DAΦNE TECHNICAL NOTE
INFN - LNF, Accelerator Division

Frascati, 09/05/2014
Note: **C-22**

New DAQ System for the DAΦNE Transfer Lines Charge Monitors

A. Stecchi, O. Coiro, L.G. Foggetta, F. Galletti, A. Ghigo, A. Michelotti,
D. Pellegrini, M. Serio, A. Stella

Typographical Conventions

Where a term is enclosed in brackets " < > ", that term is a placeholder for an alphanumeric string or a variable name
e.g. <element_name> is a placeholder for the string BCMTM001

Where two or more terms are enclosed in braces and separated by a pipe " { a | b | c } ", those terms are alternative one another
e.g. {ON|OFF} could be ON or OFF

Introduction

The system described in this note acquires and processes the raw signals provided by the toroidal charge monitors installed along the DAΦNE transfer lines.

Electron or positron pulse charges are acquired at the beam rate at each pickup, according to the different DAΦNE timing states[1] and displayed on the control system consoles. A dedicated storing process allows offline analysis, to provide evaluation of the injection efficiencies useful for machine tuning and the charge integrals over time required to monitor radiation safety.

1. System Setup

Nine integrating current transformer (ICTs), equipped with wall current bypass and electromagnetic shields, are installed over ceramic gaps in the vacuum chamber along the transfer lines which connect the Linac to the Accumulator and to the Main Rings[2], as reported in Fig. 1.

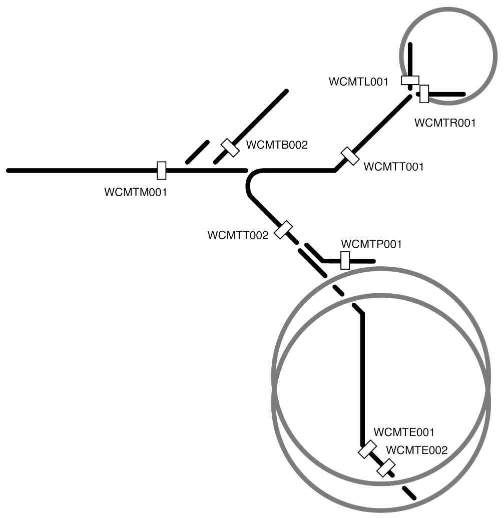


Figure 1: Locations of the BCM monitors along the DAΦNE Transfer Lines.

Each ICT is capable of integrating the fast rise-time pulse induced by the Linac bunched beam with a 20ns time constant and is connected through coaxial cables to an array of BCM detection board electronics[3] whose pseudo-DC outputs, proportional to the pulse charge, are sampled by VME ADCs.

The software code described in the following chapters controls the digital I/O channels providing to each BCM board the TTL levels which select the gain settings and enable/disable the calibration mode.

Figure 2 reports a schematic layout of the toroidal charge monitor system installed in the DAΦNE transfer lines.

