EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
ORGANISATION EUROPEENNE POUR LA RECHERCHE NUCLEAIRE

CERN – PS DIVISION

PS/ BD/ Note 99-11

DSP SOFTWARE OF THE TUNE MEASUREMENT SYSTEM FOR THE PROTON SYNCHROTRON BOOSTER ACCELERATOR

M. Gasior, J.L. Gonzalez

Abstract

The tune measurement system for the 1.4 GeV CERN Proton Synchrotron Booster accelerator is based on Fast Fourier Transform analysis, performed by a digital signal processor. This paper describes the software that has been developed for this processor to fit to new hardware. We present a 2048-point real FFT algorithm, based on a 1024-point complex FFT routine, which has allowed the sampling rate to be doubled, still respecting previous system timing constraints. We also describe the peak search algorithm and explain the process of the FFT spectrum interpolation that improves the system accuracy by about one order of magnitude. Actual measurements have shown that the system provides good results, even with input signal amplitudes much lower than the voltage corresponding to one LSB of the ADCs. This software could also be used for the 26 GeV Proton Synchrotron tune measurement system, equipped with new hardware similar to that of the Proton Synchrotron Booster machine.

Geneva, Switzerland
15 September 1999

1. Introduction

The tune measurement system for the 1.4 GeV CERN Proton Synchrotron Booster (PSB) accelerator is based on Fast Fourier Transform (FFT) analysis, performed by a digital signal processor. This paper describes the software that has been developed for this processor to fit to new hardware **[1]**.

We present a 2048-point real FFT algorithm, based on a 1024-point complex FFT routine, which has allowed the sampling rate to be doubled, still respecting previous system timing constraints. We also describe the peak search algorithm and explain the process of the FFT spectrum interpolation that improves the system accuracy by about one order of magnitude. Actual measurements have shown that the system provides good results, even with input signal amplitudes much lower than the voltage corresponding to one LSB of the ADCs.

This software could also be used for the 26 GeV Proton Synchrotron (PS) tune measurement system, equipped with new hardware similar to that of the Proton Synchrotron Booster machine.

2. Overview of the new PSB tune measurement system

The tune value, or so-called Q value, is the number of transverse betatron oscillations that particles perform about the closed orbit, during one revolution around the accelerator. To measure the tune of both the horizontal and vertical planes, it is usually necessary to excite the beam, using a kicker, in order to produce coherent oscillations that can be observed on dedicated pick-ups, which are installed in the ring. Due to the sampling action of the pick-up, the signal contains the *mode*-frequencies:

$$f_\beta = (m \pm Q) f_{rev} \tag{1}$$

where m is the mode and f_{rev} the revolution frequency.

The integer part of Q remains constant: respectively 4 and 5 for the horizontal and vertical planes in the PSB, and 6 for both planes in the PS. Thus, for the tune measurement, it is sufficient to determine the fractional part q or $(1 - q)$. Considering a mode (m = 4, 5 or 6) with a frequency between $0.1 \times f_{rev}$ and $0.5 \times f_{rev}$ leads to:

$$\frac{f_\beta}{f_{rev}} = q \quad \text{or} \quad (1-q) \tag{2}$$

Whether it is q or $(1 - q)$ can be established by other means. In the PSB and the PS, it is q in both planes.

Generally, the beam is not centred in the pick-up and the resulting signal also contains f_{rev}. Since both f_β and f_{rev} vary during acceleration, a convenient way to calculate q is to digitise the signal at a rate proportional to f_{rev} and perform FFT analysis on N samples. Taking into account the sampling rate $k_s \times f_{rev}$ and the spectral line number n_β that corresponds to the betatron frequency, equation (2) can be rewritten as:

$$q = k_s \frac{n_\beta}{N} \tag{3}$$

where n_β is the bin number corresponding to f_β, in the N point FFT spectrum,
k_s is the oversampling ratio, i.e., digitising clock = $k_s \times f_{rev}$.

Horizontal (H) and vertical (V) tune measurements can be performed separately, according to commands issued by an application program that runs on a workstation, which can display the results of a machine cycle in graphical format. This application communicates, via a network link, with the Device Stub Controller (DSC, based on a RIO8062 VME computer with PowerPC CPU) on which some real time tasks run continuously, under LynxOS. These tasks control all of the hardware in the DSC and co-operate with the digital signal processor (Motorola's floating point DSP 96002), on the DBV96 VME board **[2]**. The block diagram of the system is shown in **figure 1**.

The TG8 timing generator delivers the requested trigger signals for H and V channels, during the machine cycle. Each trigger is then synchronised, by the Burst and RF Trigger Generator modules **[3]**, to both the revolution frequency f_{rev} and the RF frequency f_{rf}, which is the frequency the beam is accelerated with. These triggers cause the kickers to produce power pulses that excite the beam during one revolution period.

Beam Offset Signal Suppressor (BOSS) units reject the revolution frequency component, which would otherwise drown the betatron frequency f_β.

Each of the four rings of the PSB, which are individually equipped with a pick-up and a BOSS, can be connected to the tune measurement system by means of an analogue multiplexer, controlled by the DSC via a standard I/O card. Since the beam intensity varies over a wide range, in order to make the most of the ADC input dynamics it is necessary to adjust the signal level before digitising it. This is done by means of a Gain Controlled Amplifier (GCA), which buffers the signal to the 14-bit NIM ADC.

