DIF - Operating Manual

Version 1.16 (19.1.10)

the DIF developers

0 Introduction

A simplified block diagram of the communication between DIF and LDA, or as it is proposed to a PC via USB bus, is shown in Fig. 1. Only the interface between LDA and DIF is shown, the remaining functionalities are combined on DIF- and LDA-side in the blocks “Control Unit”. The USB-DIF link is generally for debugging.

0.1 LDA-DIF interface

The only connection between LDA and DIF is realized with a 19-pin HDMI cable. Commands are in general 8b/10b-channel coded, while the coding and decoding blocks are fully transparent for the remaining logic/electronics. The DIF is operated with the clock from the LDA: no PLL (DCM) on the DIF, by which a fully synchronous operation of all the DIFs that are connected to one LDA is guaranteed. Clock speed (default): 100MHz (but also possible: 40-120MHz). Fast Commands from the LDA to the DIF like a trigger are transported without channel coding, as well as fast commands from DIF to the LDA like the signal “RAMFull”.

0.2 PC (USB) to DIF interface

The USB interface should “emulate” the LDA-DIF interface as much as possible in order to allow an easy switching to the LDA-DIF setup. The USB-interface does not use the 8b/10b channel coders/decoders. The DIF clock may be generated from the USB side or by a local oscillator. In the USB-setup, PLLs (DCMs) within the DIF FPGA are allowed.

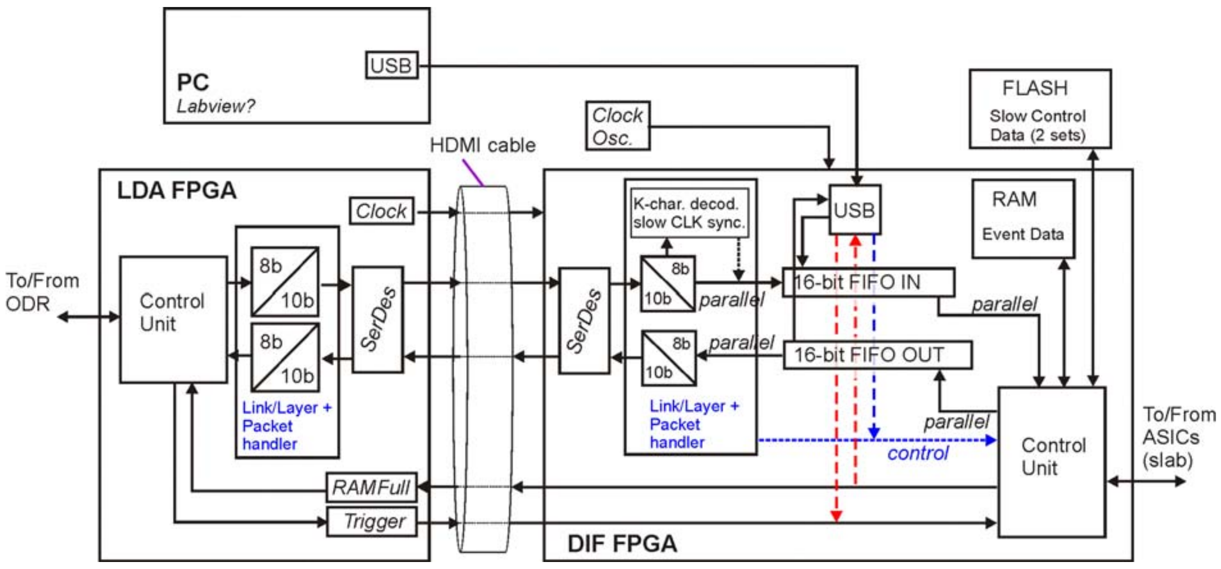

Figure 1: Interface of the DIF FPGA to the LDA, or a PC via USB

1 Transfer between LDA and DIF

The data transfer between LDA and DIF is 8b/10b channel coded. The 8b/10b coding is realized by a 5b/6b and a 3b/4b coder.
Two types of data transfer "frames" are defined between LDA and DIF [1, 2]:

1.1 (Fast-) Command Frames

The Fast-Commands are used for link-synchronization and for timing critical DIF commands (especially broadcasts to all DIFs) only.
A command frame is 16-bit long:

15 8	7 0
komma character (K)	command word (D)

The komma character K and the command word D are referenced to by K*X.Y* and D*X.Y*, respectively: X has a **5-bit resolution**, Y has a **3-bit resolution**. E.g. K28.1 is the 8-bit sequence 11100 001.

1.2 Block Transfers

A block transfer is used to transmit configuration-, result- or status information data between LDA and DIF as well as timing-uncritical commands. The length is not fixed, although only an even number of 16-bit words is allowed (see K23.7 in Table 1 and /EPD/ in Table 2) and maximum block length is 1kByte. The block:

packettype	pktID	type_modifier (command def.)	specifier	data_length	data	CRC
16 bit	16 bit	16 bit	16 bit	16 bit	data_length* 16 bit	16 bit

packettype: define block to be:

packettype (16-bit hex)	identifies packet to be	remark
0x0001	block data	
0x0002	generic command	
0x0010	command ECAL only	
0x0020	command DHCAL only	
0x0040	command AHCAL only	
0x0080	Command is for DIF-DIF Link	
0x0100	data is firmware for FPGA	
0x1000 to 0xF000	DCC identifier	only the upper 4 bits!!

Each packet identifier (second column) is assigned to a certain bit in the 16-bit packettype (first column). The bits can be combined: e.g. packettype=0x0012 means "generic command for ECAL only".

pktID: numeration of sent blocks, used to identify block losses.

type_modifier: command definition. In practise, this is the address of the respective command register inside the DIF. See table 8.
specifier (new!): specification (e.g. target address) of the command in "type_modifier". This was the first data vector before.

data_length: Number of 16-bit vectors sent in the "data"-section of the block.

data: 16-bit data vectors, e.g. slow-control data for the ASICs, temperatures, voltages, currents.

CRC: cyclic redundancy check (look for transmission errors).

1.3 Komma Characters (see section 1.1) and special sequences [1]

The 8b/10b channel coding allows for channel synchronization and maintenance the so called komma characters (K):

Komma Character K	**Task (Meaning)**
K28.0	Signals the next symbol (command word) is a SYNCCMD.
K28.1	
K28.2	
K28.3	Signals the next symbol (command word) is a COMMAND.
K28.4	Signals in the next symbol (command word) is for DIF-DIF link
K28.5	reserved for link synchronization
K28.6	
K28.7	reserved for link synchronization
K23.7	Carrier Extend. Used to PAD the end of a data frame out to an even number of Symbols, so that next frame, or IDLE sequence starts on an even footing. **/R/**
K27.7	Start of data frame **/S/**
K29.7	End of data frame **/T/**
K30.7	

Table 1: Komma Characters

Several special sequences are defined:

Set	**Sequence**	**Comment**
/I1/	/K28.5/D5.6/	Idle sequence, sent when running DP is +, flips it to -. Sent automatically.
/I2/	/K28.5/D16.2/	Idle sequence, sent when running DP is -, maintains it as -. Sent automatically.
/EPD/	**/T/R/** or **/T/R/R/**	Used to end a data frame, the addition of an extra /R/ is used to pad things out to an even number.
/LOOP/	/K28.5/D12.6/	Low-level link loop back start (DON'T SEND from User-Logic)
/ENDLOOP/	/K28.5/D16.7/	Low-level link loop back end (DON'T SEND from User-Logic)
/LINKSTART/	/K28.5/D1.4/	Link Start. (DON'T SEND from User-Logic)
/LINKACK/	/K28.5/D30.3/	Link Start ACK. (DON'T SEND from User-Logic)

Table 2: Special Sequences

2 DIF Commands (from LDA or USB to DIF FPGA)

The commands are subdivided into:
- timing critical commands (see table 3), sent with "FAST command frames" (see section 1.1)
- timing uncritical signals (see table 8) that are sent with "BLOCK transfers (see section 1.2).