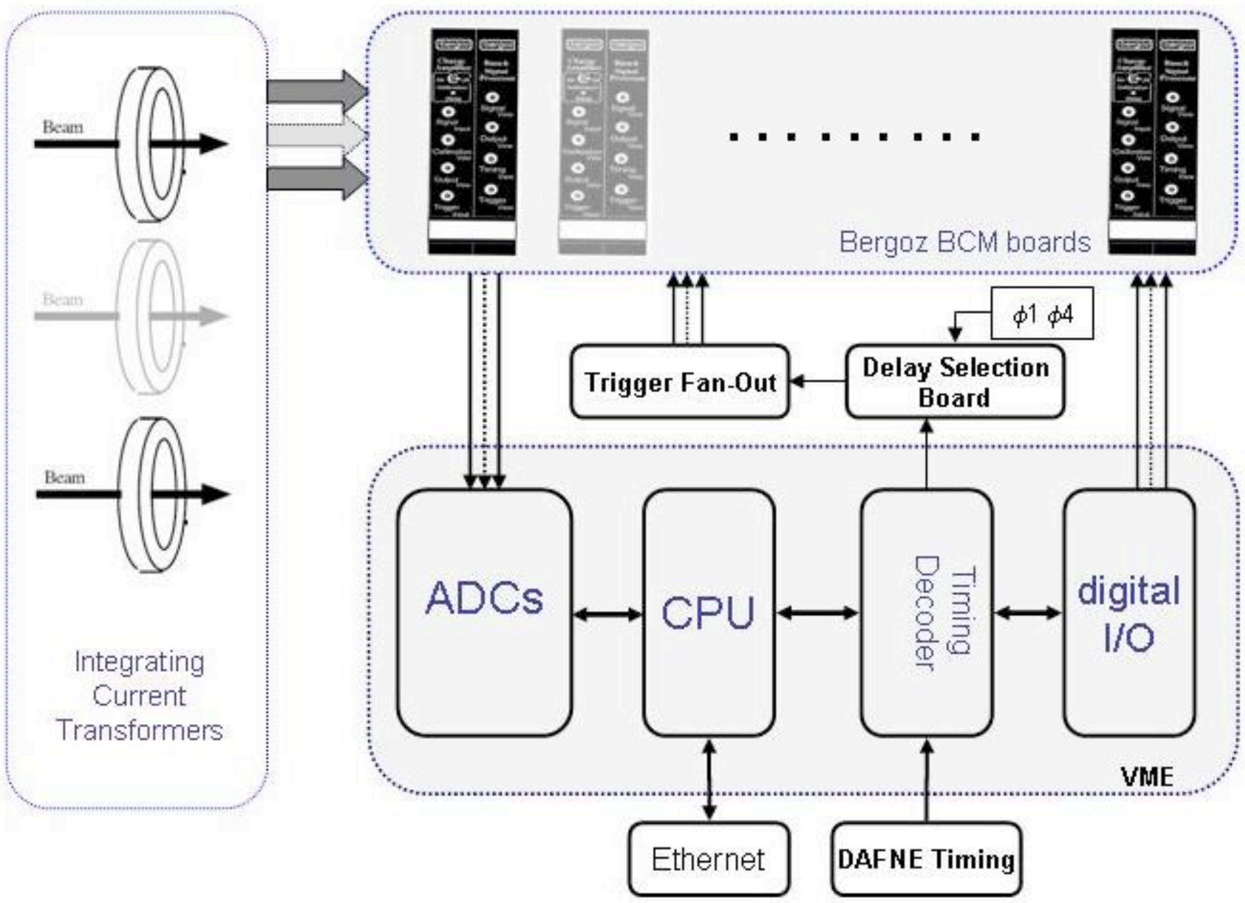


Figure 2: Schematic Layout of the Data Acquisition System.

According to the current timing state associated with the transport of e+/e- pulses along different zones of the machine:

- LSP Linac to Spectrometer
- LBT Linac to BTF
- LTA Linac to Accumulator
- AMR Accumulator to Main Rings

the beam crosses different monitor sequences as reported in Table 1: for each monitor it is indicated the relative Bergoz BCM module and the ADC channel. The last two columns indicate, both for positron and electron modes, the relevant timing state.

Table 1: BCM monitors used in the DAQ System.

Monitor	Bergoz Module CAC + BSP [A-J]	ADC Channel [0-15]	Positron mode	Electron mode
BCM TM 001	A	08	LSP, LBT, LTA	LSP, LBT, LTA
BCM TE 002	B	07	-	AMR
BCM TB 002	C	06	LBT	LBT
BCM TT 001	D	05	LTA, AMR	LTA, AMR
BCM TR 001	E	04	LTA	AMR
BCM TL 001	F	03	AMR	LTA
BCM TT 002	G	02	AMR	AMR
BCM TP 001	H	01	AMR	-
BCM TE 001	J	00	-	AMR

3. Code description

In the DAΦNE Control System each hardware device is represented by a *class*. Different device typologies correspond to different classes.

A class consists of a data structure that contains all the physical quantities that are relevant to the device control processes and five standard methods for the device management (Table 2).

Table 2: The five standard class methods. Each class expresses Instantiations of theses methods, characterized by a prefix specific for that class (e.g. for the class BCM, *BCMClose* is the instantiation of the *close* method).

Methods	Actions
loadRTDB	loads the global variables with the data taken from the configuration files
initHW	performs the hardware initialization at program start-up
ctrl	performs the continuous control of the device
cmd	executes the commands issued to the device
close	executes the closing tasks at program stop

The DAΦNE charge DAQ system employs two classes:

- BCM (Beam Charge Monitor)
  which is related to the i-th monitor as well as its associated signal conditioning front-end module. The BCM class provides methods for setting the individual gain, the inverted/not_inverted and calibrating/not_calibrating operating conditions.

- WCM (Wall Current Monitor)
  which is related to the real-time acquisition hardware along with the fast data storage system.

The instantiations of the methods for the BCM and WCM classes, are used to implement a DEVIL - that is a standard third-level application - running on the VME bus controller.

The Charge Transport Monitor system has to be deterministic for what concerns the acquisition and memorization of the beam charge value at a given time. This means that the acquisition of the beam charge - as well as all the related processing and storage tasks - has to occur synchronously with the machine timing signals (see §3.1).

This has been obtained by splitting the code into a RTA (Real-Time Acquisition) task and a SC (Slow Control) task, communicating to each other through the internal processor RAM (see §3.1 and §3.2).

