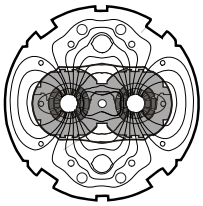
CERN
CH-1211 Geneva 23
Switzerland



the
Large
Hadron
Collider
project

LHC Project Document No.
LHC-

CERN Div./Group or Supplier/Contractor Document No.
AT-MEL

EDMS Document No.
-

Date: 2004-06-02

[Draft]

THE COMMUNICATION BETWEEN THE ACQUISITION AND MONITORING CONTROLLERS AND THE QPS GATEWAYS

Abstract

This document describes the basic parameters and the communication protocol relevant for the communication between the DQAMC, DQAMG and DQAMS type acquisition and monitoring controllers and a DQGTW front-end computer via the QPS WorldFip network.

Prepared by :	***Checked by :***	
R. Denz AT-MEL Reiner.Denz@cern.ch		

History of Changes

Rev. No.	*Date*	*Pages*	*Description of Changes*

Table of Contents

1. GENERAL COMMUNICATION PARAMETERS

Basic communication parameters for the fieldbus connection between the acquisition and monitoring controllers and the QPS gateway are summarized in Table 1. The communication protocol will be based on three different data types (see Table 2). Analog values will be transmitted as raw data coded into 16 Bit integers.

Table 1: General communication parameters

Parameter	***Value***
Bus speed	1 Mbit/s
Maximum number of WorldFip clients	60
Byte order acquisition and monitoring controller	Big endian
Used fieldbus addresses	1 to 126, 128 to 255[1]

Table 2: Data types

Data type	***Bits***	***Bytes***	***Value Range***
unsigned char	8	1	0 to 255
unsigned int	16	2	0 to 65535
unsigned long	32	4	0 to 4294967295

2. HARDWARE

All controller hardware is based on the ADuC831 micro-converter chip, which has an 8052 compatible programming interface, and the VY27257 MicroFip ASIC. All controllers have various analog and digital I/O ports built-in. The DQAMC communicates with the associated hardware via a SPI (serial peripheral interface) link, the DQAMG via an I2C bus link and the DQAMS uses an internal I2C bus to communicate with its numerous digital I/O channels.

3. ASSOCIATED HARDWARE

Tables 3 to 6 list the associated hardware for each controller type. The numbers in brackets are the actual addressable devices. These figures are important for the proper implementation of the commands related to equipment names and test modes.

[1] Address 0 is reserved for the bus arbiter, address 127 for diagnostic purposes.

Table 3: Associated hardware Local Protection Units DQLPU

Controller type	*DQQDL*	*DQHDS*
DQAMC type MB	1(2)	4
DQAMC type MQ	2(4)	2

Local Protection Units come in two variants one produced by GPV Denmark (LHC part identification numbers HCDQLPU001-GP000001 to HCDQLPU001-GP000200 and HCDQLPU002-GP000001 to HCDQLPU001-GP000070) and another produced by ECIL India (LHC part identification numbers HCDQLPU001-EL000001 to HCDQLPU001-EL001090 and HCDQLPU002-EL000001 to HCDQLPU001-EL000435). The DQQDL type circuit boards made by GPV are equipped with an ADuC812 microcontroller, whereas the ECIL devices are shipped with an ADuC831. Unfortunately both chips are not fully compatible and the firmware has to be modified respectively [1].

Table 4: Associated hardware Global Protection Units DQGPU

Controller type	*DQQDC*	*DQQDG*	*DQQDI*	*DQQDT*	*DQHDS*	*CIRCUITS*
DQAMG type A	8(16)	4(8)	0	0	0	4
DQAMG type B	6(12)	0	4(8)	0	8	2
DQAMG type C	6(12)	0	0	1(6)	8	1

Controller type	*DQQDC*	*DQQDB slave*	*DQQDB master*	*CIRCUITS*
DQAMG type D	8(16)	10(20)	3	3

Table 5: Associated hardware quench loop controllers

Controller type	*DQQLC master (even point)*	*DQQLC slave (odd point)*	*CIRCUITS*
DQQLC	1	1	3

Table 6: Associated hardware energy extraction systems

Controller type	*DQEMC*	*DQPIB/DQPIQ*
DQAMS type 13 kA	0	1
DQAMS type 600 A	1	0

4. MICROFIP VARIABLE DEFINITION

Minimum variable size is 8 Bytes (= 1 data block; maximum number of data blocks is 15, i.e. 120 Bytes). VAR 7 (time) is global and generates an external interrupt for the ADuC831 micro-converters, which allows to synchronize these agents with respect to the accelerator time. The proposed variable definitions favour periodic traffic, which might be wishful with respect to the DQAMC firmware constraints. In total all acquired data will be transferred via 4 produced variables baptised DATA0 to DATA3. So far 112 Bytes out of the 120 Bytes of the MicroFip memory will be used.

Table 7: MicroFip variable definition.

MicoFip variable	*Identifier (HEX)*	*Name*	*Size*	*Type*	*Status*
VAR 0	00xy[2]	DATA0	24 Bytes	Produced	Enabled
VAR 1	01xy	N/A	N/A	Consumed	Disabled
VAR 2	02xy	DATA1	24 Bytes	Produced	Enabled
VAR 3	03xy	N/A	N/A	Consumed	Disabled
VAR 4	04xy	DATA2	24 Bytes	Produced	Enabled
VAR 5	05xy	COMMAND	8 Bytes	Consumed	Enabled
VAR 6	06xy	DATA3	24 Bytes	Produced	Enabled
VAR 7	3000	TIME	8 Bytes	Consumed / Global	Enabled

Refreshment for variables VAR 0, VAR 2, VAR 4 and VAR 6 is set to 250 ms, promptness for VAR 5 is 5 s.

5. WORLDFIP TRANSMISSION TIMES

All transmission times are calculated according to the WorldFip protocol and listed in Table 7 and 8.

Table 8: General parameters for transmission times.

Bus parameter	*Value*	*Remark*
TMAC	1 μs	= 1 Mbit/s
TR	10 x TMAC = 10 μs	Wait time
QUESTION FRAME	64 x TMAC = 64 μs	
TURNAROUND TIME	2 x TR = 20 μs	
RESPONSE FRAME	64 x TMAC + N x 8 x TMAC	N = number of Data bytes
TRANSACTION TIME	QUESTION + TURNAROUND + RESPONSE	

[2] xy is the hexadecimal station number

Table 9: Calculated transmission times.

Variable	*Size [Bytes]*	*TX time [ms]*	*Efficiency [%]*	*Useful throughput [kBit/s]*
DATA0	24	0.340	56.47	564.71
DATA1	24	0.340	56.47	564.71
DATA2	24	0.340	56.47	564.71
DATA3	24	0.340	56.47	564.71
COMMAND	8	0.212	30.19	301.89
TIME	8	0.212	30.19	301.89

6. WORLDFIP MACRO-CYCLE

The DQAMC acquisition time is based on a quartz crystal with a nominal frequency of 11.0592 MHz and a precision of 30 ppm. One machine cycle of the ADuC831 corresponds to 12 clocks = 1.085 μs ± 0.00003 μs. An internal timer will update the acquisition time every millisecond with a maximum error of Δt = 0.03 μs. The acquisition timing error increases linearly with the refreshment period, i.e. the length of the macro-cycle. From the point of operation of the DQGTW computer a slightly relaxed macro-cycle is preferable. In the following a macro-cycle of 200 ms is proposed and calculated for 60 clients (DQAMC controllers) present on the bus. It is noteworthy that in case of standard operation (= logging) the DQGTW is not required to treat all the incoming information.

Table 10: WorldFip macro-cycle of 200 ms length

Action	*Time [ms]*	*Total time [ms]*
TIME	0.212	0.212
DELAY	10	10.212
COMMAND [1..60]	60 · 0.212 = 12.720	22.932
DELAY	5	27.932
DATA0 [1..60]	60 · 0.340 = 20.400	48.332
DELAY	5	53.332
DATA1 [1..60]	60 · 0.340 = 20.400	73.732
DELAY	5	78.732
DATA2 [1..60]	60 · 0.340 = 20.400	99.132
DELAY	5	104.132
DATA3 [1..60]	60 · 0.340 = 20.400	124.532
DELAY	75.468	200.000

The total data transmission rate of the cycle is (60 x (4 x 24 +8) +8)/200 ms = 31.24 kbyte/s.