For each command that is sent from LDA to DIF, the DIF has a dedicated **command register**. The address of this **command register** is defined:
- for FAST_Commands by the X in the incoming DX.Y command word (see section 1.1)
- for Block-Transfers by the type_modifier and the specifier (see section 1.2).

Command registers are 16-bit, and can be subdivided for several functional purposes.

The general notation is:

15 … 10	9 … 5	4	3	2	1	0
Reserved	Status Bits(4:0)	Bit	Bit	Bit	Bit	Bit
R, +0	RC, +10100	RW, +0	RS, +0	RW, +0	RW, +0	RW, +0

Note: R = Readable by the LDA,
W = Writeable by the LDA,
C = Clearable by the LDA,
S = Settable by the LDA,

+x = Value undefined after reset,
+0 = Value is 0 after reset,
+1 = Value is 1 after reset,

2.1 FAST Commands

FAST Command *see section 1.1*	**komma character**	**command word D**	**Operation**	**Change DIF State?**
reset_BCID	K28.3	D1.1	reset BCID	no
start_acquire	K28.3	D2.1 D2.2 D2.3	start data-taking (int. trig) start data-taking (ext. trigger) stop data-taking	yes
stop_readout	K28.3	D3.1 D3.2	stop data transfer DIF=>LDA continue data transfer	no
#### ECAL specific ####				
	K28.3	D5.0		
#### DHCAL specific ####				
	K28.3	D8.0		
#### AHCAL specific ####				
calibrate	K28.3	 D11.1 D11.2 D11.3 D11.4	do a calibration run: with light sys., int. trig with charge sys., int. trig with light sys., ext. trig. with charge sys., ext. trig.	yes
#### DCC identifier ####				
	K28.3	D15.0		

Table 3: FAST Commands (timing critical and broadcasts) from LDA to DIF

ECAL specific (D5.0 – D7.7), DHCAL specific (D8.0 – D10.7), AHCAL specific (D11.0 – D13.7), DCC identifier (D15.0-D17.7)

2.1.1 reset_BCID, FAST-command, set by D1.1

15 … 1	0
reserved	BCID_counter
+0	S, +0

Bit no.	Bit Field	Description
15 – 1	reserved	reserved
0	BCID_counter	RESET the BCID (bunch counter) synchronously for all DIFs (broadcast command): reset_BCID = ‘1’: reset is active for 4 clock cycles, afterwards the DIF resets this bit automatically (set by D1.1) reset_BCID = ‘0’: reset is not active.

Table 4: reset_BCID register description

This register cannot be read from the LDA. A status bit of this command is in the general register (see section 2.2.2).

2.1.2 start_acquire, FAST-command, set by D2.Y

15 … 3	2	1	0
reserved	stop	start_ext	start_int
+0	S, +0	S, +0	S, +0

Bit no.	Bit Field	Description
15 – 3	reserved	reserved
2	stop	stop data-taking synchronously for all DIFs (broadcast command): stop = ‘1’: data taking is stopped (set by D2.3) stop is reset by the DIF automatically after executing the command. stop=’0’: no action
1	start_ext	start data-taking synchronously for all DIFs (broadcast command) with external trigger: start_ext = ‘1’: data taking is started (set by D2.2) start_ext is reset by the DIF automatically after executing the command. Puts DIF into “ACTIVE” mode. start_ext=’0’: no action
0	start_int	start data-taking synchronously for all DIFs (broadcast command) with internal trigger: start_int = ‘1’: data taking is started (set by D2.1) start_int is reset by the DIF automatically after executing the command. Puts DIF into “ACTIVE” mode. start_int=’0’: no action

Table 5: start_acquire register description

This register cannot be read from the LDA. Status bits of this command is in the general register (see section 2.2.X).

2.1.3 stop_readout, FAST-command, set by D3.Y

15 … 2	1	0
reserved	CONTINUE	STOP
+0	S, +0	S, +0

Bit no.	Bit Field	Description
15 – 2	reserved	reserved

1	CONTINUE	continue readout synchronously for all DIFs (broadcast command):
		CONTINUE = ‘1’: readout is continued (set by D3.2)
		CONTINUE is reset by the DIF automatically after executing the command.
		CONTINUE=’0’: no action

0	STOP	stop readout synchronously for all DIFs (broadcast command):
		STOP = ‘1’: readout is stopped (set by D3.1)
		STOP is reset by the DIF automatically after executing the command.
		STOP=’0’: no action

Table 6: stop_readout register description

2.1.4 calibrate, FAST command, set by D11.Y, AHCAL specific

15 … 4	3	2	1	0
Reserved	CALIB3	CALIB2	CALIB1	CALIB0
+0	S, +0	S, +0	S, +0	S, +0

Bit no.	Bit Field	Description
15 – 4	reserved	reserved
3	CALIB3	start a calibration run synchronously for all DIFs (broadcast command) with charge injection, external trigger: CALIB3 = ‘1’: data taking is started (set by D11.4) CALIB3 is reset by the DIF automatically after executing the command. Puts DIF into “ACTIVE” mode. CALIB3=’0’: no action
2	CALIB2	start a calibration run synchronously for all DIFs (broadcast command) with LEDs, external trigger: CALIB2 = ‘1’: data taking is started (set by D11.3) CALIB2 is reset by the DIF automatically after executing the command. Puts DIF into “ACTIVE” mode. CALIB2=’0’: no action
1	CALIB1	start a calibration run synchronously for all DIFs (broadcast command) with charge injection, internal trigger: CALIB1 = ‘1’: data taking is started (set by D11.2)

Bit no.	Bit Field	Description
		CALIB1 is reset by the DIF automatically after executing the command. Puts DIF into “ACTIVE” mode. CALIB1=’0’: no action
0	CALIB0	start a calibration run synchronously for all DIFs (broadcast command) with LEDs, external trigger: CALIB0 = ‘1’: data taking is started (set by D11.1) CALIB0 is reset by the DIF automatically after executing the command. Puts DIF into “ACTIVE” mode. CALIB0=’0’: no action

Table 7: calibrate register description (AHCAL specific)

This register cannot be read from the LDA. Status bits of this command is in the general register (see section 2.2.X).