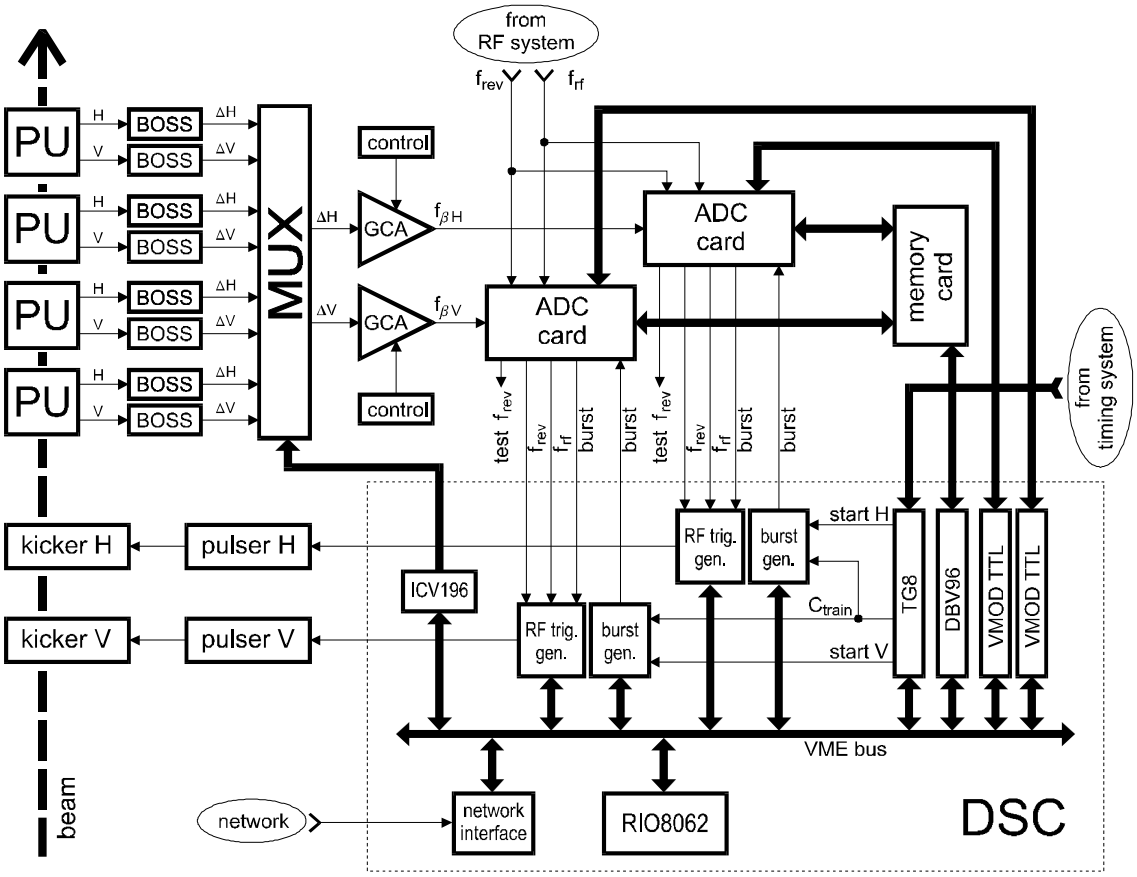

Fig. 1. Block diagram of the PSB tune measurement system.
Abbreviations: BOSS – Beam Offset Signal Suppressor; DBV96 – DSP VME board with DSP 96002/40 MHz; DSC – Device Stub Controller; GCA – Gain Controlled Amplifier; ICV196 and VMOD TTL – general purpose I/O ports; RIO8062 – VME embedded computer with PowerPC 603e/166 MHz processor; PU – pick-up; TG8 – timing generator.

The timings that initiate beam excitation also trigger data accumulation in two independent channels of the memory card, one for each plane. The DSP board takes this data, performs 2048-point real FFT analysis and searches for the spectral peak corresponding to f_β. After interpolation of its position, which improves system resolution, q values from 0.1 to 0.5 are calculated according to equation (3) with $k_s = 4$, $N = 2048$ and n_β a real number ranging from 50 to 256.

When the required data are collected, they are sent to the DSC via the shared memory, which stores them for the main application program. In case of problems during an acquisition, e.g., missing timings or ADC overflow, the DSP can send error messages to the application program, through the DSC. Because ADC overflow information is available, it can be used for automatic setting of the Gain Controlled Amplifiers.

The DSP is capable of servicing one channel in less than 5 ms; thus, one q value for each of the H and V planes can be obtained every 10 ms during a machine cycle. Furthermore, double buffering of data allows up to 5 ms delay between H and V acquisitions, so that one can check the coupling between both planes.

For testing purposes, each ADC card can internally generate f_β, f_{rev} and f_{rf} signals, controlled by the settings of external inputs or front panel switches. The simulated tune values are either 0.125 or 0.25.

3. DSP software

3.1. Starting up

Hardware or software resets of the DSC also reset the DBV96 VME board. At initialisation, the DSP executes an idle loop (at address `0x00` jump to address `0x00`) that is loaded as a boot program from a non-volatile memory located on the board. Thus, the DSC can access the DSP program memory via the VME-bus to transfer the main DSP program. An auxiliary DSP interrupt (feature of the DSP, allowing communication with its surrounding) makes the DSP run a small routine of the main program, which tells the DSC the address of DSP shared memory, where data will be exchanged. Then, the DSP returns to the idle loop until the DSC allows it to run the main program, which is executed from address `0x40000000`.

After each machine cycle, when the calculations are finished, the DSP informs the DSC that results are available in the shared memory, which is accessible via the VME bus.

The DSP software controls two LEDs on the front panel of the DBV96 board (called "*left*" and "*right*" user LEDs, according to their position). They are used to show the current state of the DSP program. When the program starts, both LEDs flash alternately during one second.

3.2. Software frame

Figure 2 shows the flow diagram of the software. An interrupt routine allows the DSC to send the commands. The DSP software is essentially an endless loop, with processing tasks.