7. HARDWARE PARAMETERS FOR ANALOG INPUT SIGNALS

In general the DQAMC controllers and the associated quench detectors are acquiring data with a sampling frequency of 200 Hz. This rate is only available for post mortem data as in case of the maximum available acquisition frequency with a WorldFip macro cycle of 200 ms length is 5 Hz.

Table 11: Hardware parameters for analog input channels.

Device	***Chip***	***ENOB***	***Channel(s)***	***Scale***	***Offset***	***Resolution*[mV]**
DQQDL	ADuC812	11.23	U_1, U_2	0.10067	-206.2	172
DQQDL	ADuC812	11.23	U_QS0	0.0001221	-0.25	0.21
DQQDL	ADuC831	11.47	U_1, U_2	0.10067	-206.2	145
DQQDL	ADuC831	11.47	U_QS0	0.0001221	-0.25	0.18
DQQDC	ADuC834					
DQQDB	ADuC834					
DQQDG						
DQAMC	ADuC831	11.47	U_HDS_n	0.30669	0.0	1.76
DQAMG	ADuC831	11.47	U_HDS_n	0.30669	0.0	1.76
DQAMS	ADuC831	11.47				

8. SIGNALS FOR THE DQAMC – DQGTW COMMUNICATION

The signal definitions are based on LHC-DQ-ES-0003 rev. 1.X. In general a signal name can be created as follows:

Signal name = <Equipment Code>.<Position>:<Quantity>

In case of the DQAMC type controllers, position specifies the location within an LHC half-cell.

Table 12: Signals sent by the DQAMC type MB controller (3 per half-cell)

Equipment code	*Module*	*Quantity*	*Bit*
MB	DQQDL	U_1	12
MB	DQQDL	U_2	12
MB	DQQDL	U_QS0	12
MB	DQQDL	ST_NQD0	1
MB	DQQDL	ST_MAGNET_OK	1
MB	DQQDL	ST_PWR_PERM	1
MB	DQAMC	TIME_STAMP_ACQ	48
MB	DQQDL	TIMESTAMP_QUENCH[3]	48
DQQDL	DQQDL	ST_COHER	1
DQQDL	DQQDL	ST_PWR	1
DQQDL	DQQDL	ST_PWR_PERM	1
DQQDL	DQQDL	TIMESTAMP_OFFSET	8
MB	DQHDS	U_HDS_1	12
MB	DQHDS	U_HDS_2	12
MB	DQHDS	U_HDS_3	12
MB	DQHDS	U_HDS_4	12
DQAMC	DQAMC	CMD_GTW	8
DQAMC	DQGTW	ST_FIP	1
DQAMC	DQAMC	ST_BUS	1
DQAMC	DQAMC	ST_TIMING	1
DQAMC	DQAMC	STATUS	5
DQAMC	DQAMC	ST_OPERATION	2
DQAMC	DQAMC	ST_BOARD_A	1
DQAMC	DQAMC	ST_COM	1
DQAMC	DQAMC	ST_PWR_PERM	1
DQAMC	DQGTW	TIMESTAMP_GTW	64

[3] Generated with the help of the TIME_STAMP_OFFSET signal.

Table 13: Signals sent by the DQAMC type MQ controller (1 per half cell)

Equipment code	*Module*	*Quantity*	*Bit*
MQ	DQQDL	U_1_EXT	12
MQ	DQQDL	U_2_EXT	12
MQ	DQQDL	U_QS0_EXT	12
MQ	DQQDL	ST_NQD0_EXT	1
MQ	DQQDL	ST_MAGNET_OK_EXT	1
MQ	DQQDL	ST_PWR_PERM_EXT	1
MQ	DQAMC	TIME_STAMP_ACQ	48
MQ	DQQDL	TIMESTAMP_QUENCH[4]	48
DQQDL	DQQDL	ST_COHER_EXT	1
DQQDL	DQQDL	ST_PWR_EXT	1
DQQDL	DQQDL	ST_PWR_PERM_EXT	1
DQQDL	DQQDL	TIMESTAMP_OFFSET_EXT	8
MQ	DQQDL	U_1_INT	12
MQ	DQQDL	U_2_INT	12
MQ	DQQDL	U_QS0_INT	12
MQ	DQQDL	ST_NQD0_INT	1
MQ	DQQDL	ST_MAGNET_OK_INT	1
MQ	DQQDL	ST_PWR_PERM_INT	1
DQQDL	DQQDL	ST_COHER_INT	1
DQQDL	DQQDL	ST_PWR_INT	1
DQQDL	DQQDL	ST_PWR_PERM_INT	1
DQQDL	DQQDL	TIMESTAMP_OFFSET_INT	8
MQ	DQHDS	U_HDS_1	12
MQ	DQHDS	U_HDS_2	12
DQAMC	DQAMC	CMD_GTW	8
DQAMC	DQGTW	ST_FIP	1
DQAMC	DQAMC	ST_BUS	1
DQAMC	DQAMC	ST_TIMING	1
DQAMC	DQAMC	STATUS	5
DQAMC	DQAMC	ST_OPERATION	2
DQAMC	DQAMC	ST_BOARD_A	1
DQAMC	DQAMC	ST_COM	1
DQAMC	DQAMC	ST_PWR_PERM	1
DQAMC	DQGTW	TIMESTAMP_GTW	64

[4] Generated with the help of the TIME_STAMP_OFFSET_EXT and TIME_STAMP_OFFSET_INT signal.

9. SIGNALS FOR THE DQAMG – DQGTW COMMUNICATION

Coming soon …

Table 14: Signals sent by the DQAMG type A controller

Equipment code	*Module*	*Quantity*	*Bit*

Table 15: Signals sent by the DQAMG type B controller

Equipment code	*Module*	*Quantity*	*Bit*

Table 16: Signals sent by the DQAMG type C controller

Equipment code	*Module*	*Quantity*	*Bit*

Table 17: Signals sent by the DQAMG type D controller

Equipment code	*Module*	*Quantity*	*Bit*

10. SIGNALS FOR THE DQQLC – DQGTW COMMUNICATION

Table 18: Signals sent by the DQQLC controller

Equipment code	*Module*	*Quantity*	*Bit*

11. SIGNALS FOR THE DQAMS – DQGTW COMMUNICATION

Table 20: Signals sent by the DQAMS type 13 kA controller

Entity	*Transmitter*	*Signal*	*Bit*

Table 21: Signals sent by the DQAMS type 600 A controller

Entity	*Transmitter*	*Signal*	*Bit*

12. IMPLEMENTATION OF THE COMMUNICATION PROTOCOL INTO THE CONTROLLER FIRMWARE

The implementation of the defined data into code will finally determine the required transmission capacity. The programming language of the DQAMC, DQAMG and DQAMS controllers is ANSI C with extensions specific to the ADuC831 micro-converter. The code used for the transmission of timing and command information is common to all kind of controllers and hardware configurations. Acquired data handling however requires specific coding.

12.1 TIME

Timing (**TIMESTAMP_GTW**) information is encoded within the **timeqps_wf_struct** structure, which has a size of 8 bytes and is allocated to the consumed variable VAR7 TIME.

```
struct timeqps_wf_struct{
      unsigned long time_seconds; // seconds since The Epoch (01.01.1970)
      unsigned long time_nanoseconds; // nanoseconds
};
```

12.2 COMMAND

All commands (**CMD_GTW**) are transmitted within a structure named **commandqps_wf_struct** with a total size of 3 bytes. This structure is allocated to the consumed variable VAR5 COMMAND.

```
struct commandqps_wf_struct{
      unsigned char command; // CMD_GTW 1 Byte
      unsigned int number_of_buffer; // number of  buffer to be sent; 0 in case of
                                     // logging;
};
```

The following table shows the definition of the various command values. This definition is valid for all DQAMC, DQAMG and DQAMS types and takes into account that a controller may supervise up to 4 circuits and up to 12 different devices. A device is always considered as a set of two redundant circuit boards labelled board A and board B. Test mode can only be entered by sending a sequence of two commands.

Table 22: Definition of the variable **command**.