3.1 Real Time Acquisition task

The RTA task performs the data acquisition and the fast storage of live data.

3.1.1 Algorithm

The DAΦNE timing distribution system broadcasts at 50 Hz, over a serial bus, a TSW (Timing State Word) that describes the incoming machine state. Besides the TSW, the timing system distributes also four marker signals (φ_1, φ_2, φ_3, φ_4) delayed by 90° (5 ms): the TSW carries the information of the machine state and the φ_n signals are used to trigger the devices interested by the incoming timing state.

The TSW is dispatched with the φ_1 trigger and describes the machine state that has to be considered as valid at the following φ_4 trigger. In the *Charge Transport Monitor, the* φ_1 trigger is used to strobe the ADC_1 (calibration ADC) whilst the φ_4 trigger is used to strobe the ADC_4 (acquisition ADC). This means that it is important to acquire the TSW at the φ_1 pulse immediately precedent the φ_4 pulse, in order to associate the proper machine state to the sampled values.

This is guaranteed by taking into account the STROBE_OVERRUN information available from the SCSR registers of the ADCs. The STROBE_OVERRUN is different from zero if more than a single strobe has occurred since the last ADC readout.

The RTA is performed by a C program that executes the deterministic acquisition of the ADCs sampled values and of the TSW, the TSW decodification and the fast storage of all these data into the processor RAM.

At first, the RTA creates two lookup tables (see Table 3): one for the positron mode (mode = 0) and one for the electron mode (mode = 1). Each lookup table has a number of lines equal to the number of possible machine states (0=LSP, 1=LBT, 2=LTA, 3=AMR / LSP+AMR) and a number of columns equal to the number of monitors plus 1. The first component of each line, counts the number of following meaningful components (that is how many monitors have to be taken into account for the machine state corresponding to that line).

The following components contain the ADC channel numbers which the monitor outputs (relevant for the given machine state) are connected to.

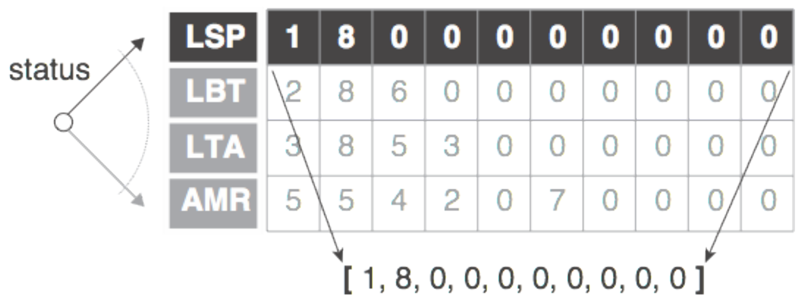
Table 3: Lookup tables *chSel_p* for the positron mode (left) and *chSel_e* for the electron mode (right).

LSP	**1**	8	0	0	0	0	0	0	0	0
LBT	**2**	8	6	0	0	0	0	0	0	0
LTA	**3**	8	5	4	0	0	0	0	0	0
AMR	**4**	3	5	2	1	0	0	0	0	0

LSP	**1**	8	0	0	0	0	0	0	0	0
LBT	**2**	8	6	0	0	0	0	0	0	0
LTA	**3**	8	5	3	0	0	0	0	0	0
AMR	**5**	4	5	2	0	7	0	0	0	0

After the creation of the lookup tables, the code calls repeatedly the function *WCMAcquire* that is dedicated to the acquisition of the ADCs and of the TSW.

The TSW register content is used to work out the current *mode* and *state*; the *mode* value is used to choose the proper lookup table (see Table 3) whilst the *state* value is used to select the proper line from the chosen lookup table (see Fig. 3).



LSP	1	8	0	0	0	0	0	0	0	0
LBT	2	8	6	0	0	0	0	0	0	0
LTA	3	8	5	3	0	0	0	0	0	0
AMR	5	5	4	2	0	7	0	0	0	0

Figure 3: Extraction of the line corresponding to the LSP state from the chSel_e lookup table. The state value is used to select the proper line that is then loaded into a 1-D array.

