	RESETDELAY0	
FTPWIDTH0	FTPWIDTH0	
TRACELENGTH0	TRACELENGTH0	
XWAIT0	XWAIT0	
ENERGYLOW0	ENERGYLOW0	
LOG2EBIN0	LOG2EBIN0	
CFDTHR0	CFDTHR0	
PSAOFFSET0	PSAOFFSET0	
PSALENGTH0	PSALENGTH0	
INTEGRATOR0	INTEGRATOR0	New option 3-5 in Rev. F/Pixie-4 No longer directly downloaded by Igor to DSP, bypassing Clib
BLCUT0	BLCUT0	
BASELINEPERCENT0	BASELINEPERCENT0	
	XAVG0	Used for averaging samples in Controltask 4 Not set by Igor
	CHANCSRC0	Additional bits used
	GATEWINDOW	Coincidence Window for GATE signal
	GATEDELAY	Delay for latching GATE signal after fast trigger
UNUSEDB0	UNUSEDB0	
CFDREG0	CFDREG0	
LOG2BWEIGHT0	LOG2BWEIGHT0	
PREAMPTAUA0	PREAMPTAUA0	
PREAMPTAUB0	PREAMPTAUB0	

Table 5: Channel input variables for DSP code

<u>Details:</u>

CHANCSRA:

Bit	Function	Notes
0	Group trigger	No change
1	Reserved	Was "individual live time"
2	Good channel	No change
3	Read always	No change
4	Enable trigger	No change
5	Trigger positive	No change
6	GFLT required	No change
7	Histogram energies	No change
8	Reserved	No change
9	Allow E<0	Was reserved. If set, allow negative number as result of energy computation in DSP, else negative energies are set to zero. List mode runs only.
10	CFD timing (in DSP)	No change
11	Enable multiplicity	DGF E/F only
12	Channel gate required	DGF F only: If set, accept only pulses for which a channel's GATE input is logic 1 (-1V) ; else only store GATE bit in event hit pattern
13	Local time	Was reserved. If set, use the local trigger to latch the time stamp even in group trigger mode, else use the distributed group trigger.
14	Estimate energy if not hit	If set, the DSP reads out energy filter values and computes the pulse height for a channel that is not hit; else then pulse height will be set to zero. Useful to get energies for pulses below trigger threshold.
15	Reserved	No change

SGA:
The index of the relay combinations of the switchable gain amplifier. For a given value of SGA, the analog gain is G = (1+Rf/Rg)/2 with

Rf = 2150 - 120*((SGA & 0x1)>0) - 270*((SGA &0x2)>0) - 560*((SGA &0x4)>0)
Rg = 1320 - 100*((SGA & 0x10)>0) - 300*((SGA & 0x20)>0)- 820*((SGA & 0x40)>0)

DIGGAIN:
The digital gain factor for compensating the difference between the user-desired voltage gain and the SGA gain. The energy computed from the raw filter sums E_F (proportional to analog pulse height) will be modified into the energy reported E_R as follows:

$E_R = E_F + E_F$ * DIGGAIN / 65536

It is recommended to keep DIGGAIN < 6554 to avoid potential binning effects.

PEAKSAMPLE, PEAKSEP

The values of PEAKSAMPLE and PEAKSEP are computed from SLOWLENGTH (SL) and SLOWGAP as follows:

```
      if(FILTERRANGE==0) {
         PEAKSAMPLE=SL+SG-7;
         PEAKSAMPLE =max(0, PEAKSAMPLE); /* keep it greater than 0 */
         PEAKSEP=PeakSample+5;
      }
      if(FILTERRANGE ==1) {
         PEAKSAMPLE =SL+SG-4;
         PEAKSAMPLE =max(2, PEAKSAMPLE); /* keep it greater than 1 */
         PEAKSEP = PEAKSAMPLE +5;
      }
      if(FILTERRANGE ==2) {
         PEAKSAMPLE =SL+SG-2;
         PEAKSEP = PEAKSAMPLE +5;
      }
      if(FILTERRANGE ==3) {
         PEAKSAMPLE =SL+SG-1;
         PEAKSEP = PEAKSAMPLE +5;
      }
      if(FILTERRANGE ==4) {
         PEAKSAMPLE =SL+SG-1;
         PEAKSEP = PEAKSAMPLE +5;
      }
      if(FILTERRANGE >=5) {
         PEAKSAMPLE =SL+SG-1;
         PEAKSEP = PEAKSAMPLE +5;
      }
      if(PEAKSEP >128) {
         PEAKSEP = PEAKSAMPLE +1;
      }
      if((PEAKSEP - PEAKSAMPLE)>7) {
         PEAKSEP = PEAKSAMPLE +7;
      }
```

PAFLENGTH, TRIGGERDELAY, USERDELAY

PAFLENGTH and TRIGGERDELAY are computed from TRACEDELAY, FILTER-RANGE, PEAKSEP, and FIFOLENGTH (=1k) as follows:

```
      if(TRACELENGTH>0)
            TRIGGERDELAY = (PEAKSEP-1)*(2^FILTERRANGE);
      else
            TRIGGERDELAY =1;

      PAFLENGTH = TRIGGERDELAY +TRACEDELAY;
      if(PAFLENGTH>(FIFOLENGTH)){
            PAFLENGTH = FIFOLENGTH -1;
            TRIGGERDELAY = PAFLENGTH - TRACEDELAY;
      }
```

The new variable USERDELAY has to be set equal to TRACEDELAY (the pre-trigger length of the waveform)

INTEGRATOR:
Setting INTEGRATOR0 to 3, 4 or 5 is the same as setting it to 1, but the computed energy is multiplied by a factor or 2, 4 or 8, respectively.