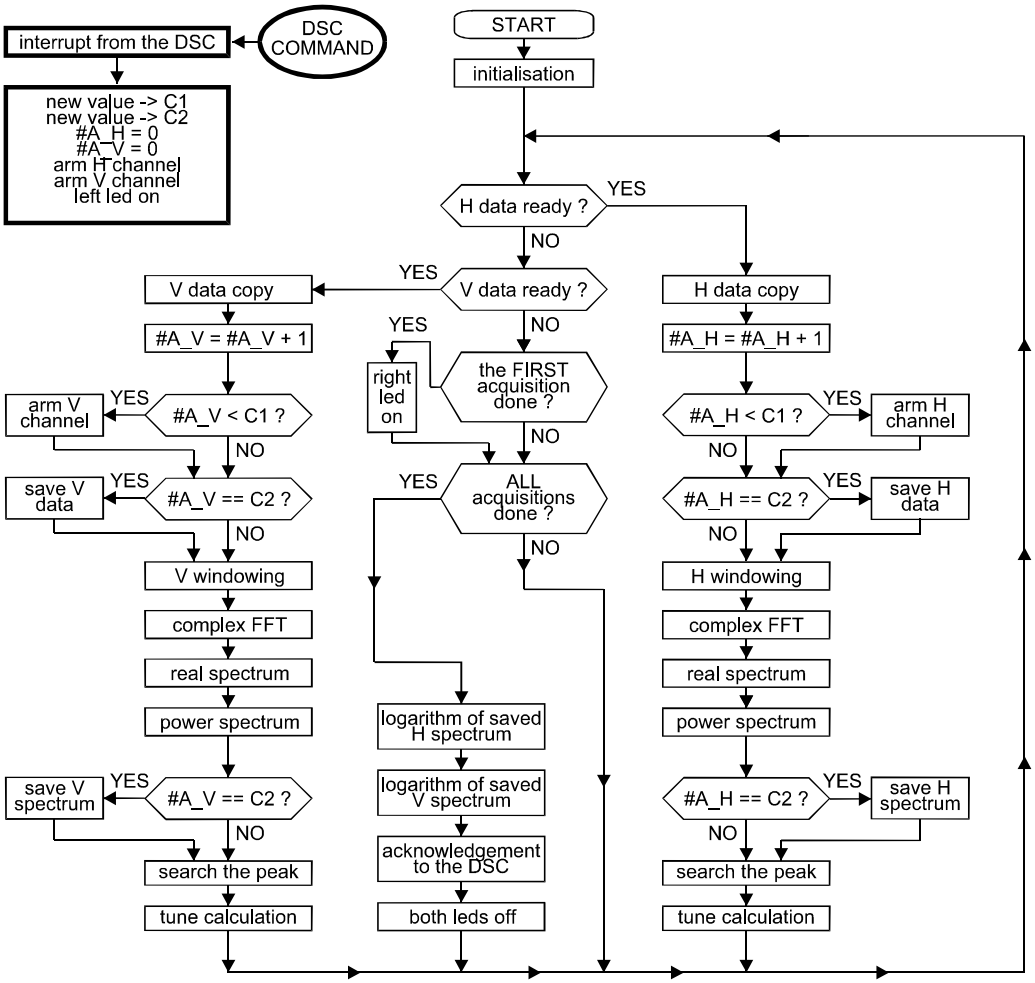

Fig. 2. Flow diagram of the DSP program. For the sake of simplicity, the error handling is not shown.
The program contains an interrupt routine and an endless loop. Interrupts allow getting commands from the DSC and the loop performs the required processing. Variables #A_H and #A_V indicate respectively the number of acquisitions that have already been done in the current machine cycle, on the horizontal and vertical planes respectively.

The command, which is transmitted using a DSP interrupt, contains two integer numbers, *C1* and *C2*. *C1* stands for the number of measurements that must be performed on each channel (i.e., number of active hardware triggers) during the current machine cycle. For each hardware trigger, an acquisition means collecting 2048 ADC samples in both channels of the memory board and the associated DSP processing, in order to calculate a tune value for the horizontal and vertical planes. *C2* is a "cursor" that points at one acquisition, for which the ADC samples and their calculated logarithmic spectrum will be saved. Due to the amount of data, which should be transmitted to the DSC, it is necessary to limit this treatment to only one acquisition per machine cycle. The set of data sent to the DSC will be described later.

After initialisation, the DSP runs the endless loop checking if data are ready, from one of the system channels (horizontal or vertical). However, as long as the DSP does not get a DSC command, the memory board channels are not activated and hardware triggers are ignored. Indeed, the system is waiting for a DSC command.

At the beginning of each one-second machine-cycle, if tune measurements are required, the DSC sends a command to the DSP and waits for results on this cycle. To show current DSP software status, i.e., a valid

command has been received, the *left* user LED is set on. When the DSP gets a command, both memory board channels are activated and next hardware triggers will start collecting the required ADC samples.

When a trigger occurs and first acquisition on one of the channels is finished, i.e., 2048 samples are available in the memory board, the DSP starts servicing this channel, while data of the second channel can still be collected for later processing. If data acquisitions end simultaneously on both channels, the horizontal channel is handled first. This may happen because the timing between the channels can be adjusted in 1 ms steps from 0 to 5 ms.

Prior to any processing, the DSP starts copying the data from the VME memory board channel to DSP memory (double buffering). When data are transferred, it re-arms the memory board channel to enable hardware triggers and allow next acquisition. Then, it calculates the tune value and stores it in the result data structure that is sent to the DSC, at the end of the machine cycle. After the very first acquisition is completed, the *right* user LED is set on.

When the tune value of the first channel is stored, the DSP handles the second channel unless the first channel is ready earlier. These actions are performed until the required acquisitions on both channels are completed. After that, the DSP sends a message to the DSC, via an interrupt, meaning that all data are present in the shared memory, and waits for next DSC command.

The DSC then reads the data in the shared memory and makes it available to the main application program, which communicates via a network link. Afterwards, both user LEDs are turned off, which means that there is no command to execute (*left* user LED) and the DSP does not process any data (*right* user LED).

The control parts of the DSP tune measurement software are written in the C language **[4]**. Signal processing and sections needing a lot of computing time (e.g., data transfers) are written directly in the assembler language of the DSP 96002 processor **[5]**. This approach combines clear data and control flows with high processing speed.

3.3. Acquisition preparation

At initialisation, the DSP generates coefficient tables for data windowing and FFT processing. Then, the system waits for a DSC command and the DSP runs the endless loop, checking for data ready from one of the two system channels (horizontal or vertical). This test is done by polling the dedicated *AcqFinished* bits, on the Control Registers of the memory board. **Table 1** presents the Control Register bit assignments.

Bit #	Name	Description
0 (LSB)	AcqArmed	0: grant access to every counter and register 1: access is restricted to control register and spare register; next active *burst* will trigger the acquisition
1	AcqRunning	1: acquisition is running; setting it causes a software trigger and acquisition starts without hardware trigger 0: acquisition not running
2	AcqFinished	1: acquisition is finished; testing this bit allows to detect the end of the acquisition 0: acquisition not finished
3	IntEnable	1: interrupt enabled; when acquisition is finished a dBeX interrupt is generated 0: interrupt disabled

Table 1. Memory board Control Register bit assignments (for memory board description and specification consult ref. [1]).