Dealing with no interrupt mechanism, the determinism is obtained by mean of the following polling sequence (see Fig. 4):

(A - C) READ ADC$_1$
- keep polling the ADC$_1$ register that indicates if the φ_1 strobe has occurred
- acquire the ADC$_1$ values

(C - D) READ TSW
- the TSW is related to *some previous* φ_1 pulse

(D - F) READ ADC$_4$
- keep polling the ADC$_4$ register that indicates if the φ_4 strobe has occurred
- if the STROBE_OVERRUN of ADC$_4$ is 0, then
  - just a single φ_4 pulse has occurred since the previous ADC$_4$ readout
  - the TSW is related to the *immediately previous* φ_1 pulse

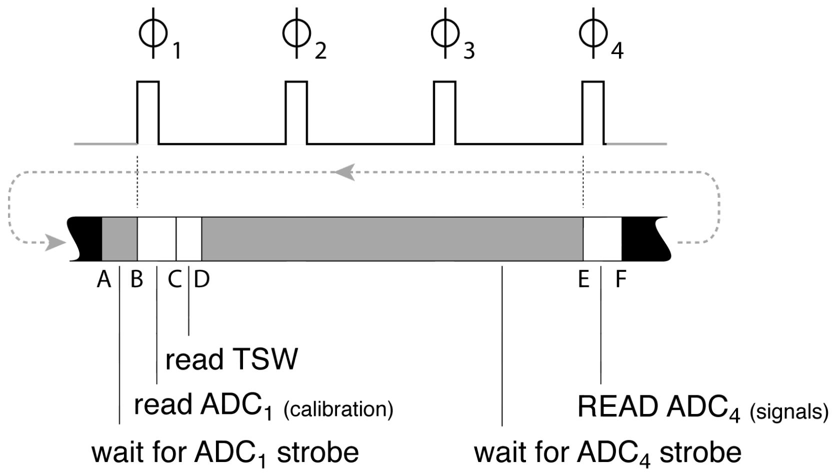


Figure 4: The D-E and A-B gaps compensate the absence of synchronism between the routine execution and the φ_n pulses, as well as any latency due to the Linux OS background tasks. All the conversion and storage functions executes during the F-A lap.

3.1.2 Data Storage

The acquired values are stored in two different ways: histogram memory arrays and timed circular buffers, both residing on RAM.

In the histogram memory arrays, each component corresponds to a monitor and contains the growing sum of its acquired charge values [nC] whilst in the circular buffers each line contains the last acquired charge values [nC] of all the monitors.

In both cases they are used fixed size storage array, with the BCM channels always in the same position.

For the histogram memory arrays the storage format is:

c0	c1	c2	c3	c4	c5	c6	c7	c8

There are dedicated histograms for each machine state and mode so that there is no need storing mode or state information besides data. The number of histograms is N*2, where N is the number of machine states and the factor 2 is due to the possible machine modes.

In histograms, only the values of the ADC channels meaningful to the current mode and state are incremented.

For the timed circular buffers, the storage format for each line is:

c0	c1	c2	c3	c4	c5	c6	c7	c8	c9

The c0 component contains a value combining both the mode and the state information by mean of the following function:

```
        value = (mode * 10 ) + state
```

In such a way, the states for the electron mode are represented by values in the interval (0-3) whilst the states for the positron mode are represented by values in the interval (10-13).

The following c1, c2, ..., c9 components contain all the ADC channel readouts converted in nC. Two running pointers (for e+ and e-) indicate the last acquisition: when the pointer reaches the bottom of the buffer, it wraps around and restarts from the beginning.

The circular buffer does not employ time tagging and the time base is settled by the 50 Hz sampling trigger. On idle machine state (state = -1) or in case of trigger lack, the buffer is not filled, with a consequent unknown time gap between the buffer lines. This means that the circular buffer grants the data pacing but gives no information about the absolute time value.

3.2 Slow Control Task

The SC task is in charge of logging the histogram arrays on disk, executing the user commands and updating the BCM readouts into the central memory area (which is accessible by the consoles).

3.2.1 Data Logging

As said above, the histogram memories reside on RAM. This means that their data are volatile and will be lost in case of a processor reboot. For such reason, it has been implemented a periodic dump of the histogram memories to log-files.

The SC task creates at 00h00 of each day a new log file (e.g. 20121231_histo.log) and then - every minute - it appends to the file a new record with all the histograms content.
The log-file is written in plain ASCII text (see Table 4 for the line format and Table 5 for an example of the log-file content).

Table 4: Format of the log file line. The "tab" character is used as fields separator and a "nl" as EOL.

Field	Format	Example
date	yyyymmdd	20121231
time	hhmmss	062023
mode (human readable)	{e I p}	e
mode (coded)	{0 I 1}	0
state (human readable)	{LSP I LBT I LAC I AMR}	AMR
state (coded)	{0 I 1 I 2 I 3}	3
monitor[0]	float64 value with 6 decimal digits [nC]	77445.739746
monitor[1]	float64 value with 6 decimal digits [nC]	0.000000
monitor[...]	...	...
monitor[8]	float64 value with 6 decimal digits [nC]	0.000000

Table 5: Example of the record appended every minute to the log file (the ellipses indicate the series of values for the intermediate monitors).