No.	*Command*	*Value (hex)*	*Scope*
0	NONE	00	DQAMC, DQAMG, DQAMS
1	SELECT_BOARD_A	01	DQAMC, DQAMG
2	SELECT_BOARD_B	02	DQAMC, DQAMG
3	PREPARE_POSITIVE_TEST_MODE	03	DQAMC, DQAMG
4	CANCEL_TEST_MODE	04	DQAMC, DQAMG, DQAMS
5	PREPARE_NEGATIVE_TEST_MODE	05	DQAMC, DQAMG
6	SEND_SNAPSHOT_C_0	06	DQAMC, DQAMG, DQAMS
7	SEND_SNAPSHOT_C_1	07	DQAMC, DQAMG, DQAMS
8	SEND_SNAPSHOT_C_2	08	DQAMC, DQAMG, DQAMS
9	SEND_SNAPSHOT_C_3	09	DQAMC, DQAMG, DQAMS
10	RESET	0A	DQAMC, DQAMG, DQAMS

No.	Command	Value (hex)	Scope
11	SEND_NAME_C_0	0B	DQAMC, DQAMG, DQAMS
12	SEND_NAME_C_1	0C	DQAMG
13	SEND_NAME_C_2	0D	DQAMG
14	SEND_NAME_C_3	0E	DQAMG
15	SEND_PATTERN_C_0	0F	DQAMC, DQAMG, DQAMS
16	SEND_PATTERN_ C_1	10	DQAMG
17	SEND_PATTERN_ C_2	11	DQAMG
18	SEND_PATTERN_ C_3	12	DQAMG
19	SEND_BUFFER_ C_0	13	DQAMC, DQAMG, DQAMS
20	SEND_BUFFER_ C_1	14	DQAMG
21	SEND_BUFFER_ C_2	15	DQAMG
22	SEND_BUFFER_ C_3	16	DQAMG
23	SEND_LOGGING_ C_0	17	DQAMC, DQAMG, DQAMS
24	SEND_LOGGING_ C_1	18	DQAMG
25	SEND_LOGGING_ C_2	19	DQAMG
26	SEND_LOGGING_ C_3	1A	DQAMG
27	SEND_TEMPERATURE_C_0	1B	DQAMC, DQAMG, DQAMS
28	SEND_TEMPERATURE_C_1	1C	DQAMG
29	SEND_TEMPERATURE_C_2	1D	DQAMG
30	SEND_TEMPERATURE_C_3	1E	DQAMG
31	ZERO_CALIBRATE_ON_C_0	1F	DQAMC, DQAMG
32	ZERO_CALIBRATE_ON_C_1	20	DQAMG
33	ZERO_CALIBRATE_ON_C_2	21	DQAMG
34	ZERO_CALIBRATE_ON_C_3	22	DQAMG
35	ZERO_CALIBRATE_OFF_C_0	23	DQAMC, DQAMG
36	ZERO_CALIBRATE_OFF_C_1	24	DQAMG
37	ZERO_CALIBRATE_OFF_C_2	25	DQAMG
38	ZERO_CALIBRATE_OFF_C_3	26	DQAMG
39	RESERVED	27	DQAMC, DQAMG, DQAMS
40	CLOSE_CIRCUIT_BREAKER	28	DQAMS
41	RESERVED	29	DQAMS
42	RESERVED	2A	DQAMS
43	RESERVED	2B	DQAMS
44	RESERVED	2C	DQAMS
45	RESERVED	2D	DQAMS
46	RESERVED	2E	DQAMS
47	RESERVED	2F	DQAMS
48	RESERVED	30	DQAMS
49	RESERVED	31	DQAMS

No.	Command	Value (hex)	Scope
50	ENTER_TEST_MODE_EQ_0	32	DQAMC, DQAMG
51	ENTER_TEST_MODE_ EQ_1	33	DQAMC, DQAMG
52	ENTER_TEST_MODE_ EQ_2	34	DQAMG
53	ENTER_TEST_MODE_ EQ_3	35	DQAMG
54	ENTER_TEST_MODE_ EQ_4	36	DQAMG
55	ENTER_TEST_MODE_ EQ_5	37	DQAMG
56	ENTER_TEST_MODE_ EQ_6	38	DQAMG
57	ENTER_TEST_MODE_ EQ_7	39	DQAMG
58	ENTER_TEST_MODE_ EQ_8	3A	DQAMG
59	ENTER_TEST_MODE_ EQ_9	3B	DQAMG
60	ENTER_TEST_MODE_ EQ_10	3C	DQAMG
61	ENTER_TEST_MODE_ EQ_11	3D	DQAMG

12.3 DATA SEGMENT DQAMC

All data produced by the DQAMC are organised in structure named **dataqps_wf_struct**. The total size of the structure is 24 Bytes and corresponds to the size of one of the produced variables DATA0 to DATA3. The structure is capable of storing all data required for logging. In case of a post mortem event the acquired data will be transmitted in pieces with one block per macro cycle. This block is divided into 4 sub blocks, each of them allocated to one of the 4 produced variables.

```
struct dataqps_wf_struct{
    unsigned char status_command_gtw;// STATUS
    unsigned int number_of_buffer; // number of post mortem block 0..65535
    unsigned char sub_buffer; // number of sub block 0..3
    unsigned long acquisition_time_sec; // TIMESTAMP_ACQ seconds after 1/1/1970
    unsigned int acquisition_time_millisec; // TIMESTAMP_ACQ ms
    unsigned int status; // status bits of DQAMC and DQQDL
    unsigned char analog_value[12]; // analog values DQAMC and DQQDL
};
```

In case of a <send test pattern> command **number_of_buffer** contains the WorldFip address of the client. In case of test mode **number_of_buffer** contains the number of equipment currently under test as long as a <send buffer> command is issued by the gateway.

12.3.1 STATUS_COMMAND VARIABLE

The definition of the variable <status_command_gtw> is common for all type of controllers and is specified in Table 16. Figure 1 shows the state diagram of DQAMC type controller.

Table 23: Definition of the variable **status_command_gtw**.

Bit	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
Content	STATUS					ST_OPERATION		ST_BOARD_A

Status	***7***	***6***	***5***	***4***	***3***	***HEX (no test/A)***	***HEX (no test/B)***
Logging off	0	0	0	0	0	00	00
Filling buffer	0	0	0	0	1	08	09
Buffer ready	0	0	0	1	0	10	11
Last block	0	0	0	1	1	18	19
Test pattern	0	0	1	0	0	20	21
Reserved	0	0	1	0	1	28	29
Sending post mortem data	0	0	1	1	0	30	31
Sending logging data	0	0	1	1	1	38	39
Sending temperature	0	1	0	0	0	40	41
Auto-zero ON	0	1	0	0	1	48	49
Auto-zero OFF	0	1	0	1	0	50	51
Resetting slaves	0	1	0	1	1	58	59
Reserved	0	1	1	0	0	60	61
Reserved	0	1	1	0	1	68	69
Sending name DQAMC	0	1	1	1	0	70	71
Sending name DQAMG	0	1	1	1	1	78	79
Sending name DQAMS	1	0	0	0	0	80	81
Preparing positive test mode	1	0	0	0	1	88	89
Preparing negative test mode	1	0	0	1	0	90	91
Test in progress Eq_0	1	0	0	1	1	98	99
Test in progress Eq_1	1	0	1	0	0	A0	A1
Test in progress Eq_2	1	0	1	0	1	A8	A9
Test in progress Eq_3	1	0	1	1	0	B0	B1
Test in progress Eq_4	1	0	1	1	1	B8	B9
Test in progress Eq_5	1	1	0	0	0	C0	C1
Test in progress Eq_6	1	1	0	0	1	C8	C9
Test in progress Eq_7	1	1	0	1	0	D0	D1
Test in progress Eq_8	1	1	0	1	1	D8	D9
Test in progress Eq_9	1	1	1	0	0	E0	E1
Test in progress Eq_10	1	1	1	0	1	E8	E9
Test in progress Eq_11	1	1	1	1	0	F0	F1
Error	1	1	1	1	1	F8	F9

Bit	***2***	***1***
Normal	0	0
Positive test	0	1
Negative test	1	0
Send buffer command	1	1

Bit	***0***
Board A selected	1
Board B selected	0

In case of command #3 <PREPARE_POSITIVE_TEST_MODE> and #4 <PREPARE_NEGATIVE_TEST_MODE> the controller will enter in the <Logging off> status with the respective test status bits set.