The memory board is connected to the DSP board through the specialised dBeX interface **[2]**, located in the DSP memory (Y memory space) at addresses starting from `Y:0xffffff80`. This address is set as the memory board base address and the memory board registers occupy next 12 locations. They are listed in **table 2**.

Channel	Address	Write access meaning	Read access meaning
0	base + 0	Address Counter bits 18-16	memory data @ address pointed by Address Counter
0	base + 1	Address Counter bits 15-0	Control Register
0	base + 2	Data Counter	Control Register
0	base + 3	Control Register	Control Register
0	base + 4	Spare Register	Control Register
1	base + 8	Address Counter bits 18-16	memory data @ address pointed by Address Counter
1	base + 9	Address Counter bits 15-0	Control Register
1	base + 10	Data Counter	Control Register
1	base + 11	Control Register	Control Register
1	base + 12	Spare Register	Control Register

Table 2. Addressing the memory board registers and counters.

The memory board channels are disabled until the DSC issues the required command. At reception of this command, the DSP activates the memory board and separate hardware triggers from the TG8 timing generator start acquisitions on both channels.

Preparing a channel for an acquisition mainly consists in setting the memory board registers, according to the following sequence:

1. Initialise the Control Register (value 0) to grant access to other registers of the memory board.
2. Set the Address Counter to the starting address, at which the data will be stored. This must be done in two steps: first load Address Counter bits 18-16 and then Address Counter bits 15-0.
3. Load the Data Counter with the number of samples that must be accumulated (i.e., 2048).
4. Set the *AcqArmed* bit in the Control Register (see table 1) to activate the channel.

After data are stored in one of the memory board channels, hardware triggers are disabled until another channel is activated (repeating the above sequence). So, this procedure is performed on both channels before each acquisition.

3.4. Channel data handling

3.4.1. Data fetching

Double buffering allows data treatment on the DSP board and acquisition in the memory board to proceed concurrently. Testing for ADC overflow occurrences is performed during the copy process to the DSP memory, while data are still in the DSP registers. This kind of information is added to every ADC sample by a dedicated encoder on the ADC board, when converting the ADC 14-bit natural binary data into 16-bit two's complement data for the DSP. The overflow information is coded, using the two most significant bits *D15* and *D14*, according to the following convention: the input signal exceeds the positive limit when $D15 = 0$ and $D14 = 1$, it exceeds the negative limit when $D15 = 1$ and $D14 = 0$, and during normal operation $D15 = D14$. When the DSP finds any overflow, a special bit is set on an error indicator, which accumulates all possible errors from the acquisitions during one machine cycle. This indicator is part of the error status word that is contained in the result data structure, which is sent to the DSC after the acquisitions are processed, or after time-out in case of lack of timings. Overflow information can be used for automatic setting of the Gain Controlled Amplifier.

After the overflow test, while copying from the memory board, integer ADC data are converted to the DSP standard floating-point format (IEEE 754-1985).

When the acquisition number in the current machine cycle corresponds to the second DSC command number, the ADC samples, which are now in floating-point format, are stored in the result structure that is transferred to the DSP at the end of the cycle.

3.4.2. Data windowing

The Discrete Fourier Transform (DFT), calculated with the FFT algorithm, assumes that the data to be analysed constitute one cycle of a periodic signal. Because in our system the signal is cut out in a random way, to allow periodic extension of such a pattern, without discontinuities at the edges, it is necessary to attenuate the beginning and the end of the data. This is achieved using windowing **[6]**.

Moreover, since we want to increase measurement accuracy, we need a clean spectrum to apply interpolation. This is facilitated with the Blackman-Harris window, which gives the biggest signal attenuation at the edges. It is shown in **figure 3**, together with some other commonly used functions.

To apply the Blackman-Harris window to the N samples located in the DSP memory, the

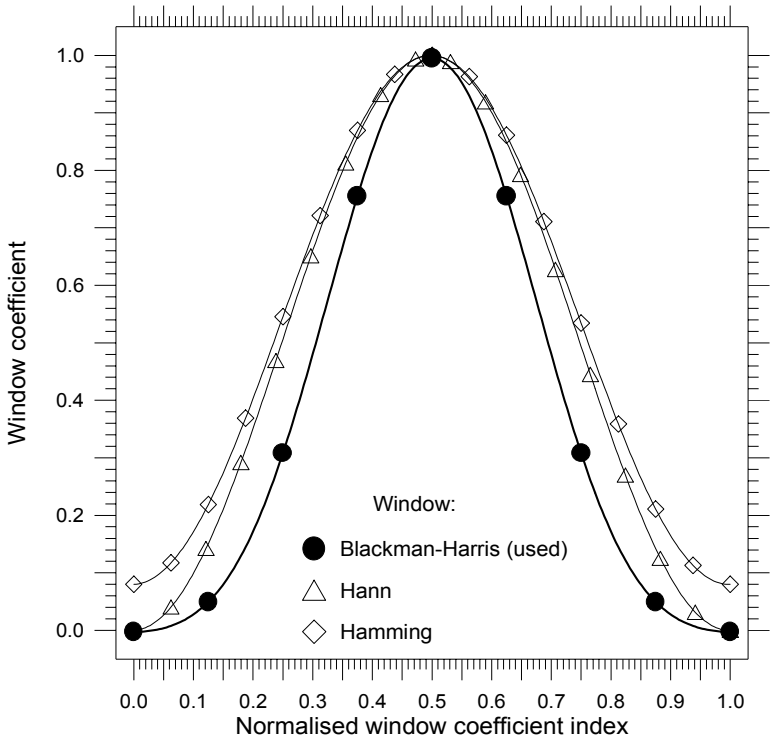

Fig. 3. Windowing functions.
The Blackman-Harris window gives the biggest border attenuation.

processor multiplies every sample by a coefficient c_i, which is previously stored in a table and calculated according to equation:

$$c_i = 0.40217 - 0.49703\ \cos\left(\frac{2\pi}{N} i\right) + 0.09392\ \cos\left(\frac{2\pi}{N} 2i\right) - 0.00183\ \cos\left(\frac{2\pi}{N} 3i\right) \qquad (4)$$

where i is the sample index and N the total number of samples (in this system $N = 2048$).

3.4.3. FFT processing

The main signal processing is based on a standard 1024-point complex FFT, completed with an additional algorithm, which produces the real signal spectrum from the complex spectrum data.