DATE AND TIME		MODE		STATUS		BEAM CHARGE MONITOR #1	• • • • • • •	BEAM CHARGE MONITOR #9
20131231	062023	e	0	LSP	0	0.000000		45818.181152
20131231	062023	e	0	LBT	1	0.000000		1253258.146973
20131231	062023	e	0	LAC	2	0.000000		90859.555664
20131231	062023	e	0	AMR	3	20353.630371		0.000000
20131231	062023	p	1	LSP	0	0.000000		9915.898437
20131231	062023	p	1	LBT	1	0.000000		313249.929199
20131231	062023	p	1	LAC	2	0.000000		149004.243164
20131231	062023	p	1	AMR	3	0.000000		0.000000

The SC task performs a straight transcription of the histogram memories to the log-file. This means that, should the processor reboot, the histograms written to the log file will restart from zero. The log files are consequently not suitable for a direct read-out of the total charge integral but must be post-processed. This has led to the development of the DJANGO project[4] which is specifically dedicated to the post-processing, storage and presentation of the data produced by this DAQ system (see Fig. 5).

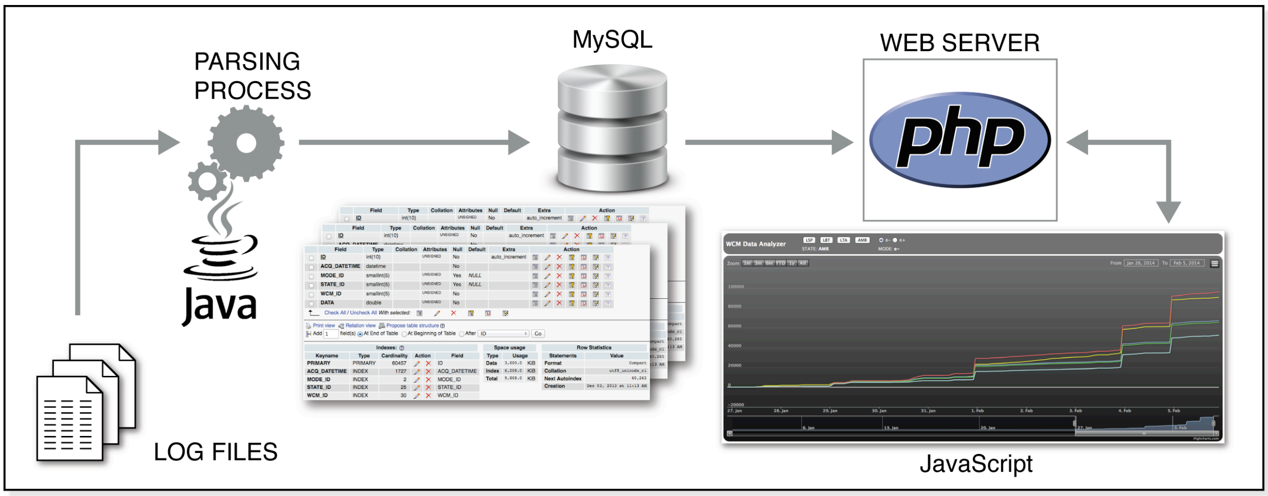


Figure 5: Workflow of the DJANGO system. A Java application processes the log files produced by the *SC task* and pushes the elaborated data into a relational database. A WEB application - employing JavaScript on the client side and PHP on the server side - displays an interactive plot of the integrated charge for the different machine states and modes.

3.2.2 User Commands

Each of the two DAQ system classes (*see par. 3*) has its own set of commands. Usually these commands are hidden to the user, being automatically created and issued by the GUI (Graphic User Interface) upon specific window actions.

A list of the implemented commands is reported below.

3.2.2.1 Commands for the Class BCM

```
SETT <elementName> {G|G1|G2},<dB>
```

This command changes the sensitivity of a BCM board [dB] and accordingly updates the corresponding conversion factor in RAM. This conversion factor [V/nC] is needed by the RTA task for a proper conversion of the ADC voltage readout [V] in charge value [nC].

elementName:	8 characters alphanumeric string that identifies the element
{G|G1|G2}:	parameter that indicates which stage has to be used to set the requested gain: G1 for stage 1, G2 for stage 2, G for both stages

dB:	value in dB of the gain. Legal values for <dB> are the following integers:
	G1 = 0, 6, 12, 20
	G2 = 6, 20
	G = 6, 12, 18, 20, 26, 32, 40

e.g.: SETT BCMTM001 G,26

```
SWTC <elementName> {INV|CAL},{ON|OFF}
```

The BCM boards can generate an internal voltage signal and use it as input for the integrating section, which is useful for calibration purposes. The boards also allow for the inversion of the input signal.

This command switches ON or OFF the Inversion and Calibration conditions of a BCM board.