2.2 BLOCK TRANSFER address map

Block Transfer Name (command) *see section 1.2*	**type_modifier (command def.)** 16-bit hex	**specifier** 16-bit hex	**Operation**
power_on	0x0002	0x0000 0x0001 0x0002 0x1000	turn power regulators off turn power regulators on automatic: controlled by DIF read power register
reset	0x0004	0x0001 0x0002 0x0004 0x0008 0x0010 0x0020 0x0100 0x1000	reset of DIF reset of slab reset all reset slow-control registers reset read reset probe reset calib read reset register
set_DIF_mode	0x0006	0x0001 0x0002 0x1000	set detector into “SLEEP” set detector into “READY” read DIF_mode register
power_pulsing	0x0008	0x0001 0x0002 0x0004 0x0008 0x0010 0x0020 0x1000	turn pwr_analog ON turn pwr_digital ON turn pwr_ss/pwr_sca ON turn pwr_adc ON turn pwr_dac ON turn all ON read power_pulsing register
transfer_sc_data	0x000A	header + sc_data 0x1000	load data from LDA to DIF header: see section 2.4.2 read transfer_sc_data register
load_sc_data	0x000C	0x0001 0x0101 0x0002 0x0102 0x0004 0x0104 0x0008 0x0108	load set1_0 from DIF to slab1 load set1_1 from DIF to slab1 load set2_0 from DIF to slab2 load set2_1 from DIF to slab2 load set3_0 from DIF to slab3 load set3_1 from DIF to slab3 load set4_0 from DIF to slab4 load set4_1 from DIF to slab4

Block Transfer Name (command) *see section 1.2*	**type_modifier (command def.)** 16-bit hex	**specifier** 16-bit hex	**Operation**
		0x0080 0x1000	**removed:** *readback all sets* read load_sc_data reg.
read_results	0x000E	0x0001 0x1000	read data slab via DIF to LDA read read_results reg.
set_control_reg	0x0010	16-bit control-register	set control register
read_status_control	0x0012	0x0001 0x0002 0x0003	readout of DIF control register readout of DIF status1 register readout of DIF status2 register
readout_info	0x0014	0x0001 0x0002 0x0004 0x0008 0x0010 0x0020 0x0040	read DIF firmware date read DIF firmware version read board's production date read DIF board-ID read board's version number read DIF serial number readout all infos
readback_sc	0x0016	header + sc_data 0x1000	read sc_data from DIF header: see section 2.4.2 read readback_sc register
FPGA_firmware	0x0018	firmware-data 0x1000	load data from LDA to DIF (n data vectors) read transfer_sc_data reg. (1 data vector)
sel_command_input	0x001A	0x0000 0x0001 0x0002 0x1000	LDA-DIF link is used DIF-DIF link is used reset input selection logic read sel_ command_input reg.
pre_spill_indication	0x001C	0x0001 0x1000	indicates an upcoming "start_acquire" read pre_spill_ind. register
read_all	0x001E	0x0001	read all registers in rising address order inside DIF
gen_fcmd	0x0020	0x0001	generate a FAST Command from a Block transfer frame
#### ECAL specific ####			
	0x0030		
#### DHCAL specific ####			
	0x0050		
#### AHCAL specific : Data Vector is 4Bytes long. ####			
set_delay_line	0x0070	0x210000YY 0x220000YY 0x230000YY 0x24010000 0x25010000 0x26010000	Delay Line 1: YY=Delay Setting Delay Line 2: YY=Delay Setting Delay Line 3: YY=Delay Setting Delay Line 1: Read setting Delay Line 2: Read setting Delay Line 3: Read Setting
CAL_enable	0x0072	0x310000YY 0x320000YY 0x330000YY 0x340000YY	LVDS_1: YY=0 (off) / 1 (on) LVDS_2: Y=0/1 LVDS_3: Y=0/1 LVDS_4: Y=0/1

Block Transfer Name (command) *see section 1.2*	**type_modifier (command def.)** 16-bit hex	**specifier** 16-bit hex	**Operation**
		0x350000YY 0x360000YY 0x370000YY 0x380000YY 0x390000YY 0x3A00YY00 0x3B00YY00	LVDS_5: Y=0/1 LVDS_6: Y=0/1 PWR_LED: YY=0 (off) / 1 (on) PWR_Charge: YY=0(off) / 1(on) Slab Power: YY=0 (off) / 1 (on) SiPM Bias: YY=0 (off) / 1 (on) Pre_Bias: YY=0 (10V) / 1 (on)
operate_DAC	0x0074	0x4100XXXX 0x4200XXXX 0x43020000 0x44020000 0x45000000 0x46000000 0x47000000 0x48000000 0x5A000000	DAC1: XXXX: Setting for LED: 0..10.000mV DAC2 XXXX: Setting for Charge Injection: 0..3500mV Read DAC1 Read DAC2 Switch ON DAC1 (LED) Switch OFF DAC1 (LED) Switch ON DAC2 Switch OFF DAC2 Calibrate DACs (takes 15 sec)
setup_ADC	0x0076	0x55000000 0x560000XX 0x57010000	Calibrate all ADCs set no. ADCs readings (XX: Byte) read no. ADC readings
read_CALIB_info	0x0078	0x11060000 0x12040000 0x13010000 0x14010000	Serial number, 6bytes return software date, 4bytes return software version, 1byte return board version, 1byte return
read_ADC	0x007A	0x6A02000X	read ADC A=1..4, X=ADC channel 0..7 ADCs: Show temperatures, supply voltages and currents
set_calib	0x007C	0x7100XXXX 0x7200XXXX 0x730000XX 0x740000XX 0x750000XX 0x760000XX 0x770000XX 0x780000XX 0x8A0B0000 **0x91000000** 0x92000000	set no. of pulse trains: XXXX: 1..5000 (default: 1) set pulse frequency: XXXX: 1..1000Hz (default: 1) set pulse duty cycle XX: 0..100% (def.: 0x32=50%) set no. of pulses: XX: 1..50 (default: 1) set delay between measurements: XX: 0..255msec (default: 0x1) set ext_trig on/off: XX: 0..1 (default: 0=off) set TCALIB Source: XX: 0= µC, 1=DIF (default: 0) select: LED or Charge Injection: XX: 1= LED, 2=Charge (default: 1) readback settings: A: value to read B: number of bytes to read **START calibration** (settings: see above) STOP measurement
set_read_reg	0x0080	0x0xyz	xyz: number of clock cycles to

Block Transfer Name (command) *see section 1.2*	**type_modifier (command def.)** 16-bit hex	**specifier** 16-bit hex	**Operation**
			generate (12bit max).
set_probe_reg	0x0082	0xABCD	ABCD: number of clock cycles to generate (15bit max)
read_poll_reg	0x0084	0x0001	DESY TB: Poll testbeam control register (2 Bytes)
Activate	0x0086	0x0001	DESY TB: enable external start_acquire

Table 8: Block transfers between LDA and DIF

type_modifier:
ECAL specific (0x0030-0x004F), DHCAL specific (0x0050-0x006F), AHCAL specific (0x0070 – 0x008F)

Change DIF State (last column table 3 and table 8) means: If the DIF changes its state from „IDLE“ to any other state by a received command, the respective operation should not be interrupted by following commands except for emergencies or resets. After completion of the tasks, the DIF changes back to “IDLE” automatically and is ready for new commands.