The following changes of controller status will occur automatically without a triggering command issued by the gateway. All other cases require an appropriate command issued by the gateway.

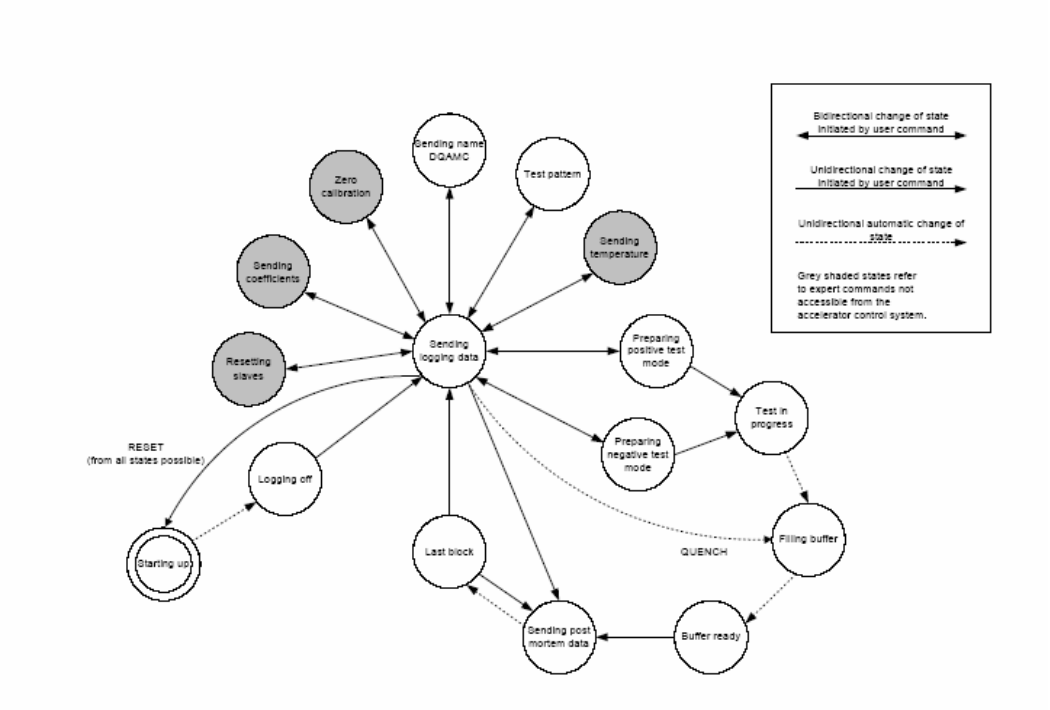


Figure 1: DQAMC state diagram

12.3.2 STATUS VARIABLE

Table 24: Definition of the variable status for DQAMC type MB

High byte								
Bit	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
Content	N/A	N/A	N/A	N/A	ST_NQD0	ST_MAGNET_OK	ST_COHER_OK	ST_PWR_PERM
Entity	N/A	N/A	N/A	N/A	MB	MB	DQQDL	MB

Low byte								
Bit	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
Content	ST_PWR_PERM	ST_COM	ST_BUS	ST_TIMING	N/A	ST_PWR_PERM	N/A	ST_PWR
Entity	DQAMC	DQAMC	DQAMC	DQAMC	N/A	DQQDL	N/A	DQQDL

Table 25: Definition of the variable status for DQAMC type MQ

High byte higher nibble				
Bit	***7***	***6***	***5***	***4***
Content	ST_NQD0_INT	ST_MAGNET_OK_INT	ST_COHER_OK_INT	ST_PWR_PERM_INT
Entity	MQ	MQ	DQQDL #2	MQ

High byte lower nibble				
Bit	***3***	***2***	***1***	***0***
Content	ST_NQD0_EXT	ST_MAGNET_OK_EXT	ST_COHER_OK_EXT	ST_PWR_PERM_EXT
Entity	MQ	MQ	DQQDL #1	MQ

Low byte higher nibble				
Bit	***7***	***6***	***5***	***4***
Content	ST_PWR_PERM	ST_COM	ST_BUS	ST_TIMING
Entity	DQAMC	DQAMC	DQAMC	DQAMC

Low byte lower nibble				
Bit	***3***	***2***	***1***	***0***
Content	ST_PWR_PERM_INT	ST_PWR_PERM_EXT	ST_PWR_INT	ST_PWR_EXT
Entity	DQQDL #2	DQQDL #1	DQQDL #2	DQQDL #1

12.3.3 ANALOG VALUE ENCODING

Table 26 and 27 show the encoding of analog values for DQAMC controllers type A and B. The data will be temporarily stored in the internal logging buffers of the DQAMC and DQQDL devices.

Table 26: Definition of the variable <analog_value> for DQAMC type MB.

Analog_value	*Byte DQGTW*	*Logging buffer DQAMC*	*Logging buffer DQQDL*	*Value*	
1	0D	1	0	U_2 8LSB	
0	0C	0	1	U_1 4MSB	U_2 4MSB
3	0F	3	2	U_1 8LSB	
2	0E	2	3	U_QS0 8LSB	
5	11	5	4	U_HDS_1 4MSB	U_QS0 4MSB
4	10	4	N/A	U_HDS_1 8LSB	
7	13	7	N/A	U_HDS_2 8LSB	
6	12	6	N/A	U_HDS_3 4MSB	U_HDS_2 4MSB
9	15	9	N/A	U_HDS_3 8LSB	
8	14	8	N/A	U_HDS_4 8LSB	
11	17	11	N/A		U_HDS_4 4MSB
10	16	10	N/A	N/A	

Table 27: Definition of the variable <analog_value> for DQAMC type MQ.

Analog_value	*Byte DQGTW*	*Logging buffer DQAMC*	*Logging buffer DQQDL*	*Value*	
1	0D	1	0	U_1_EXT 8LSB	
0	0C	0	1	U_2_EXT 4MSB	U_1_EXT 4MSB
3	0F	3	2	U_2_EXT 8LSB	
2	0E	2	3	U_QS0_EXT 8LSB	
5	11	12 & 5	4	U_HDS_1 4MSB	U_QS0_EXT 4MSB
4	10	4	0	U_1_INT 8LSB	
7	13	7	1	U_2_INT 4MSB	U_1_INT 4MSB
6	12	6	2	U_2_INT 8LSB	
9	15	9	3	U_QS0_INT 8LSB	
8	14	13 & 8	4	U_HDS_2 4MSB	U_QS0_INT 4MSB
11	17	11	N/A	U_HDS_1 8LSB	
10	16	10	N/A	U_HDS_2 8LSB	

Tables 28 to 33 explain the test data generated by a DQAMC type controller and its associated equipment in case of a < SEND_PATTERN_C_0> command. As these data will be encoded within the same functions as normally used for analog data sampling, the command allows the test of the complete communication chain.

Table 28: Test pattern sent by DQAMC type MB.

Value	*12 bit integer*	*Hex*	*Physical value*	*Unit*
U_1	3042	0BE2	100.0	V
U_2	1055	041F	-100.0	V
U_QS0	2457	0999	0.050	V
U_HDS_1	0	0000	0	V
U_HDS_2	978	03D2	250	V
U_HDS_3	1935	078F	500	V
U_HDS_4	3871	0F1F	1000	V

Table 29: Test bytes sent by DQAMC type MB

Byte	*17*	*16*	*15*	*14*	*13*	*12*	*11*	*10*	*0F*	*0E*	*0D*	*0C*
Value	00	0F	1F	8F	73	D2	00	09	99	1F	4B	E2

Table 30: Test bytes received by DQGTW (byte order inversed)

Byte	*17*	*16*	*15*	*14*	*13*	*12*	*11*	*10*	*0F*	*0E*	*0D*	*0C*
Value	0F	00	8F	1F	D2	73	09	00	1F	99	E2	4B

Table 31: Test pattern sent by DQAMC type MQ.