When the command is related to the calibration mode, it also sets the sign of the corresponding conversion factor in RAM (positive for calibration = OFF, negative for calibration = ON). The RTA task uses the sign of the conversion factors to know if the BCM boards are in calibration mode or not.

elementName:	8 characters alphanumeric string that identifies the element

{INV|CAL}:	specifies whether Inversion or Calibration has to be set:
	INV = Inversion
	CAL = Calibration

{ON|OFF}:	specifies whether the specified condition has to be turned ON or OFF:

e.g.: SWTC BCMTM001 CAL,OFF

```
INIT <elementName>
```

This command is a macro that calls twice the SWTC executor with the following parameters

SWTC <elementName> INV,OFF
SWTC <elementName> CAL,OFF

elementName:	8 characters alphanumeric string that identifies the element

e.g.: INIT BCMTM001

3.2.2.2 Commands for the Class WCM

```
CALI <elementName> {EXEC|ON|OFF}
```

Contrary to what happens in the SWTC command of the BCM class (that affects the <u>internal</u> calibration of the BCM boards), the CALI command is concerned with the calibration obtained acquiring the toroidal monitors response to known pulses sent — on the φ_1 strobe — to coils wrapped around the toroids. The responses of the toroids are then acquired reading the ADC_1, converted with the proper conversion factors and stored as pure numbers (0 - 1) into a calibration array. A flowchart of the calibration algorithm performed by the RTA is shown in Fig. 6 .

elementName:	8 characters alphanumeric string that identifies the element
{EXEC}:	acquires, converts and stores the toroids response into a calibration array
{ON|OFF}:	specifies whether the ADC_4 readouts have to be dynamically multiplied by the calibration array or not.

e.g.: CALI WCMT*001 EXEC

```
LBUF <elementName> <requestID>,<number_of_lines>
```

Dumps the required number of lines from the circular buffer, going backward from the most recent one.

elementName:	8 characters alphanumeric string that identifies the element
requestID:	a numerical identifier
number_of_lines:	the number of lines that have to be recovered from the circular buffer

The LBUF command dumps *number_of_lines* lines of the circular buffer into a local variable. Then the user has to perform a TCP_DCS query with the opcode 3 (FETCH_U8_BUFFER), that brings the content of the local variable to the console level. A match of the first I32 number of the retuned data with the previously issued requestID, ensures that the data are related to the last issued LBUF command.

e.g.: LBUF WCMT*001 123,512

```
CMDS <elementName> <hexadecimalMask>
CMDC <elementName> <hexadecimalMask>
```

these two cammands allows to issue commands to the RTA:
- CMDS sets the *handshakeMaster* register bits (as indicated in hexadecimalMask) to "1";
- CMDC sets the *handshakeMaster* register bits (as indicated in hexadecimalMask) to "0".

elementName:	8 characters alphanumeric string that identifies the element
hexadecimalMask:	32 bit mask expressed in hexadecimal

Both CMDS and CMDC commands affect only the bits indicated by the hexadecimalMask, leaving the other ones unmodified. Possible values for the hexadecimalMask are:

0x0100	perform calibration (the bit is immediately reset to "0" by the RTA)
0x0200	apply calibration (the bit remains set to "1" and must be reset with a CMDC command)

e.g.: CMDS WCMT*001 400

3.2.3 Central Memory Update

Histogram memories and timed circular buffers reside in the level 3 processor RAM. Histogram memories are dumped by the SC task to log files whilst the circular buffers are dedicated to user's applications. As said above, the access to the circular buffers from the console requires two actions: the LBUF command and the FETCH_U8_BUFFER query. This means that - even though the system bandwidth is adequate to sustain a continuous dump of the buffers - they are not confortable data sources for a live display.

For this purpose, the level 3 processor continuously stores the last acquired charge values also into the central Object Caching[1] of the Control System which allows the user's application to get the most recent data at ease.

[1] The DAΦNE Control System is designed around a central storage area. The distributed processors - hosting the control programs - update the device data into this central cache. These data are then available to be fetched by the consoles running the user applications. In its first version, the Control System was using a common VME address space for the central cache. At the moment this method is being replaced by a Key-Value DB on RAM (*Memcached*). The main advantage of using the *Memcached* software instead of the VME address space is that it guarantees a complete independence from the hardware.