2.2.1 power_on, BLOCK Transfer command, address 0x0002

15 – 2	1	0
reserved	automatic	slab_power
R, +0	RW, +0	RW, +0

Bit no.	**Bit Field**	**Description**
15 – 2	reserved	reserved
1	automatic	automatic = ‘1’: power signals (c.f. section 2.2.4) are controlled by DIF. slab_power has to be at ‘1’ for automatic operation. automatic = ‘0’: no action automatic is reset when slab_power is set to ‘0’.
0	slab_power	Switch on or off slab power: slab_power = ‘1’: slab power is on (set by specifier=0x0001) slab_power = ‘0’: slab power is off (set by specifier=0x0000)

Table 9: power_on register description

2.2.2 reset, BLOCK Transfer command, address 0x0004

15 – 5	4	3	2	1	0
reserved	reset_BCID	reset_SC	reset_all	reset_slab	reset_DIF
R, +0	R, +0	RS, +0	RS, +0	RS, +0	RS, +0

Bit no.	**Bit Field**	**Description**
15 – 4	reserved	reserved
4	reset_BCID	Reset bunch counters of the ASICs, set by FAST Command

Bit no.	Bit Field	Description
		(see section 2.1.1), read only. reset_BCID = ‘1’: reset is active for 4 clock cycles, reset_BCID = ‘0’: reset is not active.
3	reset_SC	reset of the slow control registers of the ASICs: reset_SC = ‘1’: reset active for 6 clock cycles, afterwards the DIF resets this bit automatically (set by specifier=0x0008) reset_SC = ‘0’: reset not active.
2	reset_all	general reset of DIF and slab electronics: reset_all = ‘1’: reset active for 6 clock cycles, afterwards the DIF resets this bit automatically (set by specifier=0x0004) reset_all = ‘0’: reset not active.
1	reset_slab	reset of slab electronics: reset_slab = ‘1’: reset active for 6 clock cycles, afterwards the DIF resets this bit automatically (set by specifier=0x0002) reset_slab = ‘0’: reset not active.
0	reset_DIF	reset of DIF electronics: reset_DIF = ‘1’: reset active for 6 clock cycles, afterwards the DIF resets this bit automatically (set by specifier=0x0001) reset_DIF = ‘0’: reset not active.

Table 10: Reset register description

2.2.3 DIF_mode BLOCK Transfer command, address 0x0006

15 7	6 4	3	2	1	0
Reserved	current_mode(2:0)	LOOP	SYNC	READY	SLEEP
R, +0	R, +000	R, +0	R, +0	RS, +0	RS, +0

Bit no.	Bit Field	Description
15 – 7	reserved	reserved

6 - 4	current_mode	Shows the actual mode, the DIF is in, defined by the last “DIF_mode” command from the LDA (read-only):
		current_mode = ‘000’ DIF is in SLEEP mode,
		current_mode = ‘001’ DIF is in IDLE mode,
		current_mode = ‘010’ DIF is in SYNC mode,
		current_mode = ‘011’ DIF is in LOOP mode

3	LOOP	puts DIF into LOOP mode (debugging) LOOP = ‘1’ : DIF is in LOOP mode with LDA, after accepting, the DIF resets this bit automatically and ‘current_mode is set to ‘011’ (set by FAST Command) LOOP = ‘0’ : no mode change
2	SYNC	puts DIF into SYNC mode (DIF synchronization) SYNC = ‘1’ : DIF is in SYNC mode, after accepting, the DIF resets this bit automatically and ‘current_mode is set to ‘010’ (set by FAST Command) SYNC = ‘0’ : no mode change
1	READY	puts DIF into IDLE mode (general wait and ready state): IDLE = ‘1’ : DIF is in IDLE mode, after accepting, the DIF

Bit no.	Bit Field	Description
		resets this bit automatically and ‘current_mode is set to ‘001’ (set by specifier = 0x0002) IDLE = ‘0’ : no mode change
0	SLEEP	puts DIF into SLEEP mode (powered-down wait state) SLEEP = ‘1’ : DIF is in SLEEP mode with LDA, after accepting, the DIF resets this bit automatically and ‘current_mode is set to ‘000’ (set by specifier = 0x0001) SLEEP = ‘0’ : no mode change

Table 11: DIF_mode register description

2.2.4 power_pulsing BLOCK Transfer command, address 0x0008

This command is for debugging only. The power pulsing control should be done automatically by the DIF in order to guarantee a timing-precise switching. E.g., on a “start_acquire”-command from the LDA, the DIF switches-on the slab before starting the data-taking.

15 6	5	4	3	2	1	0
Reserved	SLAB	DAC	ADC	SS_SCA	DIGITAL	ANALOG
R, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

Bit no.	Bit Field	Description
15 – 6	reserved	reserved

5	SLAB	Switch the power of the complete slab, namely the power-pulsing control signals: pwr_analog, pwr_digital, pwr_ss/pwr_sca, pwr_adc, pwr_dac.
		SLAB = ‘1’ SLAB is switched ON (set by specifier = 0x0020)
		SLAB = ‘0’ SLAB is switched OFF

4	DAC	DAC = ‘1’ pwr_dac is switched ON (set by specifier = 0x0010)
		DAC = ‘0’ pwr_dac is switched OFF

3	ADC	ADC = ‘1’ pwr_adc is switched ON (set by specifier = “0x0008)
		ADC = ‘0’ pwr_adc is switched OFF

2	SS_SCA	SS_SCA = ‘1’ pwr_ss/pwr_sca is switched ON (set by specifier = 0x0004)
		SS_SCA = ‘0’ pwr_ss/pwr_sca is switched OFF

1	DIGITAL	DIGITAL = ‘1’ pwr_digital is switched ON (set by specifier = 0x0002)
		DIGITAL = ‘0’ pwr_digital is switched OFF

0	ANALOG	ANALOG = ‘1’ pwr_analog is switched ON (set by specifier = 0x0001)
		ANALOG = ‘0’ pwr_analog is switched OFF

Table 12: power_pulsing register description

2.2.5 transfer_sc_data, BLOCK Transfer command, address 0x000A

15 … 10	9 … 2	1	0
reserved	Last_Access	CRC	LDA_DIF
R, +0	R, +0	R, +0	RS, +0

Bit no.	Bit Field	Description
15 – 10	reserved	reserved
9 – 2	Last_Access	Contents of the latest incoming 'header' (first 16-bit data vector, of which only 8-bit carry header information): for header see section 2.4.2
1	CRC	CRC = '1' CRC-check ok for last LDA-DIF data transfer CRC = '0' transmission errors in current data set
0	LDA_DIF	LDA_DIF = '1' slow-control data is transferred from LDA to DIF. Bit is reset by DIF after completion automatically. LDA_DIF = '0' no action

Table 13: transfer_sc_data register description

By this command, the slow_control configuration data sets that are stored in the Flash memory of the DIF are overwritten.