In order to calculate the 1024-point complex FFT, the 2048 windowed samples must be distributed in the following way: successive samples with even indexes constitute the real part of the complex data and odd samples form the imaginary part. Extracting the real signal spectrum, from the FFT complex results, is achieved using the mathematical rules that we derive in Appendix A.

The resulting spectrum is then transformed to a power spectrum, for the peak search routine.

3.4.4. Spectrum peak search method

The oversampling ratio of the PSB Q-measurement system, k_s in equation (3), amounts to 4. Thus, for tune value calculations, only a quarter of the total power spectrum bins must be taken into account, i.e., from 2048 points only 512 are necessary. Furthermore, since tune values vary from 0.1 to 0.5, the search is limited to the window $\langle n_B ; n_E \rangle$, where n_B and n_E are 50 and 256 respectively. To perform further interpolation and calculate the tune, one needs to find the index of the highest peak, within this power spectrum range, which corresponds to the betatron frequency f_β.

The peak search function is based on comparison of power spectrum bin values to maximum value conditions. The maximum is determined by a bin with index n_β and power spectrum value $V^2(n_\beta)$ when the four following conditions are fulfilled:

a) Previous bin is smaller than bin number n_β: $V^2(n_\beta - 1) < V^2(n_\beta)$
b) Next bin is smaller than bin number n_β: $V^2(n_\beta + 1) < V^2(n_\beta)$
c) Bin power value $V^2(n_\beta)$ is the biggest of all bins satisfying conditions a) and b) within n_B to n_E
d) Bin power value $V^2(n_\beta)$ is at least *three times* bigger than the arithmetic mean of all power bins, within indexes ranging from n_B to n_E (the root of this arithmetic mean corresponds to the RMS value). The threshold "*three times*" was determined experimentally during laboratory tests and with PSB machine measurements, as a compromise between system-input dynamics and system robustness in case of either very small or noisy betatron signals.

If no bin fulfilling these conditions can be found, meaning either that the spectrum peak is too small or the spectrum is too noisy, interpolation is omitted and zero is returned as the tune value. Zeros can be easily distinguished, on the final tune graph, as invalid results.

Among several algorithms we tried, this one was the fastest and gave best results. Most of it is written in the C language, but the calculation of the arithmetic mean is written in assembler.

3.4.5. Betatron signal spectrum interpolation

Within the PSB betatron frequency range, the biggest absolute error amounts to 0.5 bin, which leads to relative errors from 0.2 % to 1 %, depending on the peak position. This is far too much for tune value measurements. To reduce this error three solutions are possible:

a) Increase the number of points. This would need much more computing time, resulting in slowing down the system. The minimum time between two tune value calculations would be increased, leading to less information about tune changes during a machine cycle. Thus, this method cannot be used.
b) Decrease the oversampling ratio. On the ADC board, betatron signal hardware filtering is improved with higher oversampling ratio. Decreasing it would cause a degradation of the system input-dynamics. As this aspect is very important, due to the beam intensity span, which results in large variations on the betatron signal amplitude, this method is rejected.
c) Interpolate between two adjacent bins of the spectrum. Since the betatron signal contains mainly one interesting harmonic-like component, the spectrum peak will represent the betatron frequency f_β, which allows interpolation. We chose this method because its computational cost is low.

In the calculated spectrum, the betatron frequency f_β spreads over several bins, not only because of the discrete resolution of the FFT but also because:

- The betatron signal is not a pure harmonic signal. It includes attenuated revolution frequency f_{rev} harmonics, noise, etc.
- The windowing process does not cancel the discontinuity phenomenon of the acquisition.
- The coherent betatron signal f_β is damped in the time domain.

Interpolation allows the determination of the spectrum peak position when it is located between two spectrum bins. Such a situation is shown in **figure 4**. When the actual betatron frequency is bigger than the peak frequency bin, given by the peak search routine (**figure 4a**), the spectrum bin before the peak is smaller than the bin after the peak. Similarly, when the actual betatron frequency is smaller than the peak bin (**figure 4b**), the spectrum bin before the peak is bigger than the bin after the peak. This observation leads to the interpolation of the spectrum around the peak bin found.

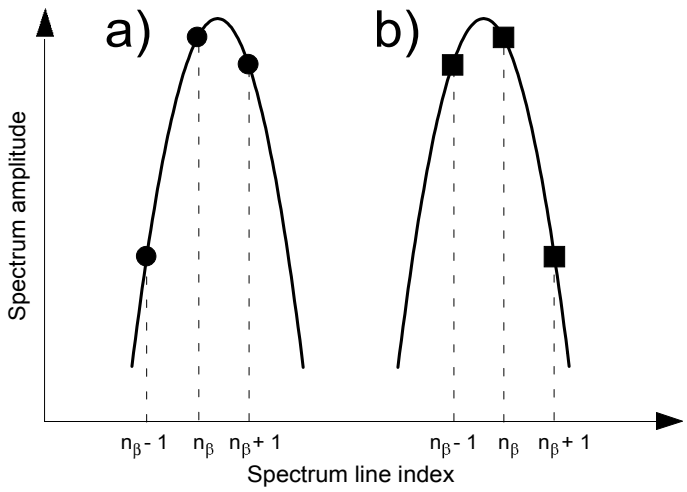

Fig. 4. Interpolation based on parabolic spectrum.
a) Interpolated peak frequency is bigger than the peak bin.
b) Interpolated peak frequency is smaller than the peak bin.

The exact peak surrounding form, which depends on the betatron signal features, is unknown. However, assuming a parabolic shape, as in **figure 4**, and naming n_β the peak index given by the peak search routine, the following equations describe the 3 bin values, related to the peak position:

$$\begin{cases} V(n_\beta - 1) = a\,(n_\beta - 1)^2 + b\,(n_\beta - 1) + c \\ V(n_\beta) = a\,n_\beta^2 + b\,n_\beta + c \\ V(n_\beta + 1) = a\,(n_\beta + 1)^2 + b\,(n_\beta + 1) + c \end{cases} \tag{5}$$

where $V(n_\beta - 1)$, $V(n_\beta)$ and $V(n_\beta + 1)$ are the spectrum values for indexes $n_\beta - 1$, n_β and $n_\beta + 1$;
a, b and c are the parabola coefficients we wish to find.