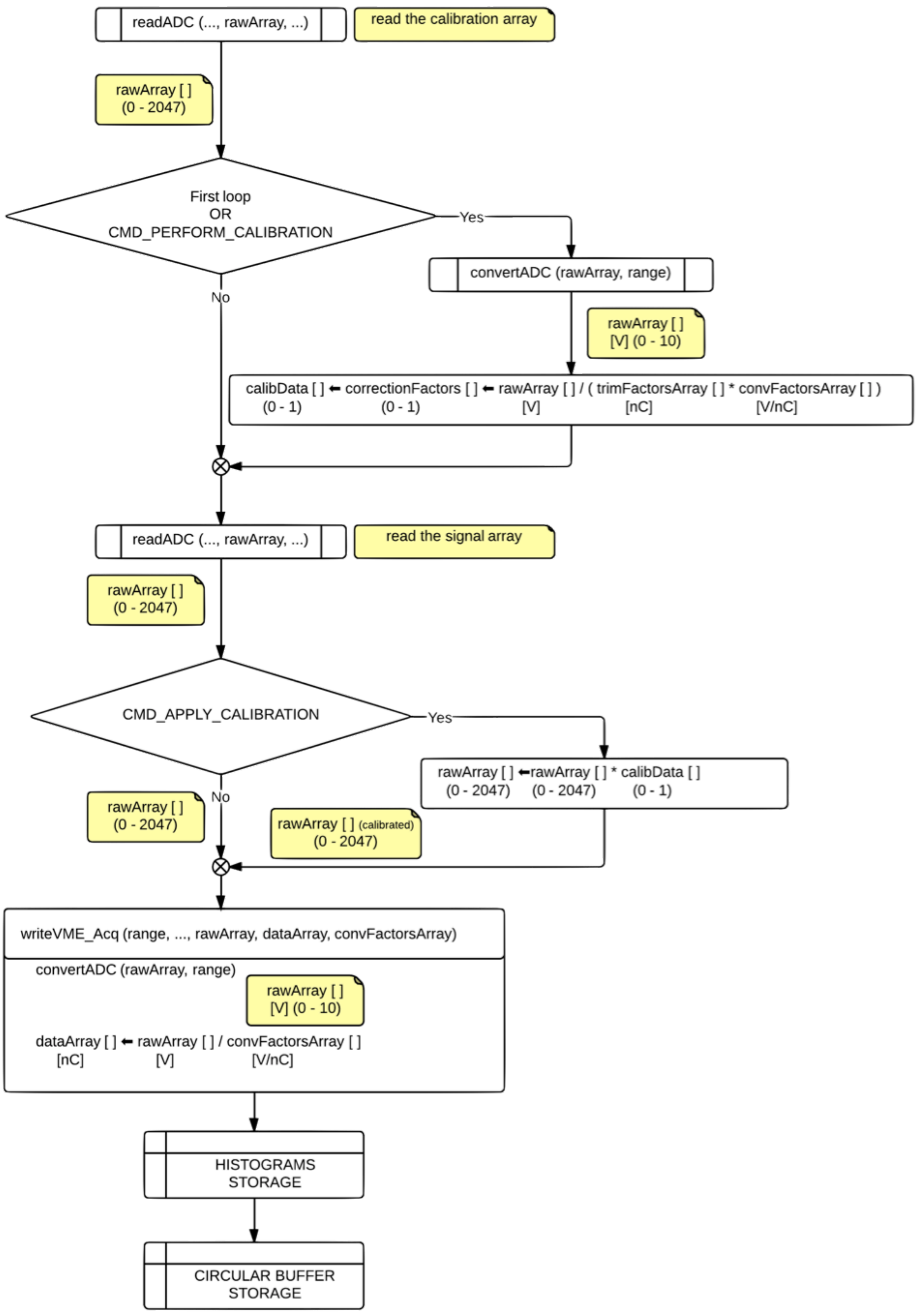

readADC (..., rawArray, ...)
read the calibration array
rawArray [ ] (0 - 2047)
First loop OR CMD_PERFORM_CALIBRATION
Yes
No
convertADC (rawArray, range)
rawArray [ ] [V] (0 - 10)
calibData [ ] ⬅ correctionFactors [ ] ⬅ rawArray [ ] / (trimFactorsArray [ ] * convFactorsArray [ ])
(0 - 1) (0 - 1) [V] [nC] [V/nC]
readADC (..., rawArray, ...)
read the signal array
rawArray [ ] (0 - 2047)
CMD_APPLY_CALIBRATION
Yes
No
rawArray [ ] ⬅rawArray [ ] * calibData [ ]
(0 - 2047) (0 - 2047) (0 - 1)
rawArray [ ] (0 - 2047)
rawArray [ ] (calibrated) (0 - 2047)
writeVME_Acq (range, ..., rawArray, dataArray, convFactorsArray)
convertADC (rawArray, range)
rawArray [ ] [V] (0 - 10)
dataArray [ ] ⬅ rawArray [ ] / convFactorsArray [ ]
[nC] [V] [V/nC]
HISTOGRAMS STORAGE
CIRCULAR BUFFER STORAGE

Figure 6: Flowchart of the calibration algorithm performed by the RTA.

4. User Interface

At console level, the user program keeps fetching the live data from the central cache.