2.2.6 load_sc_data, BLOCK Transfer command, address 0x000C

15 … 9	8	7	6 … 4	3	2	1	0
STATUS[6:0]	SET	READBACK	reserved	SLAB4	SLAB3	SLAB2	SLAB1
R, +0	RW, +0	RS, +0	R, +0	RS, +0	RS, +0	RS, +0	RS, +0

Bit no.	Bit Field	Description

15 – 9	STATUS[6:0]	if the slabs (partitions) are programmed twice with sc_data, the readback data can be used to identify problems.
		STATUS[6:4]: reserved for later use
		STATUS[3] = '1': sc_programming slab4 does not work
		STATUS[3] = '0': sc_programming slab4 ok
		STATUS[2] = '1': sc_programming slab3 does not work
		STATUS[2] = '0': sc_programming slab3 ok
		STATUS[1] = '1': sc_programming slab2 does not work
		STATUS[1] = '0': sc_programming slab2 ok
		STATUS[0] = '1': sc_programming slab1 does not work
		STATUS[0] = '0': sc_programming slab1 ok

8	SET	defines which sc-data set shall be used for slab (partition): SET = '0' default configuration is used SET = '1' alternative configuration is used See section 2.3

7	READBACK	LDA reads back the slow-control data currently stored in the DIF Flash memory. Bit is reset by DIF after completion.
		LDA_readback = '1' readback is active (set by specifier = 0x0100)
		LDA_readback = '0' no action

6 - 4	reserved	can be used to define more partitions if needed later
3	SLAB4	SLAB4 = '1' configure slab3 (partition3) with the sc-data set

Bit no.	Bit Field	Description
		defined by ‘SET’. Bit is reset by DIF after completion. (set by specifier = 0x0008) SLAB4 = ‘0’ no action
2	SLAB3	SLAB3 = ‘1’ configure slab3 (partition3) with the sc-data set defined by ‘SET’. Bit is reset by DIF after completion. (set by specifier = 0x0004) SLAB3 = ‘0’ no action
1	SLAB2	SLAB2 = ‘1’ configure slab3 (partition3) with the sc-data set defined by ‘SET’. Bit is reset by DIF after completion. (set by specifier = 0x0002) SLAB2 = ‘0’ no action
0	SLAB1	SLAB1 = ‘1’ configure slab3 (partition3) with the sc-data set defined by ‘SET’. Bit is reset by DIF after completion. (set by specifier = 0x0001) SLAB1 = ‘0’ no action

Table 14: load_sc_data command register

2.2.7 read_results, BLOCK Transfer command, address 0x000E

15 … 1	0
reserved	SLAB_DIF
R, +0	RS, +0

Bit no.	Bit Field	Description
15 – 2	reserved	reserved
1	SLAB_DIF	SLAB_DIF = ‘1’ results are read from slab via DIF to LDA (set by specifier = 0x0001). Bit is reset by DIF after completion. SLAB_DIF = ‘0’ no action

Table 15: read_results register description

The data section of the BLOCK Transfer packets for the readout data contains an additional header. See section 2.4.1.

2.2.8 DIF Control Register: set_control_reg, BLOCK Transfer command, address 0x0010

By this command the DIF control register is set. A read-access to this register is done by the command read_status_control (see next section).

The DIF CONTROL REGISTER (16-bit, RW, +0):

Bit no.	Bit Field	Description
15-6	CR15-CR6	to be defined
5	CR5	to be defined
4	CR4	to be defined
3	CR3	to be defined

Bit no.	Bit Field	Description
2	CR2	to be defined
1	CR1	to be defined
0	CR0	CR0 = ‘1’ pre-spill signal is used to indicate an upcoming ‘start_acquire’. DIF prepares for data taking (ACTIVE MODE). start_acquire is sent exactly at spill start. CR0 = ‘0’ pre-spill signal is not used. “start_acquire” is sent from DAQ well in advance before the spill, so that DIF can prepare for data taking.

Table 16: DIF Control Register

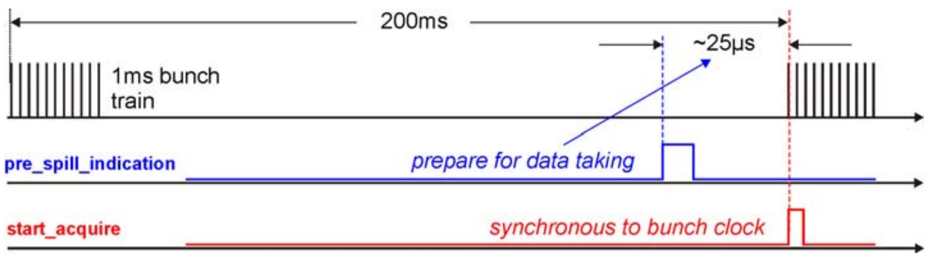

Figure 2: Usage of control register bit 0 (CR0). With CR0='1', the command pre_spill_indication is used from the DIF to prepare for a measurement (e.g. turn on slab power). The start_acquire is used for synchronous indication of the start of the bunch-train.

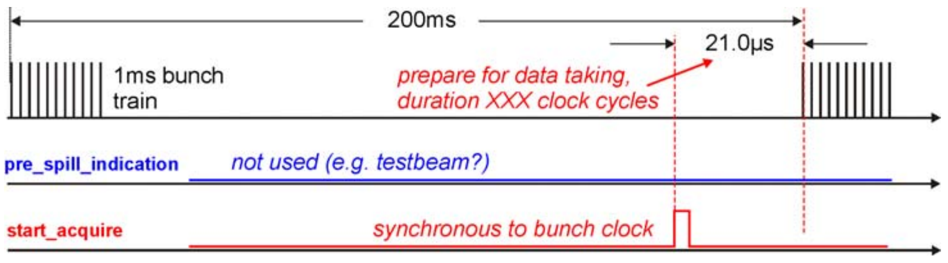
200ms
21.0µs
1ms bunch train
prepare for data taking, duration XXX clock cycles
pre_spill_indication
not used (e.g. testbeam?)
start_acquire
synchronous to bunch clock

Figure 3: Usage of control register bit 0 (CR0). With CR0='0', the command pre_spill_indication is not used. The DIF prepares for a measurement (e.g. turn on slab power) following the start_acquire that is sent XXX clock cycles (to be defined) before the start of the bunch-train.

2.2.9 read_status_control, BLOCK Transfer command, address 0x0012

15 … 2	1	0
reserved	STATUS	CONTROL
R, +0	R, +0	R, +0

Bit no.	Bit Field	Description
15 – 2	reserved	reserved
1	STATUS	STATUS = ‘1’ read DIF Status Register (set by specifier = 0x0002). Bit is reset by DIF after completion. STATUS = ‘0’ no action
0	CONTROL	CONTROL = ‘1’ read DIF Control Register (set by specifier = 0x0001). Bit is reset by DIF after completion.