XAVG:
Only used in Controltask 4 for reading untriggered traces. XAVG stores the weight in the geometric-weight averaging scheme to remove higher frequency signal and noise components. The value is calculated as follows:

For a given sampling interval dt (in us), calculate the integer intdt = dt/0.0133
If intdt>13, XAVG = floor(65536/((intdt-3)/5))
If intdt<=13, XAVG = 65535.

CHANCSRC:

Bit	Function	Notes
0	GFLT polarity	Optional inversion of GFLT input signal before being used in event validation
1	GATE acceptance polarity	Optional inversion of GATE status before being used in event validation
2	Use GFLT for GATE	If set, use GFLT input for fast validation of signal rising edge of pulse
3	Disable pileup inspection	If set, use GFLT input for fast validation of signal rising edge of pulse
4	Disable out-of-range rejection	If set, pulses are accepted even if the ADC input goes out of range
5	Invert pileup inspection	If set, only accept events with pileup
6	Pause pileup inspection	If set, disable pileup inspection for 32 clock cycles (426 ns).
7	GATE edge polarity inversion	Optional inversion of GATE input signal before starting GATE window on rising edge
8-15	Reserved	No change

GATEDELAY, GATEWINDOW:
These variables set the coincidence window for the Gate signal to reject events. At the rising edge of the Gate signal, and internal Gate status bit goes high for the duration of GATEWINDOW. A GATEDELAY after a fast trigger the status bit is latched into GATEBIT. GATEBIT can be used to reject events in the FPGA, and it is reported in the hit pattern in the list mode data stream for offline processing if no online rejection is desirable.

3.3 Module Output Variables

DGF Rev. E	DGF Rev. F/Pixie-4 (version 3.9)	Notes
DECIMATION	DECIMATION	Decimation is not read from Fippi, it is set by Filterrange This is a copy of the value in Filterrange to improve backwards compatibility.
REALTIMEA	REALTIMEA	

REALTIMEB	REALTIMEB	
REALTIMEC	REALTIMEC	
RUNTIMEA	RUNTIMEA	
RUNTIMEB	RUNTIMEB	
RUNTIMEC	RUNTIMEC	
GSLTTIMEA	GSLTTIMEA	
GSLTTIMEB	GSLTTIMEB	
GSLTTIMEC	GSLTTIMEC	
NUMEVENTSA	NUMEVENTSA	
NUMEVENTSB	NUMEVENTSB	
DSPERROR	DSPERROR	
SYNCHDONE	SYNCHDONE	
TEMPERATURE	TEMPERATURE	
BUFHEADLEN	BUFHEADLEN	
EVENTHEADLEN	EVENTHEADLEN	
CHANHEADLEN	CHANHEADLEN	
	EMWORDS	Number of 16 bit words in external memory
	EMWORDS2	Number of 16 bit words in external memory
	TOTALTIMEA	Closest to the true lab time passed since the most recent "new run" command (the first spill in a series)
	TOTALTIMEB	
	TOTALTIMEC	
U14	U14	
USEROUT	USEROUT	
AOUTBUFFER	AOUTBUFFER	This address changed to accommodate an almost 4k intermediate buffer for compressed list mode runs (was 2k).
LOUTBUFFER	LOUTBUFFER	
AECORR	U15	AECORR removed
LECORR		LECORR removed
ATCORR		ATCORR removed
LTCORR		LTCORR removed
HARDWAREID	HARDWAREID	
HARDVARIANT	HARDVARIANT	
FIFOLENGTH	FIFOLENGTH	
FIPPIID	FIPPIID	
FIPPIVARIANT	FIPPIVARIANT	
INTRFCID	INTRFCID	
INTRFCVARIANT	INTRFCVARIANT	
DSPRELEASE	DSPRELEASE	
DSPBUILD	DSPBUILD	

Table 6: Module output variables

<u>Details:</u>

EMWORDS:
In list mode runs with the 32x buffer option enabled, EMWORDS contains the number of 16 bit words in external memory ready to be read out after 32 buffers are transferred. In double buffer runs, EMWORDS and EMWORDS2 contain the number of 16 bit words ready to be

read out in list mode block 1 and list mode block 2 of the external memory, respectively. See section “Data Acquisition” for details.

TOTALTIMEA, TOTALTIMEB, TOTALTIMEC:
A 48-bit clock to track the total time an acquisition was requested by the host. RUNTIME excludes the time waiting for host readout, TOTALTIME is the closest to the true lab time passed since the most recent “new run” command (the first spill in a series). A,B,C words are as for the RealTime clock. Compute the total time using the following formula:
TotalTime =(TOTALTIMEA * 64K^2 + TOTALTIMEB * 64K + TOTALTIMEC) * 12.5ns