Value	*12 bit integer*	*Hex*	*Physical value*	*Unit*
U_1_EXT	3042	0BE2	100.0	V
U_2_EXT	1055	041F	-100.0	V
U_QS0_EXT	2457	0999	0.050	V
U_1_INT	1055	041F	-100.0	V
U_2_INT	3042	0BE2	100.0	V
U_QS0_INT	1638	0666	-0.050	V
U_HDS_1	0	0000	0	V
U_HDS_2	3871	0F1F	1000	V

Table 32: Test bytes sent by DQAMC type MQ

Byte	*17*	*16*	*15*	*14*	*13*	*12*	*11*	*10*	*0F*	*0E*	*0D*	*0C*
Value	1F	00	F6	66	E2	B4	1F	09	99	1F	4B	E2

Table 33: Test bytes received by DQGTW (byte order inversed)

Byte	*17*	*16*	*15*	*14*	*13*	*12*	*11*	*10*	*0F*	*0E*	*0D*	*0C*
Value	00	1F	66	F6	B4	E2	09	1F	1F	99	E2	4B

Finally tables 34 and 35 explain how analog values sent by the DQAMC type controllers can be converted into physical values.

Table 34: Analog value decoding for DQAMC type MB.

Value	*Scale*	*Offset*	*Unit*	*Remark*	*DQAMC status*
U_1	0.10067	–206.2	V		Sending logging data
U_2	0.10067	–206.2	V		Sending logging data
U_2	-0.61	350.0	°C	DQQDL chip temperature	Sending temperature
U_QS0	0.0001221	-0.25	V		Sending logging data
U_HDS_1	0.30669	0	V		Sending logging data
U_HDS_2	0.30669	0	V		Sending logging data
U_HDS_2	-0.61	350.0	°C	DQAMC chip temperature	Sending temperature
U_HDS_3	0.30669	0	V		Sending logging data
U_HDS_4	0.30669	0	V		Sending logging data

Table 35. Analog value decoding for DQAMC type MQ.

Value	*Scale*	*Offset*	*Unit*	*Remark*	*DQAMC status*
U_1_EXT	0.10067	–206.2	V		Sending logging data
U_2_EXT	0.10067	–206.2	V		Sending logging data
U_2_EXT	-0.61	350.0	°C	DQQDL #0 chip temperature	Sending temperature
U_QS0_EXT	0.0001221	-0.25	V		Sending logging data
U_1_INT	0.10067	–206.2	V		Sending logging data
U_2_INT	0.10067	–206.2	V		Sending logging data
U_2_INT	-0.61	350.0	°C	DQQDL #1 chip temperature	Sending temperature
U_QS0_INT	0.0001221	-0.25	V		Sending logging data
U_HDS_1	0.30669	0	V		Sending logging data
U_HDS_2	0.30669	0	V		Sending logging data
U_HDS_2	-0.61	350.0	°C	DQAMC chip temperature	Sending temperature

In case the DQAMC input stages are tested with a high impedance voltage source, conversion factors for U_HDS_1 to U_HDS_4 must be changed to 0.25835.

In case of the <SEND_TEMPERATURE_C0> command, the reading of the on-chip temperature sensors of the ADuC812 / ADuC831 devices will be mapped into the analog data segment. All other data will be read as usual.

12.3.4 POWER PERMIT CONDITIONS

The power permit conditions for the various entities are defined as follows.

1. Entity DQQDL

   ST_PWR_PERM = (|U_QS0|<20 mV) AND (ST_NQD0=1) AND (ST_COHER_OK=1) AND (ST_PWR=1)

2. Entity DQAMC

   ST_PWR_PERM = (ST_COM=1) AND (ST_BUS=1) AND (ST_TIMING=1)

3. Entity MB

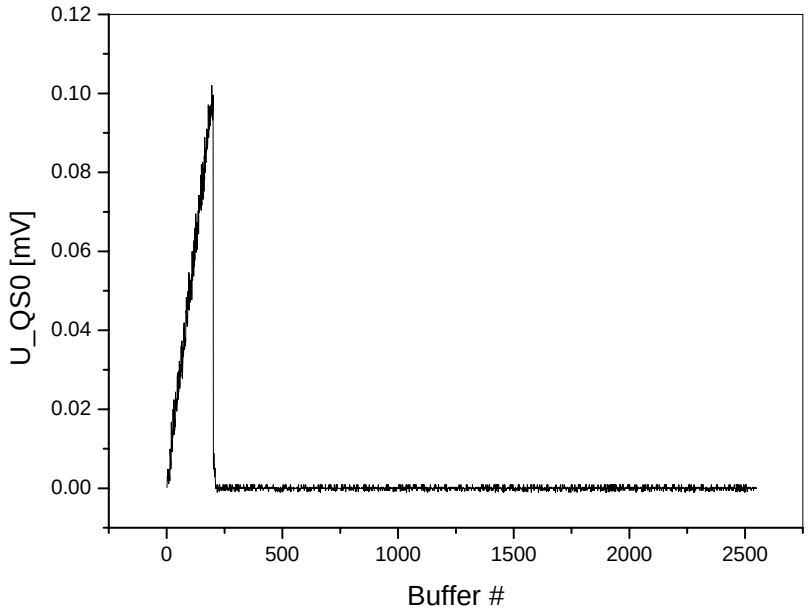
   ST_PWR_PERM = ST_MAGNET_OK AND ST_PWR_PERM(DQQDL) AND ST_PWR_PERM(DQAMC) AND (U_HDS_1>810V) AND (U_HDS_2>810V) AND (U_HDS_3>810V) AND (U_HDS_4>810V)

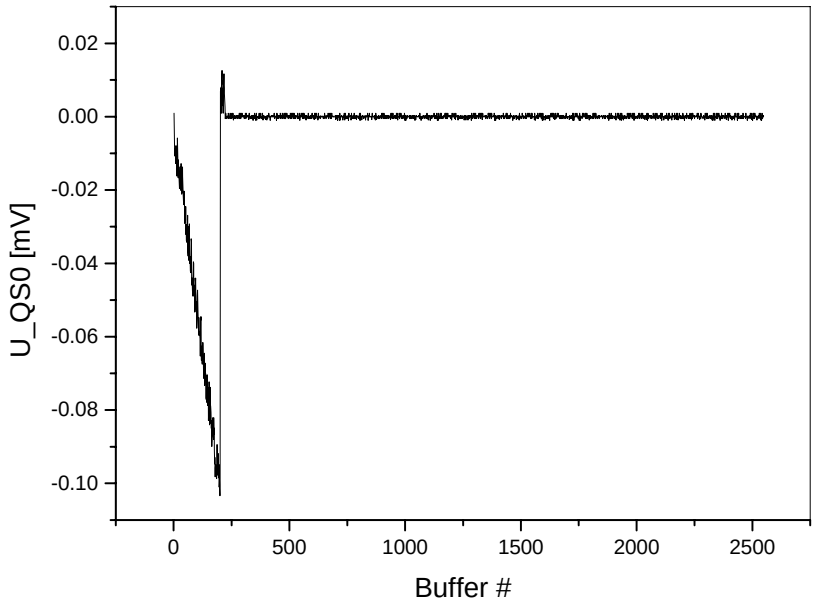
In addition to the above mentioned conditions it is mandatory that the DQAMC status is <Sending logging data> or <Filling buffer>.

The verification of the power permit conditions on the gateway level requires to read data from all redundant boards.

12.3.5 TEST_MODE

In case an <ENTER_TEST_MODE_EQ0> command has been successfully issued the corresponding local quench detection board will create a pre-defined test signal with the help of its on board DAC. This signal allows the verification of the detection threshold of the analog detection circuit. Depending whether a < PREPARE_POSITIVE_TEST_MODE> or < PREPARE_NEGATIVE_TEST_MODE> command has been sent the positive or negative polarity will be tested.