Bit no.	**Bit Field**	**Description**
		CONTROL = ‘0’ no action

Table 17: read_status_control register description

The DIF STATUS1 REGISTER (16-bit, R, +0):

Bit no.	**Bit Field**	**Description**

12 - 9	ST12 – ST9	ST12-ST9 = ‘0000’ DIF is in SLEEP mode,
		ST12-ST9 = ‘0001’ DIF is in IDLE mode,
		ST12-ST9 = ‘0010’ DIF is in SYNC mode,
		ST12-ST9 = ‘0011’ DIF is in LOOP mode
		ST12-ST9 = ‘0100’ DIF is in ACTIVE mode,
		ST12-ST9 = ‘0101’ DIF is in READOUT mode,
		ST12-ST9 = ‘0110’ DIF is in CONFIG mode current_mode
		ST12-ST9 = ‘0111’ DIF is in DEBUG mode,
		ST12-ST9 = ‘1000’ DIF is in CALIBRATE mode
		ST12-ST9 = ‘1111’ DIF is in ERROR mode

8	ST8	ST8 = ‘1’: current operation caused timeout (watchdog triggered) ST8 = ‘0’: no timeout
7	ST7	ST7 = ‘1’: some supply voltages/currents show bad values ST7 = ‘0’: power consumption ok

6	ST6	ST6 = ‘1’: commands from LDA are sent to DIF-DIF link
		ST6 = ‘0’: DIF-DIF link not active

5	ST5	ST5 = ‘1’: Commands from DIF-DIF link are used
		ST5 = ‘0’: Commands from LDA-DIF interface are used

4	ST4	ST4 = ‘1’: Slow Control programming of ASICs (slabs) shows errors (readback). See which partition in section 2.2.6 ST4 = ‘0’: SC_programming ok

3	ST3	ST3 = ‘1’: temperature too high (in-detector or DIF)
		ST3 = ‘0’: temperature ok

2	ST2	ST2 = ‘1’: Command rejected: DIF is active with other command. ST2 = ‘0’: incoming command is executed

1	ST1	ST1 = ‘1’: incoming command unknown
		ST1 = ‘0’: incoming command ok

0	ST0	ST0 = ‘1’: CRC of incoming frame shows errors ST0 = ‘0’: Incoming frame is valid (no transmission errors)

Table 18: DIF Status Register

The DIF STATUS2 REGISTER (16-bit, R, +0):

Bit no.	**Bit Field**	**Description**
12 - 9		
8		
7		
6		
5		

Bit no.	**Bit Field**	**Description**
4		
3		
2		
1	EndReadout	EndReadout='1': The last ASIC in readout chain has sent a EndReadout to DIF. This bit is cleared by DIF on a 'start_readout' command. EndReadout='0': no EndReadout has arrived at DIF
0	SCASat	SCASat='1': One of the ASICs has sent a SCASat to DIF. This bit is cleared by DIF on a 'start_acquire' command SCASat='0': no SCASat from ASICs arrived at DIF

Table 18: DIF Status Register

2.2.10 readout_info, BLOCK Transfer command, address 0x0014

15 7	6	5	4	3	2	1	0
reserved	ALL	SERIAL	VERSION	ID	DATE	FWvers	FWdate
R, +0	R, +def	R, +def	R, +def	R, +def	R, +def	R, +def	R, +def

Bit no.	**Bit Field**	**Description**
15 – 7	reserved	reserved
6	ALL	ALL = '1' read all 'info' registers (set by specifier = 0x0040). Bit is reset by DIF after completion. ALL = '0' no action
5	SERIAL	SERIAL = '1' read DIF SERIAL number (set by specifier = 0x0020). Bit is reset by DIF after completion. SERIAL = '0' no action
4	VERSION	VERSION = '1' read DIF board version number (set by specifier = 0x0010). Bit is reset by DIF after completion. VERSION = '0' no action
3	ID	ID = '1' read DIF board ID number (set by specifier = 0x0008). Bit is reset by DIF after completion. ID = '0' no action
2	DATE	DATE = '1' read DIF production date (set by specifier = 0x0004). Bit is reset by DIF after completion. DATE = '0' no action
1	FWvers	FWvers = '1' read DIF Firmware version number (set by specifier = 0x0002). Bit is reset by DIF after completion. FWvers = '0' no action
0	FWdate	FWdate = '1' read DIF Firmware date (set by specifier = 0x0001). Bit is reset by DIF after completion. FWdate = '0' no action

Table 19: readout_info register description

Each of the information (DIF SERIAL number, DIF board version number, DIF board ID number, DIF production date, …) are 16-bit and part of the DIF firmware (cannot be changed via LDA).

2.2.11 readback_sc, BLOCK Transfer command, address 0x0016

15 … 9	8 … 1	0
reserved	Last_Access	DIF_LDA
R, +0	R, +0	RS, +0

Bit no.	Bit Field	Description
15 – 9	reserved	reserved
8 – 1	Last_Access	Contents of the latest incoming 'header' (first 16-bit data vector, of which only 8-bit carry header information): for header see section 2.4.2
0	DIF_LDA	DIF_LDA = '1' slow-control data is transferred from DIF to LDA. Bit is reset by DIF after completion automatically. DIF_LDA = '0' no action

Table 20: transfer_sc_data register description

2.2.12 FPGA firmware, BLOCK Transfer command, address 0x0018

15 … 2	1	0
reserved	CRC	LDA_DIF
R, +0	R, +0	RS, +0

Bit no.	Bit Field	Description
15 – 2	reserved	reserved
1	CRC	CRC = '1' CRC-check ok for last LDA-DIF data transfer CRC = '0' transmission errors in current data set
0	LDA_DIF	LDA_DIF = '1' DIF firmware data is transferred from LDA to DIF (set by specifier = 0x0001). Bit is reset by DIF after completion automatically. LDA_DIF = '0' no action

Table 21: FPGA firmware download register description

2.2.13 sel_command_input, BLOCK Transfer command, address 0x001A

15 … 2	1	0
reserved	RES_SEL	INPUT_SEL
R, +0	RS, +0	RW, +0

Bit no.	Bit Field	Description
15 – 2	reserved	reserved
1	RES_SEL	RES_SEL = '1' input selection logic is reset for 4 clock cycles. Bit is reset by DIF after completion automatically. (set by specifier=0x0002) RES_SEL= '0' no action
0	INPUT_SEL	INPUT_SEL = '1' DIF-DIF input is used

Bit no.	Bit Field	Description
		(set by specifier = 0x0001). INPUT_SEL = '0' LDA-DIF input is used (**standard conf.**) (set by specifier = 0x0000).