3.4 Channel Output Variables

DGF Rev. E	DGF Rev. F/Pixie-4 (version 3.9)	Notes
LIVETIMEA0	LIVETIMEA0	Change in the way livetime is counted
LIVETIMEB0	LIVETIMEB0	
LIVETIMEC0	LIVETIMEC0	
FASTPEAKSA0	FASTPEAKSA0	
FASTPEAKSB0	FASTPEAKSB0	
OVERFLOWA0		Removed OVERFLOWA0
OVERFLOWB0		Removed OVERFLOWB0
INSPECA0		Removed INSPECA0
INSPECB0		Removed INSPECB0
UNDERFLOWA0		Removed UNDERFLOWA0
UNDERFLOWB0		Removed UNDERFLOWB0
ADCPERDACA0		Removed ADCPERDACA0
ADCPERDACB0		Removed ADCPERDACB0
	NOUTA0	(counting channel OCR)
	NOUTA0	(counting channel OCR)
	FTDTA0	Fast trigger dead time
	FTDTB0	
	FTDTC0	
	SFDTA0	Slow filter dead time
	SFDTB0	
	SFDTC0	
	GCOUNTA0	Gate count rate
	GCOUNTB0	
	GDTA0	Gate time
	GDTB0	
	GDTC0	
	ICR0	Current input count rate
	OORF0	Current out-of-range fraction
UNUSEDC0	UNUSEDC0	

Table 7: Channel output variables

FTDTA, FTDTB, FTDTC:
Fast Trigger dead time is the time the fast trigger output was above threshold and thus not ready to detect further triggers, as measured by the trigger/filter FPGA. See the user manual

for a description of the measured time. Convert the three words into a time using the formula (note missing factor 16):
FTDT = (FTDTA *64K^2 + FTDTB *64K + FTDTC) *12.5ns

SFDTA, SFDTB, SFDTC:
Slow Filter Dead Time is the time the associated with each pulse that prohibited acquisition of a second pulse, for example due to pileup inspection or DSP readout. See the user manual for a description of the measured time. Convert the three words into a time using the formula:
SFDT = (SFDTA *64K^2 + SFDTB *64K + SFDTC) *16 * 12.5ns

GCOUNTA, GCOUNTB:
The number of gate pulses for this channel (high, low)
GCOUNT = GCOUNTA *65536 + GCOUNTB

NOUTA, NOUTB:
The number of output counts in this channel (high, low)
NOUT = NOUTA *65536 + NOUTB

GDTA, GDTB, GDTC:Gate Dead Time is the time during which a channel was gated. See the user manual for a description of the measured time. Convert the three words into a time using the formula:
GDT = (GDTA *64K^2 + GDTB *64K + GDTC) *16 * 12.5ns

ICR: ICR is an averaged measure of the current input count rate. It is updated if a run is in progress or not. The averaging is implemented such that at every update,

$\text{Average}_{\text{new}}$ = ($\text{Average}_{\text{old}}$ + Number fast triggers in update period)/2

The value reported in the variable ICR is equal to 2*$\text{Average}_{\text{new}}$. Updates occur every 32*64K clock cycles. Thus to compute the rate in counts/s, the value in ICR has to be divided by 32*64K * 12.5ns. The reported value is precise to about 50 counts/s, with a maximum count rate of about one million counts/s

OORF: OORF is an averaged measure of the fraction of time the channel is out of range. It is updated if a run is in progress or not. The averaging is implemented such that at every update,

$\text{Average}_{\text{new}}$ = ($\text{Average}_{\text{old}}$ + Time out of range/64)/2

The value reported in the variable OORF is equal to 2*$\text{Average}_{\text{new}}$. Updates occur every 32*64K clock cycles. Thus to compute the out of range fraction in percent, the value in OORF has to be multiplied by (100% / 64K).

3.5 User Code

User code should be upwards compatible between Pixie-4, DGF E and DGF F; newer code may give access to more variables in the intrface.inc file. Access to external memory or registers (using “io” statements) is different in DGF E compared to Pixie-4 and DGF F (but normally not supported for user code).

4 Firmware

4.1 Firmware Files

The DGF Rev. D/E uses Xilinx Spartan XL FPGAs for the System FPGA and the trigger/filter FPGAs. There are several firmware files for the trigger/filter FPGAs for different decimation (averaging samples before processing in order to increase filter times), and for each decimation there is a specific file for Rev. D and for Rev. E. To select a decimation, different files have to be downloaded into the FPGA. There is only one System FPGA file, used for both Rev. D and Rev. E.

The DGF Rev. F and the Pixie-4 use Spartan2 FPGAs. Here, the decimation in the trigger/filter FPGA is a user parameter; always the same file is downloaded into the FPGA and the DSP applies the parameter FILTERRANGE to select a decimation. The same trigger/filter FPGA file is used for both DGF Rev. F and Pixie-4, but the System FPGA is different.

In general, the firmware files for should be assumed to be of different size (number of data words). However, for the Rev. F both System and Fippi FPGA are 166980 bytes. In some Windows versions the Jorway driver does seem to be limited to <64K words in a block write, so the XIA C library downloads data in blocks of 32K words max.

4.2 Config FPGA Registers for Host

The Config or Interface FPGA is configured from a PROM at power up. It is then used to configure the other FPGAs with data from the host computer. There are 4 registers in the Config FPGA relevant for host I/O operations. The booting procedure is described in a later section.

Version Register:

The Config FPGA's version register can be read by the host to obtain hardware information. It is a 16 bit register; the individual bits are defined in the table below. Since the FPGA boots from a PROM, the information in this register is valid to identify the hardware.