Solving this set of equations for variables a, b and c, one gets:

$$\begin{cases} a = \frac{1}{2}V(n_\beta - 1) - V(n_\beta) + \frac{1}{2}V(n_\beta + 1) \\ b = -2\,a\,n_\beta + \frac{1}{2}\left(V(n_\beta + 1) - V(n_\beta - 1)\right) \\ c = a\,n_\beta^2 - \frac{1}{2}n_\beta\left(V(n_\beta + 1) - V(n_\beta - 1)\right) + V(n_\beta) \end{cases} \tag{6}$$

The interpolated peak index n'_β, which is the parabola peak, is a function of coefficients a and b:

$$n'_\beta = -\frac{b}{2\,a} = \frac{2\,a\,n_\beta - \frac{1}{2}\left[V(n_\beta + 1) - V(n_\beta - 1)\right]}{2\,a} = n_\beta - \frac{\frac{1}{2}\left[V(n_\beta + 1) - V(n_\beta - 1)\right]}{V(n_\beta - 1) - 2V(n_\beta) + V(n_\beta + 1)} \tag{7}$$

The value n'_β, from this equation, is inserted into equation (3) in order to calculate the interpolated tune value.

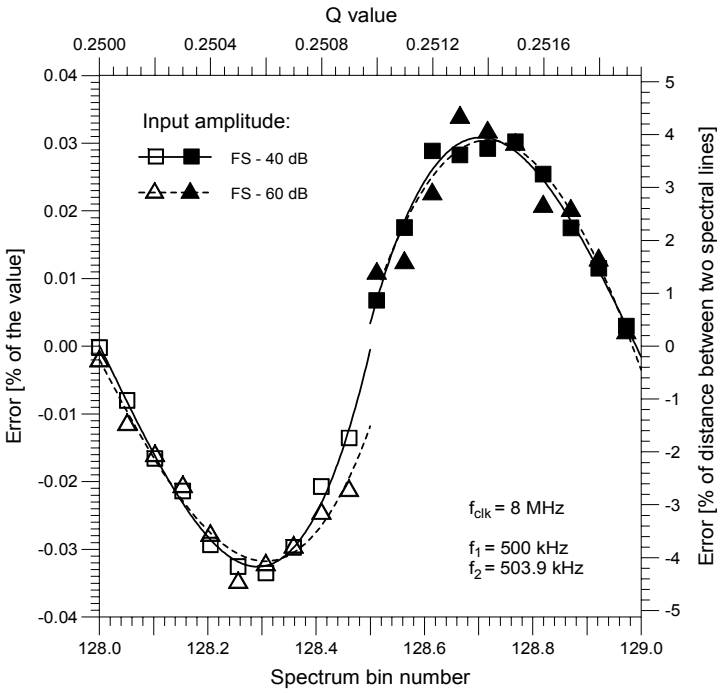

Fig. 5. Harmonic signal spectrum interpolation error.
Square markers and solid line curves result from peak-peak input amplitudes of 40 dB below ADC full scale (FS), while triangles and dashed line curves are from input amplitudes of 60 dB below ADC FS. Markers stand for measurement data points and lines correspond to polynomial interpolation of order 3. Left Y-axis represents the relative error and right Y-axis the absolute error.

Figure 5 illustrates the interpolation error, using 2 input signals with peak-peak amplitudes respectively 40 dB and 60 dB below ADC full scale. Input frequency was varied around the middle of the system working range, from f_1, corresponding to spectral line index 128; to f_2 corresponding to bin index 129. For the system clock of f_{clk} = 8 MHz, f_1 and f_2 are respectively 500 kHz and 503.9 kHz. They represent Q values from 0.25 to around 0.252 (upper X-axis).

On left half of the chart, interpolation is around bin number 128 (i.e., on bins 127, 128 and 129) and on right half of the chart, around bin number 129 (i.e., on bins 128, 129 and 130).

The chart shows that the biggest interpolation errors occur when points are away from a bin, at ¼ of the distance between two adjacent bins. Notice that worst case relative errors are about 0.04 % and absolute errors are 5 % of the adjacent bin distance, while they would have been 0.4 % and 50 % respectively without interpolation. Thus, interpolation improves resolution by one order of magnitude. However, for the total frequency span, relative error varies from 0.1 % to about 0.02 %, corresponding to bin numbers 50 and 256 respectively.

In these error measurements, with relatively high input signal amplitudes, the system resolution is determined by interpolation accuracy. When the input amplitude decreases, noise prevents the interpolation process to work accurately and errors increase. Thus, the influence of the input level on system accuracy depends on system noise performance. The system dynamics will be examined later.

3.4.6. Acquisition results

```
#define NUMBER_OF_Q_VALUES      200            /* Maximum number of Q values per plane, calculated
                                                  during one machine cycle.                          */

#define NUMBER_OF_SENT_FFT      512            /* FFT values per plane sent to the DSC.
                                                  They are logarithms of the power spectrum and they
                                                  are saved for an acquisition indicated by the second
                                                  number of the DSC command.                         */

#define NUMBER_OF_RAW_ADC_DATA  2048           /* ADC raw data values per plane.
                                                  They are saved for an acquisition indicated by the
                                                  second number of the DSC command.                  */

#define NUMBER_OF_ERR_CHANNELS  2              /* Number of error channels.
                                                  First one is a variable whose bits consist of error
                                                  flags from all acquisitions of one machine cycle.
                                                  Second one is introduced for future development.   */

struct
{
    float Q_Hor[NUMBER_Q_VALUES];              /* Collected H Q values to be sent to the DSC         */
    float Q_Ver[NUMBER_Q_VALUES];              /* Collected V Q values to be sent to the DSC         */
    float FFT_Hor[NUMBER_SENT_FFT];            /* Logarithm of H power spectrum to be sent to the DSC */
    float FFT_Ver[NUMBER_SENT_FFT];            /* Logarithm of V power spectrum to be sent to the DSC */
    float RawAdcData_Hor[NUMBER_RAW_ADC_DATA]; /* ADC H raw data to be sent to the DSC               */
    float RawAdcData_Ver[NUMBER_RAW_ADC_DATA]; /* ADC V raw data to be sent to the DSC               */
    float ErrorChannels[NUMBER_ERR_CHANNELS];  /* Error channels DSP -> DSC carrying error indicators */
} ResultsForDsc;
```

Listing 1. The result structure sent from the DSP to the DSC.