0.02
0.00
-0.02
-0.04
-0.06
-0.08
-0.10
U_QS0 [mV]
0
500
1000
1500
2000
2500
Buffer #

12.3.6 DATA SEGMENT DQAMC/DQAMG/DQAMS IN CASE OF SEND_NAME COMMAND

In case a <SEND_NAME> command has been issued by the DQGTW the DQAMC will send its functional name in the LHC, the controller type and its WorldFip subscriber number using a modified structure **dataqps_wf_struct** and the corresponding definitions. Based on the functional position the corresponding LHC part identification numbers can be found in the respective QPS databases.

```
struct dataqps_wf_struct{
      unsigned char status_command_gtw;
      unsigned char sub_buffer; // WorldFip subscriber number
      unsigned char name_value[22]; // 22 bytes
};
```

Table 36: Coding of the functional position.

name_value[i]	*Content*	*Remark*
0	Not used	Always 0
1	Controller type	
2	Sector	Only for DQAMC type controllers[5]
3	Half cell	Only for DQAMC type controllers
4	Controller in half cell	Only for DQAMC type controllers
5	Underground area	Only for DQAMG/DQAMS type controllers
6	Controller in underground area	Only for DQAMG/DQAMS type controllers
7	Selected heater firing	Only for DQAMC type controllers
8	Version	Firmware DQAMC: Version.Revision
9	Revision	Firmware DQAMC: Version.Revision
10	TIME_STAMP_OFFSET	Only for DQAMC type MB controllers
11	TIME_STAMP_OFFSET_EXT	Only for DQAMC type MQ controllers
12	TIME_STAMP_OFFSET_INT	Only for DQAMC type MQ controllers
13	Version	Firmware DQQDL: Version.Revision
14	Revision	Firmware DQQDL: Version.Revision
15 … 21	Not used yet	Always 0

[5] The name refers to the entity “DQAMC” not to the name found in the installation plans.

name_value[1]	***Controller type***
0	DQAMC type MB
1	DQAMC type MQ
2	DQAMG type A
3	DQAMG type B
4	DQAMG type C
5	DQAMG type D
6	DQAMG type E
7	DQAMG type F
8	DQAMS type A
9	DQAMS type B

name_value[2]	***Sector***
0	L1
1	R1
2	L2
3	R2
4	L3
5	R3
6	L4
7	R4
8	L5
9	R5
10	L6
11	R6
12	L7
13	R7
14	L8
15	R8

name_value[3]	***Cell***
8 ... 34	8 ... 34

name_value[4]	***Rack in tunnel***
0 ... 4	A ... E

name_value[5]	*Underground area*
0	RR13
1	RR17
2	UJ14
3	UJ16
4	RE18
5	RE22
6	UA23
7	UA27
8	RE28
9	RE32
10	UJ33
11	RE38
12	RE42
13	UA43
14	UA47
15	RE48
16	RE52
17	UJ56
18	RR53
19	RR57
20	USC55
21	RE58
22	RE62
23	UA63
24	UA67
25	RE68
26	RE72
27	RR73
28	RR77
29	RE78
30	RE82
31	UA83
32	UA87
33	RE88
34	RE12

name_value[6]	***Controller***
0 ... 18	A ... S

name_value[7]	***Selected heater firing***
0	Disabled
1	Enabled & triggered from an odd point
2	Enabled & triggered from an even point

A DQLPU type MQ can only be triggered from an even point.

12.3.6.1 EXAMPLES

A DQAMC type MB controller may send the following data coded in **name_value[22]**:

name_value[i]	***Value***
0	0
1	0
2	14
3	34
4	2
5	0
6	0
7	1
8	5
9	4
10	3
11	0
12	0
13	3
14	7

The functional name can be decoded as DQAMC.A34L8, DQAMC firmware 5.4 and DQQDL firmware 3.7 is running and the timestamp offset is 3 ms.

A DQAMG type B controller may send the following data coded in **name_value[22]**:

Name_value[i]	***Value***
0	0
1	3
2	0
3	0
4	0
5	14

6	2
7	0
8	5
9	4
10	0
11	2
12	3
13	3
14	7

The functional name can be decoded as DQAMG.C.UA47, DQAMC firmware 5.4 and DQQDL firmware 3.7 is running and the timestamp offsets are 2 and 3 ms.

13. DQAMC – INTERNAL COMMUNICATION

This chapter describes the internal communication between the DQAMC controller and its associated equipment, namely the Local Quench Detectors DQQDL.

13.1 LOGGING BUFFER DEFINITION TYPE MB

Ibuf[i]	***Content type A***							
1	U_1 8LSB							
0	U_2 4MSB				U_1 4MSB			
3	U_2 8LSB							
2	U_QS_0 8LSB							
5	U_HDS_1 4MSB				UQS_0 4MSB			
4	U_HDS_1 8LSB							
7	U_HDS_2 8LSB							
6	U_HDS_3 4MSB				U_HDS_2 4MSB			
9	U_HDS_3 8LSB							
8	U_HDS_4 8LSB							
11	U_HDS_4 4MSB							
10	ST2	ST1	ST0	ST_NQD0	MAGNET_OK	ST_COHER	ST_PWR_PERM	ST_PWR
12	N/A							
13	N/A							
14	N/A							
15	N/A							

13.2 LOGGING BUFFER DEFINITION TYPE B

Lbuf[i]	***Content type B***							
1	U1_1 8LSB							
0	U2_1 4MSB				U1_1 4MSB			
3	U2_1 8LSB							
2	U_QS0_1 8LSB							
5					U_QS0_1 4MSB			
4	U1_2 8LSB							
7	U2_2 4MSB				U1_2 4MSB			
6	U2_2 8LSB							
9	U_QS0_2 8LSB							
8					U_QS0_2 4MSB			
11	U_HDS1 8LSB							
10	U_HDS2 4MSB				U_HDS1 4MSB			
13	U_HDS2 8LSB							
12	ST2[6]	ST1	ST0	NQD0	MAGNET_OK	COHER_OK	PWR_PERM	PWR_OK
15	ST2[7]	ST1	ST0	NQD0	MAGNET_OK	COHER_OK	PWR_PERM	PWR_OK
14	N/A							

[6] DQQDL #1
[7] DQQDL #2

13.3 DQAMC - DQQDL COMMUNICATION PROTOCOL

The DQAMC controller communicates with the associated DQQDL type quench detectors via a galvanically isolated SPI link. The DQQDL firmware is independent of the DQLPU type and unique for all devices.

13.3.1 BASIC COMMUNICATION PARAMETERS

The tables show the settings of the SPICON register of the ADuC8XX device.

	7	***6***	***5***	***4***	***3***	***2***	***1***	***0***	***HEX***
SPICON	ISPI	WCOL	SPE	SPIM	CPOL	CPHA	SPR1	SPR0	
DQQDL	0	0	1	0	0	1	0	0	24
DQAMC	0	0	1	1	0	1	1	1	37

Client	***Name***	***SS of client wired to port***	***MB***	***MQ***
1	DQQDL_1A	P3.4	MB	MQ external aperture
2	DQQDL_1B	P3.5	MB	MQ external aperture
3	DQQDL_2A	P3.0	N/A	MQ internal aperture
4	DQQDL_2B	P3.1	N/A	MQ internal aperture

In case of the DQAMC type MQ controller no serial cable must be connected during exploitation of the device as P3.0 and P3.1 are shared with the serial port.

13.4 DQAMC → DQQDL COMMANDS

No.	Command	Description	DQQDL status
0	Send logging buffer 5	Status	No change
1	Send logging buffer 0	Logging data	7
2	Send logging buffer 1		7
3	Send logging buffer 2		7
4	Send logging buffer 3		7
5	Send logging buffer 4		7
6	Send logging buffer 5		7
7	Reserved		No change
8	Reserved		No change
9	Enter positive test mode	Test mode	
10	Enter negative test mode		
11	Zero calibration ON	Analog input channel calibration	
12	Zero calibration OFF		
13	Reset	Reset	→0
14	Send temperature buffer 0		
15	Send temperature buffer 1		
16	Send temperature buffer 2		
17	Send temperature buffer 3		
18	Send temperature buffer 4		
19	Send temperature buffer 5		
14→254	Reserved		
255	Push data / new command		No change
>255	Send buffer		6

No.	Command	Description	DQQDL status

0	Send status	Status data	No change
1	Send logging buffer	Logging data	7
2	Send pattern	Test pattern	3
3	Send temperature	Send temperature (ADuC812/831)	8
4	Zero calibration	Calibration	9
5	Send coefficients		10
6	Reserved	N/A	No change
7	Reserved	N/A	No change
8	Reserved	N/A	No change
9	Enter positive test mode	Positive test mode	4
10	Enter negative test mode	Negative test mode	5
11	Last block		6→3
12	Reserved	N/A	No change
13	Reset	Reset	0
14→254	Reserved	N/A	No change
255	Push data / new command	Push one byte to SPI buffer.	No change
>255	Send buffer		6