	Pixie-4	**DGF Rev. E**	**DGF Rev. F**	**Description**
Address/ command	**0000000C (read)**	**F1, A13 (read)**	**F1, A13 (read)**	
Bit 0-3	0x2	0x4	0x5	Revision
Bit 4-7	0x0 = 1K FIFO	0x1= 4K FIFO	0x0 = 1K FIFO	Variant
Bit 8-11	0x7	0x5	0x5	Type
Bit 12-15	0xA	0xA	0xA	Product ID

Interface Control/Status Register (ICSR):
The Config FPGA's ICSR register is used to program the other FPGFAs

	Pixie-4	**DGF Rev. E**	**DGF Rev. F**	**Notes**
Address/ command	**00000004 (write) 00000008 (read)**	**F17, A8 (write) F1, A8 (read)**	**F17, A8 (write) F1, A8 (read)**	
Bit 0	Reset System	Reset System	Reset System	
1	Reset Fippi I	--	--	
2	Reset Fippi II	--	--	
3	?	--	--	
4	?	Reset Fippi0	Reset Fippi I	
5	?	Reset Fippi1		
6	--	Reset Fippi2	Reset Fippi II	
7	--	Reset Fippi3		
8	--	--	--	
9	--	--	--	
10	--	Switchbus2	--	
11	--	--	--	
12	--	--	--	
13	--	Switchbus5	--	
14	--	--	--	
15	--	--	--	

Notes:
- In bits 0-7, Write "1" to reset, check if "0" after configuring.
- For DGF Rev. F, writing to bit 5 (and 7) is ignored, reading from bit 5 (7) obtains the duplicated value from bit 4 (6).
- In DGF Rev. F, the Switchbus bits are applied by DSP, no need to write to ICSR
- The Pixie-4 has further registers for pullup control and I2C I/O to EEPROM

Write_SysFPGA, Write_FipFPGA, Write_Data:
Data written to these registers is used to configure FPGAs through the host computer. Write_SysFPGA (F17,A10) and Write_FipFPGA (F17, A9) are used in DGF Rev. E and F to send configuration data to the System FPGA and trigger/filter FPGAs (Fippi), respectively. Write_Data (0x00000004) is used for both purposes on the Pixie-4.

4.3 System FPGA Registers for Host

Version Register:
The System FPGA's version register can be read by the host to obtain version information. It is a 16 bit register; the individual bits are defined in the table below. Since the System FPGA firmware file is downloaded by the host, any hardware information taken from the register is only as good as the host recognizing the hardware. The version register is also read by the DSP and stored in the DSP output variables HARDWAREID and HARDVARIANT.

	Pixie-4 Rev. C/D	DGF Rev. E	DGF Rev. F	Notes
Address/ command	**0100xx**	**F1, A5**	**F1, A5**	
Bit 0-3	0xC, (0xD if specific)	0x4	0x5	Revision
Bit 4-7	0x1	0x1	0x0	Variant
Bit 8-11	0x7	0x5	0x5	Type
Bit 12-15	Incremented by 1 for each new release	0xA	Incremented by 1 for each new release	Build# or Product ID

CSR Register:

The System FPGA's CSR register is used to control data acquisition and boot the DSP.

	Pixie-4	DGF Rev. E	DGF Rev. F	Notes
Address/ command	**0x0110xx (write) 0x0110xx (read)**	**F17, A0 (write) F1, A0 (read)**	**F17, A0 (write) F1, A0 (read)**	
Bit 0	RunEna	RunEna	RunEna	Host r/w
1	(future) NewRun	NewRun	(future) NewRun	Host r/w
2	Host active (PCI)		Host active (USB)	Host r/w
3	EnaLAM	EnaLAM	EnaLAM	Host r/w
4	DSPReset	DSPReset	DSPReset	Host r/w
5	SyncCtrl		SyncCtrl	Host read only
6	future: HostReadFlag		future: HostReadFlag	Host read only
7	DSPWRtrace	SynchFlag	DSPWRtrace	Host read only
8	SynchFlag		SynchFlag	Host read only
9	Live*		Live*	Host read only
10	future PingPong0/1		future PingPong0/1	Host read only
11	future OddWordFlag		future OddWordFlag	Host read only
12	DSPErr (reserved)	DSPErr	DSPErr (reserved)	Host read only
13	Active	Active	Active	Host read only
14	LAMState	LAMState	LAMState	Host read only
15		Live*		Host read only

Notes:

- Bit 2 is set by the host to claim access to the external memory, holding off DSP MCA increments and DSP storing of list mode data (see section 8.4)
- Make sure bit 4 is cleared when writing to CSR for starting run, else the DSP will reboot.
- Bit 5 is tied to the "Busy" output on the front panel (used to be bit 13 for Rev. E). This allows starting/stopping modules synchronously without affecting the "Active" bit. It is useful in 2 occasions: 1. When the module detects a runstart condition, bit 13 is set immediately, bit 5 and the front panel are set when the module is ready to take data. This avoids the problem of polling bit 13 too soon and the host assuming the run has finished before it actually started. 2. In 32x buffer mode, bit 5 and the front panel toggle whenever a buffer is filled, but but bit 13 only is cleared after the 32ns buffer is filled and the module has to be read out.
- Bit 7 is set by the DSP when it transfers data to the external memory (see section 8.4).

5 CAMAC Commands

The lowest level communication with the DGF modules is through CAMAC commands (F/A codes with data read or writes). Table 8 lists the commands to be used by programmers for both revisions, those listed are identical. Their use is further described in the following sections.