At the end of the machine cycle, when the tune measurement processing has been done, the "cursor" acquisition can be calculated using previously stored data. Processing of the base 10 logarithm power spectra for the horizontal and vertical planes needs a few milliseconds, but since it is executed after the acquisitions, it does not affect the system timings.

After cursor treatment, the DSP informs the DSC that data are ready, except when errors occur during a machine cycle (case discussed later). This is done through an interrupt, and the DSC, which already knows the address of the result buffer, copies the data from the DSP memory, via the dBeX interface, to its own memory. **Listing 1** shows the result structure.

In the DSP program, float error elements have integer mirrors, which are convenient for fast bit manipulation. The error element, which is transmitted to the DSC, is a float and allows treating all the data as one block. Later on, the DSC converts these error indicators back to integer format and tests error flags.

The float format is the standard IEEE 754-1985, which is understood by the DSP, the DSC and workstation processors, while integer formats may depend on particular processors or compilers.

3.5. Exception handling

Usually, DSC commands cannot be transferred to the DSP before the previous cycle is finished because, after sending a command, the DSC waits for the acquisition results. However, if the DSC does not get a DSP acknowledgement after a few seconds, a time-out action is taken up. In that case, the DSC reads incomplete DSP results, with dedicated error flags, which indicate that the acquisition was not finished. This and other error information is used to produce some DSC error messages.

Listing 2 shows error masks corresponding to error flags in the first error element. If a flag is set, the application program can display the associated error message. The second error element is currently unused and the DSP program sets it to zero for compatibility with eventual upgrades. However, if it is found to be non-zero, `ERR_MESG_ERRCHANLTWO` can be displayed (during tests or development).

```
/* ================  Error masks and messages sent from the DSP to the DSC.  ================
   ================  They are defined in both the DSP and the DSC.           ================

   In the DSP program, all operations are performed on the integer elements of the array:

      int ErrorChannels[NUMBER_ERR_CHANNELS], where NUMBER_ERR_CHANNELS is 2.

   After operations, these elements are ascribed to the appropriate float format to the array:

      float ErrorChannels[NUMBER_ERR_CHANNELS];

   This float array is transmitted with other results from the DSP to the DSC. There, it is
   converted back to the DSC integer format.

   Only ErrorChannels[0] is used. ErrorChannels[1] should be set to 0 for future compatibility;
   however, ERR_MESG_ERRCHANLTWO can be displayed if it is not 0.

   Integer format error flags, with their masks, and error messages are defined below.
   Each error message is valid if the associated flag is set.
*/

#define  ERR_MASK_OVF_H           0x01                          /* ==== MASK                ==== */
#define  ERR_MESG_OVF_H           "HORIZONTAL ADC overflow."    /* ==== appropriate MESsaGe ==== */

#define  ERR_MASK_OVF_V           0x02
#define  ERR_MESG_OVF_V           "VERTICAL ADC overflow."

#define  ERR_MASK_NOTALL_H        0x04
#define  ERR_MESG_NOTALL_H        "HORIZONTAL measurement not completed."

#define  ERR_MASK_NOTALL_V        0x08
#define  ERR_MESG_NOTALL_V        "VERTICAL measurement not completed."

#define  ERR_MASK_BADMEASNUMB     0x10
#define  ERR_MESG_BADMEASNUMB     "BAD measurement number. Measurements not done."

#define  ERR_MASK_BADRWDTAPTR     0x20
#define  ERR_MESG_BADRWDTAPTR     "BAD raw data pointer. Measurements not done."

#define  ERR_MESG_ERRCHANLTWO     "Nonzero second error channel. Source to be defined."
```

Listing 2. Error masks and associated messages.

Flags corresponding to the masks `ERR_MASK_OVF_H` and `ERR_MASK_OVF_V` are set when any H or V ADC sample in the cycle contains the overflow information. This information can be used for the gain setting of the Gain Controlled Amplifiers.

Flags related to ERR_MASK_NOTALL_H and ERR_MASK_NOTALL_V are set when a time-out has occurred in the corresponding channel, usually due to missing triggers. These masks are set at the beginning of each cycle and reset when the acquisitions are completed. Thus, when the DSP hangs up, these flags remain set.

Flags corresponding to ERR_MASK_BADMEASNUMB or ERR_MASK_BADRWDTAPTR are set when the DSP does not understand the related first or second number of the DSC command. This could happen not only because these numbers are either negative or not integers, but also because the first command number exceeds its upper limit NUMBER_OF_Q_VALUES (see **listing 1**) and, respectively, the second number is greater than the first number. In such situations, only the flags are important. So, to avoid the delay of time-outs, the DSP only sets the flags, skips acquisitions and sends an acknowledgement to inform the DSC that results are ready.

4. DSC software

The DSC hosts the Equipment Module (EM) QMEA-V and the Real Time Task (RTT) qmeas. The EM is a collection of procedures and data that allows application programs to communicate with the equipment installed in the VME crate. Before transferring the requests to the RTT, the EM must check that they are legal, and subsequently receive resultant messages. The RTT manages all system functions: reception and treatment of control messages, handling of timing information, programming of the hardware I/O modules, communication with the DSP, and transmission of measurement results to the EM.

The Q-Measurement uses a PS programming model **[7, 8]**, based on the standard communication protocol. Like the rest of the accelerator equipment, this system draws information about the characteristics of a machine cycle from the so-called Program Line Sequencer (PLS) telegram. The EM and the RTT exchange information via control and acquisition message queues. Before the RTT is executed, the DSC start-up file rc.local is used to install these message queues as well as the device drivers, and to run the PLS software and the EM as background processes.

4.1. The Real Time Task

The program main(), which is located in the source file ini.c, is one of the six threads that constitute the RTT (**figure 6**). At start-up, to initialise the DSP, it calls the function read_def_values(), defined in set.c, which in turn uses special_setting() to launch open_dbv96k(), both situated in spe.c. Once the DSP is running, main() opens the message queues and the memory buffers, to copy the control and acquisition information corresponding to each user line. Then, prior to acting as a signal handler, it creates the generic threads w_pls(), ser() and acq_meas(), as well as the specific threads ctl() and meas(). Their respective source files are pls.c, ser.c, acq_meas.c, ctl.c and meas.c.