Table 22: select command input register description

2.2.14 pre_spill_indication, BLOCK Transfer command, address 0x001C

15 1	0
reserved	PRE_SPILL
R, +0	RS, +0

Bit no.	Bit Field	Description
15 – 1	reserved	reserved
0	PRE_SPILL	PRE_SPILL = '1': a "start_acquire is expected within XXXX clock cycles. Puts DIF into "ACTIVE" mode. Bit is reset by DIF after completion automatically. (set by specifier=0x0001) PRE_SPILL = '0': no action

Table 23: pre_spill_indication register description

2.2.15 read_all, BLOCK Transfer command, address 0x001E

15 1	0
reserved	R_ALL
R, +0	RS, +0

Bit no.	Bit Field	Description
15 – 1	reserved	reserved
0	R_ALL	R_ALL = '1': readout of all DIF registers by a BLOCK Transfer from DIF to DAQ (set by specifier=0x0001). Register with smallest address is first in BLOCK Transfer frame. R_ALL = '0': no action

Table 24: read_all register description

2.2.16 gen_fcmd, BLOCK Transfer command, address 0x0020

15 1	0
reserved	fcmd
R, +0	RS, +0

Bit no.	**Bit Field**	**Description**
15 – 1	reserved	reserved
0	fcmd	fcmd = ‘1’: a Fast Command is generated inside DIF from an incoming BLOCK Transfer frame. For debugging only. fcmd = ‘0’: no action

Table 25: gen_fcmd register description

2.2.25 read_poll_reg, BLOCK Transfer command, address 0x0084

15 … 1	0
reserved	poll
R, +0	RS,+0

Bit no.	**Bit Field**	**Description**
15 – 1	reserved	reserved
0	poll	poll = ‘1’: DESY testbeam signal (**AHCAL specific**): Labview tests if DIF has finished the measurement by polling the register ‘DONE’ (see description below). Standard read access. poll = ‘0’: no action

Table 26: read_poll_reg register description

‘DONE’ register description (16 bit, read only):

15 … 11	10	9	8	7 … 3	2	1	0
reserved	reserved	reserved	reserved	TRIG	BUSY	SPILL	DONE
R	R	R	R	R	R	R	R

DONE=1: Measurement completed, SPILL has ended.
SPILL=1: SPILL is still active.
BUSY=1: DIF is busy due to readout or data taking. No new start of measurement allowed. This is a copy of the hardware output.
TRIG (5bits): Number of received triggers during SPILL. If number exceeds 16, overrun in ASICs.

The ‘DONE’ register is cleared after a received command ‘Activate’ (see section 2.2.26) from Labview.

2.2.26 Activate, BLOCK Transfer command, address 0x0086

15 … 1	0
reserved	activate
R, +0	RS,

Bit no.	**Bit Field**	**Description**
15 – 1	reserved	reserved
0	activate	activate = ‘1’: DESY testbeam measurement is started from

Bit no.	**Bit Field**	**Description**
		Labview. DIF waits after a received ‘activate’ for an external hardware signal from testbeam electronics ‘spill_structure’ to start the real measurement. **AHCAL specific!** activate = ‘0’: no action

Table 27: Activate register description

2.3 address map information (preliminary, to be modified!)

The addresses in this section refer to the type_modifier sent in the incoming BLOCK Transfer commands. So the addresses are the same as in table 8.

type_modifier address range (16-bit hex)	**function**	**remarks**
0x0000 – 0x0090	Block Transfer Commands	
0x1000 – 0x1FFF	slab1: default sc_data set	
0x2000 – 0x2FFF	slab1: alternative sc_data set	
0x3000 – 0x3FFF	slab2: default sc_data set	
0x4000 – 0x4FFF	slab2: alternative sc_data set	
0x5000 – 0x5FFF	slab3: default sc_data set	
0x6000 – 0x6FFF	slab3: alternative sc_data set	
0x7000 – 0x7FFF	slab4: default sc_data set	
0x8000 – 0x8FFF	slab4: alternative sc_data set	

Table 26: address map of the type_modifier for the BLOCK Transfers. Preliminary!!

2.4 Slow-Control and Readout data packet formats

In this section, the BLOCK-transfer data packet format for the transfer of slow-control data and readout data is described [3].

2.4.1 readout data – BLOCK transfer packet format

The Block Transfer data packets for the transfer of readout data are defined in table 25. The overall format complies with section 1.2. Inside the data block, there is further information about the origin of the data inside the packet.

Field	**Subfield**	**Comments**
PACKETTYPE (16b)		
PACKETID (16b)		
TYPEMODIFIER (16b)		
DATALENGTH (16b)		
DATA	localDIFID (6b) + ROpacketID (10b)	
	ROLastPacket (1b) + ROChainID (3b) + ROSpillID (12b)	
	ROCData	1 to 505 16bit-words
CRC (16b)		

Table 27: Readout data packet structure

localDIFID (6b) : address of the DIF
ROpacketID (10b): counter of readout packets. ROpacketID is reset at the beginning of a readout sequence.
ROLastPacket (1b): '1' indicates the last packet of a readout operation
ROCChainID (3b): number ('address') of the readout chain (partition, maybe=slab). The ROChainID forms together with the localDIFID and the ChipID (see below) a full address.
ROSpillID (12b): counter of the spills (bunch trains). Starts at '0' after an overflow.

When the data section of an ASIC ends in the packet, two further 16-bit vectors have to be added, forming the **'End-of-Chip-data-field':**

1) '11' + ChipType (2b) + ChipAcqMode (2b) + ChipID (10b)
2) '10' + DataSize (14b)

ChipType: '01' for SKIROC, '10' for SPIROC, '11' for HARDROC
ChipAcqMode (for HARDROC): '01' digital, '10' analogue
ChipID: ChipID sent from the ROC inside the data.
DataSize: number of 16-bit vectors containing readout data of the respective ROC

The BLOCK Transfer packets shown in table 25 have a maximum packet length of 1kByte. If the data of one ASIC is smaller than 1kByte, the data of the next ASIC is appended (interleaved) into the same packet. The chains are treated separately (in parallel) for this packet generation. The resulting packet stream from DIF to the LDA (or DCC) is shown in Fig. 4.

At the end of the readout of all chains, the DIF adds the so called **ROSummary** (16-bit word, cf. Fig. 4 as well):

- '0100 0100' + no. ROCs connected to the DIF (8bits), at complete (regular) readouts of all chains.
- '0100 0101' + no. ROCs connected to the DIF (8bits), at 'forced stops' of the readout ('stop_readout' command, see section 2.1.3).

Finally, the readout may be concluded by the so called ROOptions trailer [3]. But the ROOptions are not mandatory.