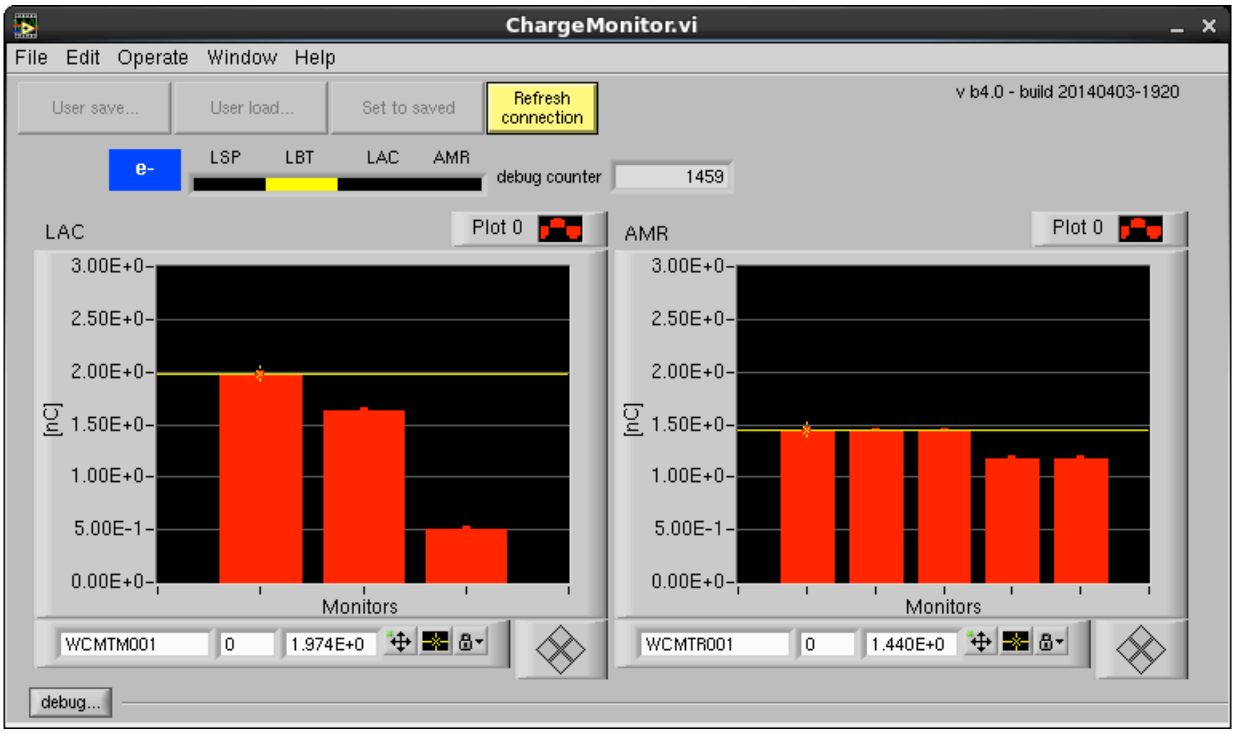


Figure 7: The user interface. At the top of the window, two indicators display the machine mode and status. The two bar-graphs below, show the charge across the monitors for the LAC and AMR states.

Currently, it has been developed a simple but effective user interface (Fig. 7) which displays the machine mode, status and the live graphs indicating the intensity of the beam charge along the transfer lines paths. Further versions will be released on the base of the users' needs.

5. Disclaimer

One of the possible uses of this DAQ system is to monitor over time - for radiation safety purposes - the integral of the beam charge injected into the DAΦNE damping ring and main rings.

With regards to this specific application, it has to be stressed that - even though all the hardware employed in the system (power supplies, electronic boards, processors, network, storage systems, and so on…) is high-end - the software has been designed to be compatible with the DAΦNE Control System framework rather than to run on redundant systems.

This means that, even though the information returned by this DAQ system can be considered reliable, it must not be used under any circumstance on matters of health security or general safety.

References

[1] G.Di Pirro et al: "DAΦNE Timing States Update", DAFNE Technical Note CD-8, February 1997 http://www.lnf.infn.it/acceleratori/dafne/NOTEDAFNE/CD/CD-8.pdf

[2] A. Ghigo, et al: "DAΦNE Beam Instrumentation", Proc. Of Beam Instrumentation Workshop (1998) USA, AIP Conference Proceedings 451, p.183.

[3] Bergoz Instrumentation: BCM Charge Monitor User's Manual, Bergoz Instr 01630 Saint Genis Pouilly (France).

[4] F. Agostini, P. Ciuffetti, A. Stecchi, "DJANGO: A Tool for the Analysis and Presentation of the DAΦNE Beam Integrated Charge Data", DAΦNE Technical Note C-21, February 2014.