Command, F,A code	**Action**	**Notes**
Write_CSR, F(17)A(0)	Write to CSR	
Read_CSR, F(1)A(0)	Read CSR	
Write_ICSR, F(17)A(8)	Write to ICSR	
Read_ICSR, F(1) A(8)	Read ICSR	
Write_TSAR, F(17)A(1)	Write to TSAR	
Read_TSAR, F(1)A(1)	Read TSAR	Disabled. Normally no need for host reads.
Write_WrdCnt, F(1)A(2)	Write to word count register	Disabled Normally no need for host writes
Read_WrdCnt, F(17)A(2)	Read word count register	
Write_Data, F(16)A(0)	Write data to DSP memory	
Read_Data, F(0)A(0)	Read data from DSP memory	
Read_Data_fast, F(5)A(0)	Level-1 fast CAMAC DSP data read	May not be fully tested in initial release of DGF F firmware/software.
Read_Version_Sys, F(1)A(5)	Read version of System FPGA	
Read_Version_Conf, F(1)A(13)	Read version of Config FPGA	
Write_FipFPGA, F(17)A(9)	Write configuration data for Fippi	
Write_SysFPGA, F(17)A(10)	Write configuration data for System	

Table 8: List of CAMAC commands

6 USB interface

The USB interface is used only to read out the external memory of the DGF Rev. F. Booting of the module, setting of parameters, starting/polling/stopping runs use the CAMAC interface with the same commands as previous revisions. Reading out MCA data or list mode data (in “32 buffer per spill” mode) from the external memory uses the USB interface. Single buffer list mode data (from DSP memory) is read out through CAMAC as before.

6.1 Drivers

For a (Windows) PC to communicate with the USB interface of a DGF Rev. F, it needs two driver files:

1. xia_usb2.inf contains the setup information to pick the correct driver file. It links devices with XIA’s Vendor ID to the driver file provided by Cypress
2. CyUsb.sys is the system driver file provided by Cypress.

When Windows recognizes a DGF Rev. F plugged into a USB port, it should be pointed to these drivers (located in the “drivers” folder of the XIA software distribution). When drivers are installed correctly, the DGF will appear in Window’s device manager as “XIA DGF-4C Spectrometer (Rev. F)”

6.2 DLL functions

Cypress provides a Windows development kit for USB interface development. XIA used this kit to generate a dll library which provides USB I/O functions to the main XIA C library. This dll (“USBdll.dll”) has to be copied to the Windows/System32 folder. The DLL defines 4 USB I/O functions, but only 2 (open and read) are used in the C library:

1. xia_usb2_open (int dev, HANDLE *h)
   Opens the device with the specified number (dev) and returns a valid HANDLE to the device or NULL if the device could not be opened.

2. xia_usb2_close (HANDLE h)
   Closes a device handle (h) previously opened via. xia_usb2_open().

3. xia_usb2_read (HANDLE h, unsigned long addr, unsigned long n_bytes, byte_t *buf);
   Reads the specified number of bytes from the specified address into the buffer *buf.

4. xia_usb2_write (HANDLE h, unsigned long addr, unsigned long n_bytes, byte_t *buf);
   Writes the specified number of bytes from the buffer *buf to the specified address.

The USB “address space” is defined as follows

0x0	– beginning of MCA memory (4 blocks of 32K words)
0x20000	- beginning of List mode memory (128K words)
0x10000000	- address of EEPROM storing serial number

While the addresses specify locations of 32bit words in the DGF’s external memory, the USB DLL functions require the number of *bytes* as the argument of how much data to read. Any address from 0x0 to 0x3FFFF can be addressed (though rarely necessary), but the EEPROM address is only a “code word” to read the specified number of bytes from the EEPROM memory, starting EEPROM memory address 0. The EEPROM can hold 16K bytes, currently only the first 2 are used to store the module’s serial number.

Thus typical operations are the following:

a) read from address 0x0 128K words (= 512K bytes) to read all spectra,
b) read from address 0x2000 two times the number of words specified in DSP variables EMwords (= number of 16 bit list mode data words acquired), i.e. (2 x EMwords) bytes
c) in double buffer mode, read from address 0x20000 [or 0x30000] two times the number of words specified in DSP variables EMwords [or EMwords2]
d) read from address 0x10000000 2 bytes to obtain the serial number.

The USB chip has an internal FIFO for 512 bytes. It is therefore most efficient to read data in multiples of 512 bytes; reading fewer bytes takes about the same time (or more?) that 512.

7 Booting

7.1 CAMAC boot sequence

Having the same basic architecture, the booting process for all models follows the same steps:

1. A “Config” or “Interface” FPGA is configured at power up from PROM
2. The host computer configures the System FPGA and trigger/filter FPGAs through the “Config FPGA”
3. The host computer boots the DSP through the System FPGA
4. The host computer sets DSP variables through the System FPGA
5. The host computer starts a control task run to apply parameters to the FPGAs (“ProgramFippi”)

The C libraries provided as part of the software take care of the slight differences in DGF Rev. E and DGF Rev. F (different C libraries are provided for the Pixie-4). User control software can simply call the appropriate boot function as before. For those users executing CAMAC commands directly, the new sequence of commands for all DGF models is listed below (**Differences to the sequence for Rev. D/E modules in bold**). If only one revision is present in the system, steps to “read hardware version” can be omitted.
This sequence assumes the version register can now be read before the System FPGA is configured; still to be tested. (Previously a read before all System FPGAs in the crate were configured might have locked up the CAMAC bus). Reading the version register allows the software to automatically select the appropriate file to download. However, if the version register can not be read before downloading the System FPGA, *the user has to specify* version D/E or F for each board, and the hardware version has to be read only in step 3.