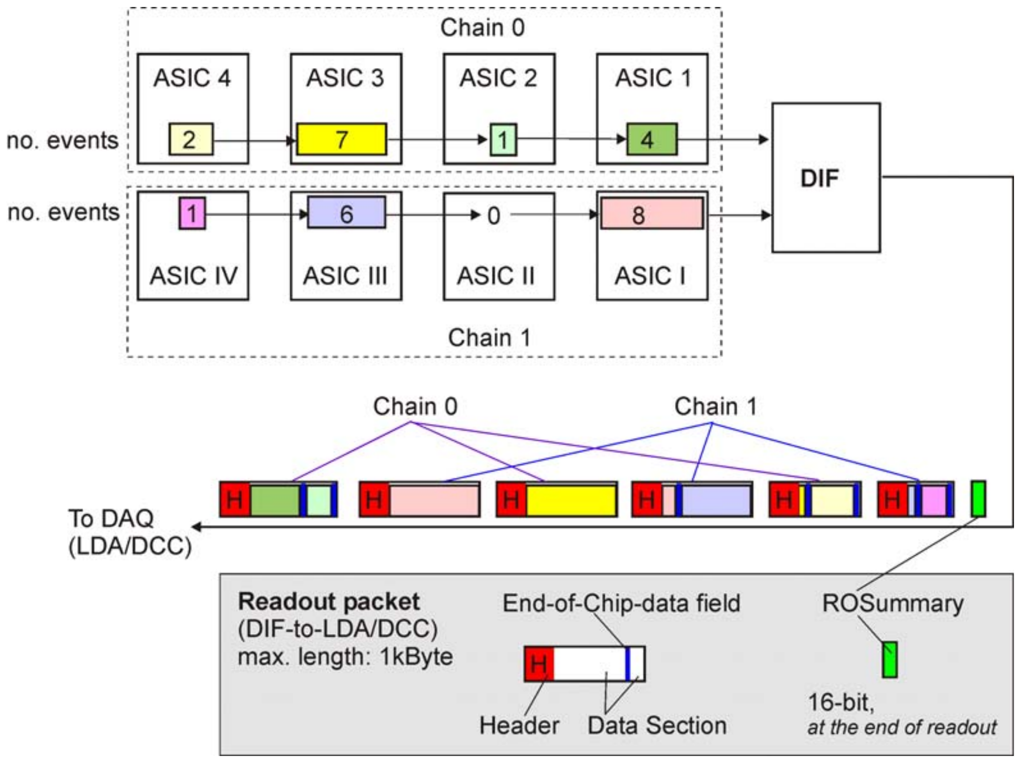

Figure 4: Output packets for a readout process for (exemplary) 2 readout chains.

2.4.2 slow-control data – BLOCK transfer packet format

The BLOCK Transfer data packets for the transfer of slow-control data are defined in table 26. The overall format complies with section 1.2. Inside the data block, there is further information about the destination of the data inside the packet. The packet is intentionally similar to the readout data packet.

Field	Subfield	Comments
PACKETTYPE (16b)		
PACKETID (16b)		
TYPEMODIFIER (16b)		
DATALENGTH (16b)		
DATA	localDIFID (6b) + ROpacketID (10b)	16 bit
	SCDataSet (1b) + ROChainID (3b) + ASICAddress (12b)	16 bit
	SCData	1 to 505 16bit-words
CRC (16b)		

Table 28: Block Transfer packet structure for the transfer of Slow-Control data

SCDataSet (1b): slow-control data in packet is default (‘0’) or alternative (‘1’) set.
ASICAddress (12b): address of the ROC the slow-control data is for.
all others: see section 2.4.1

3 DIF States and State Diagram

In the first DIF version, the top level finite-state-machine (FSM) has a simple structure with only three general (top-level) states as shown in Fig. 4. Additionally, the system can make use of "programmed" config/status registers. The global DAQ supervises global command sequencing and ensures that current operations are not disturbed by other operations. More complex FSMs could be used for the DIF-ASICs (ASICs inside the detector) interface modules, but these are "independent" to ensure the sequencing of the chips operation would not be disturbed/interrupted by other DIF functions (in a given global state of the DIF).

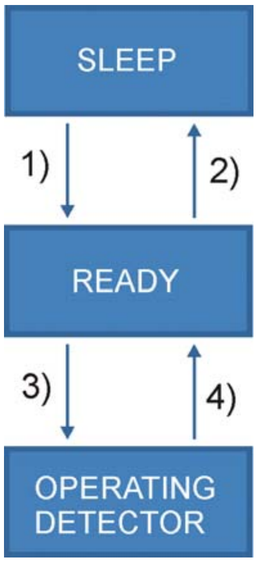

Figure 5: DIF States

SLEEP: Detector is powered-down, DIF still communicates with the DAQ (LDA). DIF configuration (register write and read) is possible. ASICs slow-control configuration can take place in this mode.
READY: All operations are allowed. ASICs slow-control configuration can take place in this mode. Measurements can only be started if the detector is properly configured (slow control data is loaded into ASICs), no errors have occurred.
OPERATING DETECTOR: DIF can only execute commands that do not disturb the current data-taking/data-conversion/readout operation (i.e. no slow-control loading, no debugging). ASICs are operated autonomously by the DIF-ASIC interface. All necessary functions for operation (e.g. trigger) are generated by hardware.

State Transitions: Commands & Conditions (see Fig. 4):
1) Transition executed on: power_on, pre_spill_indication (asynchronous, see Fig. 2), set_DIF_mode, power_pulsing (debugging)
2) Transition executed on: “turn power regulators off” set by option of power_on command, set_DIF_mode, power_pulsing (debugging)
3) this transition is allowed only when slow_control configuration is done and no errors occured.
Transition executed on: start_acquire, pre_spill_indication (synchronous, see Fig. 3), read_results command.
4) Transition executed on: “stop_data taking” set by option of start_acquire command, RAMFull/SCASat condition, “readout_results done” condition, reset command (all options)

References

[1] Marc Kelly’s web page:
http://www.hep.manchester.ac.uk/u/mpkelly/calice/lda/Calice_LDA_Overview.html

[2] Matthew Warren et al. “DAQ Status and Overview”, CALICE week Manchester, Electronics Readout session II, Sept. 8th-10th, 2008

[3] The DIF developers “Format of the read-out data of the DIF”, version 1.0.1, April 2009

Revision History

Version / Date	Changes *with respect to last version*
1.7 (3.12.2008)	Reference document
1.8 (5.1.2009)	- Revision History added - DIF Status Register defined partly (section 2.2.9) - stop_readout added (FAST Command, new section 2.1.3) - reset_SC (reset of slow-control data only) added (section 2.2.2) - Figure2 and Figure3 added (section 2.2.8)
1.9 (6.2.2009)	- power_on command extended (section 2.2.1) - DIF FSM changed (section 3)
1.10 (24.2.2009)	- STATUS2 register has been added (see section 2.2.9) - command ‘read_results’ changed to “send while receive” (section 2.2.7) - State ‘IDLE’ was renamed to ‘READY’ (section 2.2.3) - ‘ALL’ option added to readout_info command (section 2.2.10) - State transitions have been defined (section 3, preliminary)
1.11 (17.3.2009)	- ‘transfer_sc_data’ command changed: header added. - readback option removed from command ‘load_sc_data’ (replaced by ‘readback_sc’ command) - ‘readback_sc’ command added (new section 2.2.11) - new section for AHCAL specific block transfer commands, preliminary - maximum block length information of BLOCK Transfers (1kByte) added (section 1.2)
1.12 (2.4.2009)	- new BLOCK transfer packet format for slow-control and readout data (new section 2.4) - commands ‘transfer_sc_data’ and ‘readback_sc’ changed with respect to the new packet formats for readout- and sc-data (sections 2.2.5 and 2.2.11). - Reference to section 2.4.1 added for read_results command (section 2.2.7). - document’s title changed - introduction shortened (section 0) - Section 0.2 shortened
1.13 (18.5.2009)	- new AHCAL-specific BLOCK Transfer Commands (table 8)
1.14 (17.6.2009)	- last column in table 8 removed (this is defined in section 3) - revision of the AHCAL specific commands (table 8)
1.15 (2.12.2009)	- “specifier” (now instead of first data vector) added to BLOCK Transfer frames (section 1.2) - commands “read_all” and “gen_fcmd” added (sections 2.2.15 and 2.2.16) - AHCAL specific: Calibrate DACs command added

	- AHCAL specific: poll-register for DESY testbeam added
1.16 (19.1.2009)	- AHCAL specific commands ‘Activate’ and ‘Read_Poll_Reg’ added (see sections 2.2.25 and 2.2.26)