7	***6***	***5***	***4***	***3***	***2***	***1***	***0***	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
m	m	m	N	n	n	n	n	n	n	N	n	n	n	n	0
Buffer i 1<i<6			Number of post mortem buffer j 0<j<2549												0

13.4.1 DQQDL DEVICE STATUS

Value	***Device status***
0	RUNNING
1	FILLING BUFFER
2	BUFFER READY
3	SENDING TEST PATTERN
4	POSITIVE TEST
5	NEGATIVE TEST
6	SENDING BUFFER
7	LOGGING
8	TEMPERATURE
9	CALIBRATING
10	SENDING COEFFICIENTS
11	RESERVED
12	RESERVED
13	RESERVED
14	RESERVED
15	RESERVED

13.5 DQAMC → DQQDL COMMUNICATION SEQUENCE

		1st DQAMC cycle	***2nd DQAMC cycle***	***3rd DQAMC cycle***	***Next cycle***
DQAMC	***Sending***	0xFF	Low byte of command	High byte of command	0xFF
	Receiving	1 byte of data	N/A	N/A	1 byte of data
DQQDL	***Sending***	1 byte of data	N/A	N/A	1 byte of data
	Receiving	0xFF	Low byte of command	High byte of command	0xFF

13.6 SYNCHRONIZATION

The DQAMC controller firmware is responsible for time stamping. The trigger of any post-mortem event is polled with a sampling frequency of 1 kHz.

13.7 DQQDL LOGGING BUFFER

Ibuf[i]	***Content DQQDL***	
0	U1_1 8LSB	
1	U2_1 4MSB	U1_1 4MSB

2	U2_1 8LSB							
3	U_QS0_1 8LSB							
4					U_QS0_1 4MSB			
5	ST2	ST1	ST0	NQD0	MAGNET_OK	COHER_OK	PWR_PERM	PWR_OK
6	N/A							
7	N/A							

13.8 DQQDL TEST MODE

The test mode signal is generated with the help of the two on-board DACs of the DQQDL ADuC812 controller. In order to route this signal to the analog input stage of the quench detector the DQAMC controller must close the corresponding relay and issue the <enter test mode> command. If test mode is enabled the signal is directly connected to the instrumentation amplifiers of the DQQDL.

High byte						Lowbyte		Command	

High byte									Low byte									Command
8	7	6	5	4	3	2	1	0	1	0	7	6	5	4	3	2	1	

14. DATA ANALYSIS DQAMC TYPE MB AND MQ

14.1 ANALYSIS OF POST MORTEM BLOCKS DQAMC TYPE MB

The first step of the data analysis is to check the integrity of the buffer content.

14.2 DATA SEGMENT DQAMG TYPE A

All data produced by the DQAMG type A are organised in a structure named **dataqps_wf_struct**. The total size of the structure is 24 Bytes and corresponds to the size of one of the produced variables DATA0 to DATA3. The structure contains the data for one circuit supervised by a global protection unit DQGPU type A. A DQGPU type A can protect a maximum of 4 circuits. In case of a post mortem event in any of the supervised circuits the acquired data will be transmitted within the same structure and the other circuits will continue transmitting logging data.

Circuit	*Data*
1	DATA0
2	DATA1
3	DATA2
4	DATA3

```
struct dataqps_wf_struct{
    unsigned char status_command_gtw;
    unsigned int number_of_buffer;// number of post mortem block 0..65535
    unsigned long acquisition_time_sec;// seconds after 1/1/1970
    unsigned int acquisition_time_millisec;// ms
    unsigned char dqamg_status;// status bits of DQAMG
    unsigned char dqqdc_status;// status bits of DQQDC
    unsigned char dqqdg_status;// status bits of DQQDG
    unsigned char analog_value[12];// analog values DQQDC and DQQDG
};
```

The coding of the <status_command_gtw> variable is as in table 9. A post mortem event in any of the associated quench detectors creates a hardware interrupt on the DQAMG circuit board. This interrupt will be used to timestamp the post mortem event.

Table X. Definition of the status variables for DQAMG type A.

dqamg_status								
Bit	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
Content	N/A	FIP_OK	LOC_BUS_OK	TIMING_OK	COM_LOST	PWR_PERM_CALC	FAN0_OK	FAN1_OK

dqqdc_status								
Bit	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
Content	HTS_RESOL	LEAD_OK	COHER_OK	PWR_PERM	HTS_RESOL	LEAD_OK	COHER_OK	PWR_PERM
Device	LD1				LD2			

dqqdg_status								
Bit	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
Content	N/A	N/A	N/A	N/A	CIRCUIT_OK	COHER_OK	PWR_OK	PWR_PERM

Table X. Definition of the variable **analog_value** for DQAMG type A.

	High byte		***Low byte***	
Byte	0D		0C	
analog_value[0]	U_HTS1 4MSB	U_RES1 4MSB	U_RES1 8LSB	
Byte	0F		0E	
analog_value[1]	U_RES2 8LSB		U_HTS1 8LSB	
Byte	11		10	
analog_value[2]	U_HTS2 8LSB		U_HTS2 4MSB	U_RES2 4MSB
Byte	13		12	
analog_value[3]	I_DCCT 4MSB	U_DIFF 4MSB	U_DIFF 8LSB	
Byte	15		14	
analog_value[4]	I_DIDT 8LSB		I_DCCT 8LSB	
Byte	17		16	
analog_value[5]	U_RES 8LSB		U_RES 4MSB	I_DIDT 4MSB

Remarks:

- If bit 7 in dqqdc_status is set, the corresponding DQQDC will not transmit UResn but send UHTSn with full 24 Bit resolution. In that case decoding is as follows:

HTS_RESOL=1	U_HTSn High byte	U_HTSn Middle byte		U_HTSn Low byte
HTS_RESOL=0	U_HTSn 8LSB	U_HTSn 4MSB	U_RESn 4MSB	U_RESn 8LSB

- Lcirc must be calculated by the gateway
- GainInput = obsolete
- PWR_OK allocated to dqqdg_status but read directly by DQAMG
- FANi_OK allocated and read directly by DQAMG

14.3 DQAMG TYPE A INTERNAL COMMUNICATION

14.3.1 I2C BUS LINK

All DQAMG type controllers use an I2C bus link for the communication with the associated equipment.

Table X. Definition of I2C bus addresses for DQGPU type A.

Equipment	*Device*	*Address*	*Write*	*Read*
1	DQQDG CIRCUIT #1 BOARD A	10h	20h	21h
	DQQDG CIRCUIT #1 BOARD B	11h	22h	23h
2	DQQDC CIRCUIT #1 LEAD 1 A	12h	24h	25h
	DQQDC CIRCUIT #1 LEAD 1 B	13h	26h	27h
3	DQQDC CIRCUIT #1 LEAD 2 A	14h	28h	29h
	DQQDC CIRCUIT #1 LEAD 2 B	15h	2Ah	2Bh
4	DQQDG CIRCUIT #2 BOARD A	16h	2Ch	2Dh
	DQQDG CIRCUIT #2 BOARD B	17h	2Eh	2Fh
5	DQQDC CIRCUIT #2 LEAD 1 A	18h	30h	31h
	DQQDC CIRCUIT #2 LEAD 1 B	19h	32h	33h
6	DQQDC CIRCUIT #2 LEAD 2 A	1Ah	34h	35h
	DQQDC CIRCUIT #2 LEAD 2 B	1Bh	36h	37h
7	DQQDG CIRCUIT #3 BOARD A	1Ch	38h	39h
	DQQDG CIRCUIT #3 BOARD B	1Dh	3Ah	3Bh
8	DQQDC CIRCUIT #3 LEAD 1 A	1Eh	3Ch	3Dh
	DQQDC CIRCUIT #3 LEAD 1 B	1Fh	3Eh	3Fh
9	DQQDC CIRCUIT #3 LEAD 2 A	20h	40h	41h
	DQQDC CIRCUIT #3 LEAD 2 B	21h	42h	43h
10	DQQDG CIRCUIT #4 BOARD A	22h	44h	45h
	DQQDG CIRCUIT #4 BOARD B	23h	46h	47h
11	DQQDC CIRCUIT #4 LEAD 1 A	24h	48h	49h
	DQQDC CIRCUIT #4 LEAD 1 B	25h	4Ah	4Bh
12	DQQDC CIRCUIT #4 LEAD 2 A	26h	4Ch	4Dh
	DQQDC CIRCUIT #4 LEAD 2 B	27h	4Eh	4Fh

Table X. Definition of I2C bus addresses for DQGPU type B.