Action	**Old CAMAC command sequence for Rev D, E**	**New CAMAC command sequence for Rev D, E, F**	**Data**	**Notes**
1. Read hardware version		**Read_Version, F(1)A(13)**		**Result’s lower 4 bits contain revision number (D=3,E=4,F=5)**
2. Configure System FPGA	Write_ICSR, F(17)A(8)	Write_ICSR, F(17)A(8)	0x1	Writing this bit resets FPGA
	Wait at least 50ms	Wait at least 50ms	--	
	Write_SysFPGA, F(17)A(10)	Write_SysFPGA, F(17)A(10)	Configuration data (166980 bytes)	**Use configuration file according to revision found above (D/E, or F) Do not read from Rev. D/E until all modules’ System FPGAs are configured** **check if still true!**

3. Read hardware version	**Read_Version, F(1)A(13)**			
4. Configure Trigger/ Filter FPGA	Write_ICSR, F(17)A(8)	Write_ICSR, F(17)A(8)	0xF0	Writing these bits resets FPGAs
	Wait at least 50ms	Wait at least 50ms	--	
	Write_FipFPGA, F(17)A(9)	Write_FipFPGA, F(17)A(9)	Configuration data (166980 bytes)	**Use configuration file according to revision found above (D, E, F)**
5. Confirm FPGA Downloads	Read_ICSR, F(1)A(8)	Read_ICSR, F(1)A(8)		If result = 0, all downloads were successful
6. Set Switchbus	**Write_ICSR, F(17)A(8)**	**Write_ICSR, F(17)A(8) for D/E only. No need to write in Rev. F, but writing does not harm**	**When taking switchbus bits from ModCSRA, make sure to mask all bits except bits 10 and 13.**	**For DGF Rev. F, the switchbus (trigger termination) is set by DSP during controltask 5 ("ProgramFippi")**
7. Boot DSP	Write_CSR, F(17)A(0)	Write_CSR, F(17)A(0)	0x10	
	Wait 50ms	Wait 50ms	--	
	Write_TSAR, F(17)A(1)	Write_TSAR, F(17)A(1)	1	Set DSP memory address to 1
	Write_Memory, F(16)A(0)	Write_Memory, F(16)A(0)	DSPcode[2], DSPcode[3],…, DSPcode[N]	
	Write_TSAR, F(17)A(1)	Write_TSAR, F(17)A(1)	0	Set DSP memory address to 0
	Write_Memory, F(16)A(0)	Write_Memory, F(16)A(0)	DSPcode[0],DSPcode[1]	

Notes

- Even though there are only 2 Trigger/Filter FPGAs on the DGF Rev. F (each containing logic for 2 channels), there are still 4 ICSR bits controlling configuration of the FPGA for compatibility reasons. The lower 2 bits control FPGA I (channels 0,1), the upper 2 bits control FPGA II (channels 2,3).
- As described above, all trigger/filter FPGAs are configured at the same time with the same data. In principle they can be configured individually, they one has to remember to do 4 individual downloads for the DGF Rev. E, but only 2 for the DGF Rev. F. In the Rev. F, the filter logic in channel 0 and 1 (or 2 and 3) is always identical and all channels 0-3 are set to the same decimation.

7.2 USB setup

The USB connection can be plugged in at any time, but for the PC to communicate with a particular DGF Rev. F module, each USB connection has to be assigned to a particular module. In the C library provided by XIA, this is accomplished by going through the following steps:

1. Before booting (e.g. in Igor interface), user enters module number (1-Nmod), CAMAC slot number (1-23) and serial number (e.g. 1404) for each module.
2. After booting all (Nmod) modules, try to "open" Nmod modules through USB connection and obtain a handle to each USB device.
3. Read the module's serial numbers (stored on USB EEPROM) from each device found
4. Match serial numbers with values entered by the user for each module number and thus link each USB device to a module number

8 Downloading Parameters

The procedure to download parameters to the DSP is the same as before, i.e. DGF Rev. F follows the same steps as DGF Rev. E.

9 Data Acquisition

To acquire data in list mode runs, the host has to start a run, poll for run stop (or data ready) when the module's output buffer is full, then read out the data. A subsequent run can be *resumed* to add to the same run statistics and MCA (it's also faster to resume that to restart). For MCA runs, the host starts a run and stops it after the desired time, then reads out the MCA data. These procedures are described in detail below.

9.1 Options for repeated list mode runs (not available in DGF Rev. E)

Traditionally, list mode runs took one 8K buffer of data. In the Pixie-4 and the DGF Rev. F, list mode runs can be set up to acquire several 8K buffers and transfer them to external memory, from where they can be read out more efficiently. This function is controlled by the variables MODCSRA and DBLBUFCSR:

If MODCSRA bit 0 = 1 and DBLBUFCSR bit 0 = 0, the DSP transfers a filled 8K buffer and resumes data acquisition <u>32 times</u>. Buffer fills and transfers are synchronized between modules. Acquisition stops after the 32nd buffer, the CSR bit 13 is cleared and (optionally) bit 14 is set. The host reads the external memory (in the DGF F through the USB interface), then resumes data acquisition. Note that in systems with several modules, there are now *groups of 32 buffers per module* following each other in the output data file, not individual buffers.

If MODCSRA bit 0 = 1 and DBLBUFCSR bit 0 = 1, the DSP transfers a filled 8K buffer and resumes data acquisition <u>16 times</u>. Buffer fills and transfers are synchronized between modules. After the 16th buffer, it sets bit 14 in the CSR to indicate data is ready, then

switches to a different block in the external memory and resumes acquisition. The host can then read out the data from the external memory, notify the DSP the memory block is now available again by setting bit 1 in DBLBUFCSR, and wait for the next 16 buffers to become available. Runs can continue indefinitely unless the host is too slow to read out the memory – if the DSP ever fills both blocks, it stops the acquisition (and clears CSR bit 13). Note that in systems with several modules, there are now *groups of 16 buffers per module* following each other in the output data file, not individual buffers.
In low count rate applications, buffers might fill slowly, so it might take a long time to get 16 buffers for the next update of list mode data. For more frequent updates, set MAXEVENTS to a smaller number so that the 8K buffers are only partially filled before transfer to the external memory.

9.2 Start/Stop/Resume

The procedure to start or resume a run and to stop is described below. Differences are highlighted. As always, CSR bits should be changed by reading the CSR from the module, changing a specific bit only, then writing it back.

Step	**Pixie-4**	**DGF Rev. E**	**DGF Rev. F**	**Notes**
1. Set DSP parameters appropriately (filter settings, runtask, etc)	XXX (PLX I/O function) clear MCA by EM write from host	As before, Write_TSAR (F17,A1, Address of DSPpars.) 256x Write_Data (F16,A0, DSPpar. values)	As DGF E, Write_TSAR (F17,A1, Address of DSPpars.) 256x Write_Data (F16,A0, DSPpar. Value)	In Pixie-4 and DGF F, set RESUME = 1 for a new run (clear MCA and statistics); RESUME =0 for a resumed run (no clear) Resumed runs must follow a new run with no change in DSP parameters (except INSYNC and RESUME)
2. Set CSR bit 0 to start run	XXX (PLX I/O function)	As before, Write_CSR (F17,A0, CSR value)	As DGF E, Write_CSR (F17,A0, CSR value)	In DGF E, set CSR bit 1for a new run (clear MCA and statistics), clear bit 1 for a resumed run (no clear)
3. Poll CSR	XXX (PLX I/O function)	As before, Read_CSR (F1,A0, CSR value)	As DGF E, Read_CSR (F1,A0, CSR value)	
4. Determine run status, then go to step 3 or 5	Bit 13 set while run in progress. Bit 14 (always) set when data ready	Bit 13 set while run in progress. Bit 14 set when data ready if bit 3 (LAMenable) was set	Bit 13 set while run in progress. Bit 14 set when data ready if bit 3 (LAMenable) was set	List mode runs stop by themselves when buffer is full
5. If desired (or timeout), stop run by clearing CSR bit 0.	XXX (PLX I/O function)	As before, Write_CSR (F17,A0, CSR value)	Possibly Write_TSAR (F17,A1, Address of RESUME)	For Pixie-4 and DGF F, in repeated list mode runs with more than one module, set DSP

			Write_Data (F16,A0, value=2), then Write_CSR (F17,A0, CSR value)	variable RESUME = 2 before writing CSR.

9.3 List mode data readout from DSP memory (1 buffer/spill)

The procedure to read out an 8K buffer from DSP memory when the run is stopped is essentially the same in all models:

Step	Pixie-4	DGF Rev. E	DGF Rev. F	Notes
1. Read Word Count Register to get Nwords	XXX (PLX I/O function)	As before, Read_WrdCnt (F1,A2, Nwords)	As DGF E, Read_WrdCnt (F1,A2, Nwords)	Reading WCR also clears LAM bit
2 Set address of output buffer	XXX (PLX I/O function)	As before, Write_TSAR (F17,A1, Address of Outbuffer)	As before, Write_TSAR (F17,A1, Address of Outbuffer)	
3. Read Nwords	XXX (PLX I/O function)	As before, Nwords x Read_Data (F0,A0, LMdata)	As DGF E, Nwords x Read_Data (F0,A0, LMdata)	

9.4 List mode data readout from external memory (32 buffers/spill)

The procedure to read out 32 8K buffers from external memory (EM) only applies to Pixie-4 and DGF Rev. F. The run is assumed to be stopped at this point. The DGF Rev. F reads the external memory through the USB interface only. As always, CSR bits should be changed by reading the CSR from the module, changing a specific bit only, then writing it back.

Step	Pixie-4	DGF Rev. F	Notes
1. Read DSP parameters to get Nwords	XXX (PLX I/O function)	Write_TSAR (F17,A1, Address of DSPpars.) 256x Read_Data (F0,A0, DSPpar. Value)	Number of 16 bit words is Nwords = (Emwords)*65536 + (Emwords+1)
2. Dummy read to WordCountRegister	XXX (PLX I/O function)	Read_WrdCnt (F1,A2, dummy)	
3. Compute number of 32bit words	Nw32 = floor(Nwords/2)	Nw8 = Nwords *2 Increment to $Nw8_{512}$ =	

Nw32 or number of bytes Nw8	Add 1 if Nwords is odd	next largest multiple of 512 bytes	
4. Set CSR bit 2 to indicate host access to EM	XXX (PLX I/O function)	Write_CSR (F17,A0, CSR value)	
5. Wait for CSR bit 7 to be zero (DSP no longer uses EM)	XXX (PLX I/O function)	Read_CSR (F1,A0, CSR value)	
6. Read Nw32 words from EM	XXX (PLX I/O function)	USB readout, $Nw8_{512}$ bytes	EM address is 128K
7. Clear CSR bit 2 to release EM to DSP	XXX (PLX I/O function)	Write_CSR (F17,A0, CSR value)	
8. Save to file	omit upper half of last 32bit word if Nwords is odd.	only save Nw8 bytes, not $Nw8_{512}$	