Equipment	*Device*	*Address*	*Write*	*Read*
1	DQQDG CIRCUIT #1 BOARD A	10h	20h	21h
	DQQDG CIRCUIT #1 BOARD B	11h	22h	23h
2	DQQDG CIRCUIT #1 BOARD C	12h	24h	25h
	DQQDG CIRCUIT #1 BOARD D	13h	26h	27h
3	DQQDC CIRCUIT #1 LEAD 1 A	14h	28h	29h
	DQQDC CIRCUIT #1 LEAD 1 B	15h	2Ah	2Bh
4	DQQDC CIRCUIT #1 LEAD 2 A	16h	2Ch	2Dh
	DQQDC CIRCUIT #1 LEAD 2 B	17h	2Eh	2Fh
5	DQQDC CIRCUIT #1 LEAD 3 A	18h	30h	31h
	DQQDC CIRCUIT #1 LEAD 3 B	19h	32h	33h
6	DQQDG CIRCUIT #2 BOARD A	1Ah	34h	35h
	DQQDG CIRCUIT #2 BOARD B	1Bh	36h	37h
7	DQQDG CIRCUIT #2 BOARD C	1Ch	38h	39h
	DQQDG CIRCUIT #2 BOARD D	1Dh	3Ah	3Bh
8	DQQDC CIRCUIT #2 LEAD 1 A	1Eh	3Ch	3Dh
	DQQDC CIRCUIT #2 LEAD 1 B	1Fh	3Eh	3Fh
9	DQQDC CIRCUIT #2 LEAD 2 A	20h	40h	41h
	DQQDC CIRCUIT #2 LEAD 2 B	21h	42h	43h
10	DQQDC CIRCUIT #2 LEAD 3 A	22h	44h	45h
	DQQDC CIRCUIT #2 LEAD 3 B	23h	46h	47h

Byte	7	6	5	4	3	2	1	0
Content	I2C address	Status	UHTS_H	UHTS_M	UHTS_L	URES_H	URES_M	URES_L

Bit	7	6	5	4	3	2	1	0
Content	N/A	N/A	N/A	N/A	HTS_RESOL	LEAD_OK	COHER_OK	PWR_PERM

14.3.2 POST MORTEM BUFFERS

Device	Start	End	Sampling frequency	Length	Bytes
DQQDG	-5.0 s	+5.0 s	100 Hz	1000	7000
DQQDC	-5.0 s	+5.0 s	10 Hz	100	

14.3.3 DQAMG → DQQDG COMUNICATION SEQUENCE

The DQAMG sends a command of 2 bytes length (I2C write mode) and the DQQDG answers by sending a data block of 7 bytes length (I2C read mode).

14.3.4 DQAMG → DQQDG COMMANDS

No.	Command	Description	DQQDG status
0	SEND_STATUS	Status	RUNNING
1	SEND_LOGGING	Logging data	LOGGING
2	Reserved	N/A	N/A
3	SEND_PATTERN	Test pattern	SENDING TEST PATTERN
4	ZERO_CALIBRATE	Zero input calibration	CALIBRATING
5	SEND_COEFFICIENTS	Calibration coefficients	SENDING COEFFICIENTS
6	Reserved	N/A	N/A
7	Reserved	N/A	N/A
8	Reserved	N/A	N/A
9	ENTER_POSITIVE_TEST_MODE		POSITIVE TEST MODE
10	ENTER_NEGATIVE_TEST_MODE		NEGATIVE TEST MODE
11	Reserved	N/A	N/A
12	Reserved	N/A	N/A
13	RESET	Reset	
14→254	Reserved	N/A	N/A
255	Reserved	N/A	N/A
256	SEND_BUFFER #0	Post mortem	SENDING BUFFER

257 … 1254	SEND_BUFFER #(n-256)	Post mortem	SENDING BUFFER
1255	SEND_BUFFER #999	Post mortem	SENDING BUFFER

14.3.5 DQQDG → DQAMG DATA BLOCK

Byte	***Content***		***Applicable device status***
0	U_DIFF 8LSB		LOGGING
1	I_DCCT 4MSB	U_DIFF 4MSB	SENDING BUFFER
2	I_DCCT 8LSB		POSITIVE TEST MODE
3	I_DIDT 8LSB		NEGATIVE TEST MODE
4	U_RES 4MSB	I_DIDT 4MSB	
5	U_RES 8LSB		
6	DQQDG status		ALL

In case a <SEND PATTERN> command is issued the analog values will be replaced by fixed 12 Bit integer values corresponding to the following physical values:

Signal	***Physical value***	***Integer value***
U_DIFF	150 mV	
I_DCCT	300 A	
I_DIDT	10 A/s	
U_RES	50 mV	

The status byte 6 is coded as follows:

Bit	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
Content	Device status				CMDCHK	CIRCUIT_OK	COHER_OK	PWR_PERM

CMDCHK is HIGH if command has been accepted otherwise low.

14.3.6 DQQDG DEVICE STATUS

Value	***Device status***
0	RUNNING
1	FILLING BUFFER
2	BUFFER READY
3	SENDING TEST PATTERN
4	POSITIVE TEST
5	NEGATIVE TEST
6	SENDING BUFFER
7	LOGGING
8	CALIBRATING
9	SENDING COEFFICIENTS
10	RESERVED
11	RESERVED
12	RESERVED
13	RESERVED
14	RESERVED
15	RESERVED

14.3.7 DQQDI → DQAMG DATA BLOCK

14.3.8 DQQDT → DQAMG DATA BLOCK

14.3.9 DQQDC → DQAMG DATA BLOCK

Byte	***Content***		***Applicable device status***
0	U_RES_LD1 8LSB		LOGGING
1	U_HTS_LD1 4MSB	U_RES_LD1 4MSB	SENDING BUFFER
2	UHTS_LD 8LSB		POSITIVE TEST MODE
3	U_RES_LD1 8LSB		NEGATIVE TEST MODE
4	U_HTS_LD1 4MSB	U_RES_LD1 4MSB	
5	UHTS_LD 8LSB		
6	DQQDC status (LD1 & LD2)		ALL

14.4 THE….

15. FIRMWARE VERSIONS

Table X. DQAMC firmware versions.

Version	*Description*	*Changes to prior versions & bug-fixes*
<3.0	R&D versions referring to the old communication protocol	
3.0	1st working version implementing latest protocol	Implementation of the new communication protocol. Proper synchronisation to the fieldbus. DQAMC requires a few macro-cycle to synchronize to the external interrupt created by the global TIME variable.
3.1		Start-up value for DQAMC status set to zero. Wrong UHDSn encoding in logging mode fixed. Timeout value for test-mode implemented. In case no trigger signal is received from the associated DQQDL, the DQAMC will switch after 300 macro-cycles (= 60 sec) to Status 1 (filling buffer).
3.2		Bug in coding of UHDS1 and UHDS3 fixed and verified. SPI communication revised and several bugs fixed. Send name command now correctly treated.
3.3		
3.4		DQQDL status transmission corrected. DQAMC sends buffer number in case of filling buffer status. DQQDL sync optimised.
3.8	Version used for demo 07-May-2004	
4.1	1st version implementing new command structure	
4.2	1st version implementing type B code	

Table X. DQQDL firmware versions.

Version	*Description*	*Changes to prior versions & bug-fixes*
1.2	1st working version	
1.3		SPI communication revised. Transmission of real ADC data enabled.
1.4		Test mode corrected.
1.5		Timer 1 bug fixed. DQQDL now running @ 200Hz. External interrupt in case of quench.
1.9	Version used for demo 07-May-2004	
2.1	1st version implementing new command structure	

16. REFERENCES

[1] Analog Devices MicroConverter® Technical Note – uC020