9.5 List mode data readout in double buffer mode (16 buffers/spill)

The procedure to read out an 16 8K buffers from external memory (EM) only applies to Pixie-4 and DGF Rev. F. In this case, the run is still in progress. The DGF Rev. F reads the external memory through the USB interface only. As always, CSR bits should be changed by reading the CSR from the module, changing a specific bit only, then writing it back. Note that in the future, the DBLBUFCSR control bits might be moved into the CSR and the word count register might be used to specify the number of *32bit* words. PRELIMINARY

Step	Pixie-4	DGF Rev. F	Notes
1. Read DSP parameters to get Nwords1,2 DBLBUFCSR	XXX (PLX I/O function)	Write_TSAR (F17,A1, Address of DSPpars.) 256x Read_Data (F0,A0, DSPpar. Value)	Number of 16bit words in block 1 is Nwords1 = (Emwords)*65536 + (Emwords+1) In block 2 it is Nwords2 = (Emwords2)*65536 + (Emwords2+1)
2. Determine block:	if Nwords1>0: block 1 if Nwords2>0: block 2 if both: read both (block 1 first if bit 3 in DBLBUFCSR is set)		
3. Dummy read to WordCountRegister	XXX (PLX I/O function)	Read_WrdCnt (F1,A2, dummy)	Clears WCR bit 14
4. Compute number of 32bit words Nw32 or number of bytes Nw8	Nw32 = floor (Nwords1,2 / 2) Add 1 if Nwords1,2 is odd	Nw8 = Nwords *2 Increment to $Nw8_{512}$ = next largest multiple of 512 bytes	
5. Set CSR bit 2 to indicate host access to EM	XXX (PLX I/O function)	Write_CSR (F17,A0, CSR value)	

6. Wait for CSR bit 7 to be zero (DSP no longer uses EM)	XXX (PLX I/O function)	Read_CSR (F1,A0, CSR value)	
7. Read Nw32 words from EM	XXX (PLX I/O function)	USB readout, $Nw8_{512}$ bytes	EM address block 1 = 128K EM address block 2 = 192K
8. Clear CSR bit 2 to release EM to DSP	XXX (PLX I/O function)	Write_CSR (F17,A0, CSR value)	
9. Set DBLBUFCSR bit 1 to indicate host has read.	XXX (PLX I/O function)	Write_TSAR (F17,A1, Address of DBLBUDCSR.) Write_Data (F16,A0, DBLBUFCSR value)	
10. Save to file.	omit upper half of last 32bit word if Nwords is odd	only save Nw8 bytes, not $Nw8_{512}$	

9.6 MCA data readout

The procedure to read out the MCA data in the Pixie-4 and the DGF Rev. F is very similar to reading list mode data from external memory (see above). MCA datacan be read at any time. In the DGF Rev. E, the MCA data is paged through the DSP and can only be read when the run is stopped. (as described in the DGF Rev. E user/programmer manuals).

Step	Pixie-4	DGF Rev. F	Notes
1. Set CSR bit 2 to indicate host access to EM	XXX (PLX I/O function)	Write_CSR (F17,A0, CSR value)	
2. Wait for CSR bit 7 to be zero (DSP no longer uses EM)	XXX (PLX I/O function)	Read_CSR (F1,A0, CSR value)	
3. Read 128K words from EM	XXX (PLX I/O function)	USB readout, 512K bytes	EM address is 0
4. Clear CSR bit 2 to release EM to DSP	XXX (PLX I/O function)	Write_CSR (F17,A0, CSR value)	
5. Save to file			

10 Output Data Formats

The list mode output data follows the traditional format, with 2 exceptions: the "format" buffer header word distinguishes the models (and thus the sampling interval in the traces) and the hit pattern contains additional information.

Word	**Pixie-4 (code 3.5+)**	**DGF Rev. E**	**DGF Rev. F**	**Notes**
BufHeader0	Nwords in buffer	Nwords in buffer	Nwords in buffer	
BufHeader1	Module Number	Module Number	Module Number	
BufHeader2	Runtask +0x2000	Runtask +0x1000	Runtask +0x3000	Old Pixie-4 code: Runtask +0x1000
BufHeader3	Time	Time	Time	
BufHeader4	Time	Time	Time	
BufHeader5	Time	Time	Time	
EventHeader0 Bits 0-3: Bits 4-7: Bits 8-11: Bits 12-15:	 data in channel 0-3 mod. Coinc status channel 0-3 was hit GATE flag ch. 0-3	 data in channel 0-3 all zero all zero all zero	 data in channel 0-3 all zero channel 0-3 was hit GATE flag ch. 0-3	 If hit, energy is valid, else it may be zero or an "estimate"
EventHeader1	Time	Time	Time	
EventHeader2	Time	Time	Time	
ChannelHeader0	Nwords in channel	Nwords in channel	Nwords in channel	
ChannelHeader1	Time	Time	Time	
ChannelHeader2	Energy	Energy	Energy	
ChannelHeader3	XIA PSA	XIA PSA	XIA PSA	
ChannelHeader4	User PSA	User PSA	User PSA	
ChannelHeader5	reserved	GSLT	reserved	
ChannelHeader6	reserved	GSLT	reserved	
ChannelHeader7	reserved	GSLT	reserved	
ChannelHeader8	time	time	time	
Trace Sampling interval	13.3ns	25ns	12.5ns	