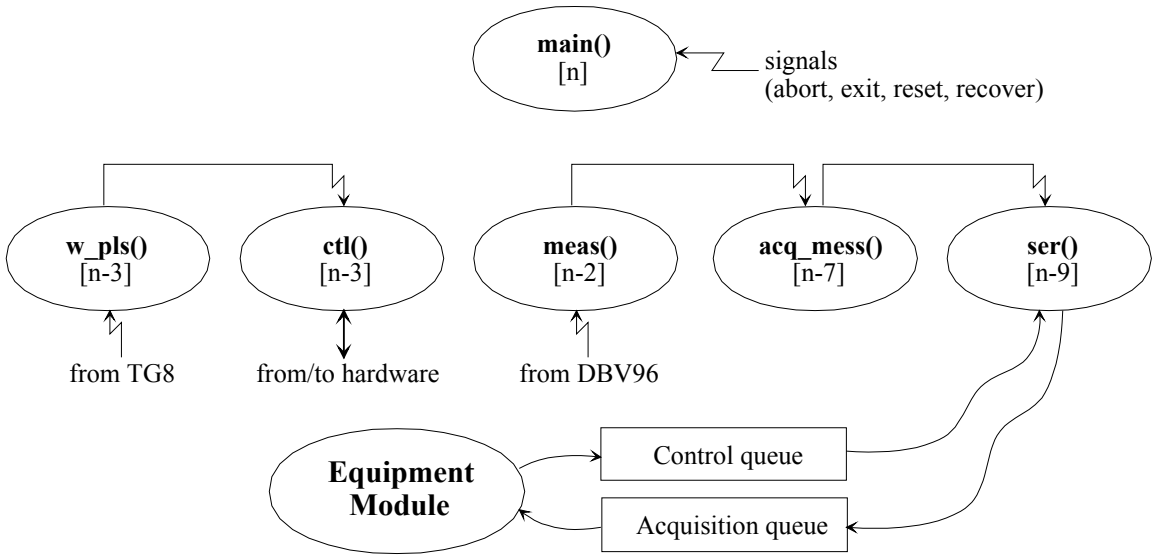

Fig. 6. General view of the RTT qmeas. The priority of qmeas, n = 50, is defined in rc.local.

As the threads are created, they are given a priority higher than that of main(), thus enabling them to run. While running, they readjust their priority to their designed values. Normally, all these threads are dormant until a signal or an interrupt wakes them.

4.2. Reception of a control message

When the EM puts a message into the control queue, the thread `ser()` is woken by a software signal. The purpose of `ser()` is to get and treat this message according to its type: synchronous or asynchronous.

Upon receipt of a synchronous message, `ser()` calls the function `syncserv()`, which services the message after the function `check_sync_message()` has verified its validity. The response to the EM depends on the contents of the message, predefined in the instrumentation protocol:

- `CTRL_B_U` (Control Back User) asks `ser()` to send back the control parameters of the user contained in the message
- `DATA_B_U` (Data Back User) and `DATA_B_C` (Data Back Cycle) request last data acquired for the specified user and cycle respectively
- `DATA_B_H` (Data Back Header) requires only the sending back of the message header
- `TEST_ARR` (Test Array) asks for an array of uncommitted floating point numbers, used to pass any data that is not predefined in the EM
- `FAUL_B_U` (Fault Back User) needs the equipment errors for the corresponding user
- `MESS_ERR` (Message Error) indicates that the received message is erroneous and cannot be interpreted.

In each case, `syncserv()` copies either the required information or an error indicator into the acquisition message structure that is sent to the EM. Then, `ser()` sleeps while waiting for a new message.

Asynchronous messages are handled in a similar manner, using the functions `asyncserv()` and `check_async_message()`. The difference with synchronous messages is that there is no immediate response to the EM. Moreover, when a control request is transferred via a test array, `spe_async()` is called for treatment (see `spe.c`). Then, `ser()` keeps waiting for next message. Only the demands contained in the specialist (`tst1`) or actuation (`ccac`) fields of the control message are executed:

- `TST`: allows the hardware to be set in simulation mode (0: OFF, 1: simulate $q1$, 2: simulate $q2$), for testing purposes
- `ACT_RESET` asks `main()` to release the memory buffers, kill the other 5 threads and create them again, in order to restart the RTT
- `ACT_RECOV` asks `main()` to only kill the 5 threads and re-create them
- `ACT_TEST` and `ACT_LOCAL` request to put the system in test and local modes respectively
- `ACT_ON` and `ACT_OFF` are concerned with requests for measurements on a specific user line (`ON` enables the measurement and `OFF` disables it).

4.3. Reception of a PLS telegram

On the arrival of a PLS telegram, the thread `w_pls()` is woken by an interrupt from a TG8. After extracting the user line and the cycle number, this thread sends a software signal to waken `ctl()`.

The purpose of `ctl()` is to perform all the necessary controls for the requested measurement, calling the function `do_control()`, after which it goes back to sleep, with `sigwait()`, until it gets a new signal.

The main control actions are: setting the number of acquisitions and their interval, in the VMOD-Burst Generators, enabling the kickers and setting their pulse duration, in the VMOD-RFTrigger modules, selecting the PSB ring and adjusting the Gain Controlled Amplifiers, in the VME I/O modules (ICV196, VMOD-DOR), and sending the acquisition parameters to the DSP, with the function `SendCommandToDBV()`.

4.4. Reception and transmission of measurement results

The thread `meas()`, which is waiting on `ReadResultFromDBV()`, is woken by the DSP when the Q-Measurement data are available. Subsequently, the function `signaler_acq_mess()` wakes up the thread `acq_meas()`, which continues building the message for the EM, using the function `traite_acq()` defined in `spe.c`, and stores the data in the acquisition buffer for the current user and cycle, with the function `stocker()`. The acquisition message is then sent to the queue and `meas()` goes back to sleep, awaiting a new signal.

5. Results

Several sets of acquisitions, of one thousand measurements each, have been used to estimate the yields of measurements that satisfy predefined accuracy levels, as a function of input signal amplitude, when actual frequency position lies between two spectrum bins. We assume the system works if at least 90 % of measurements are good.

Figure 7 presents worst-case results for 500.98 kHz, corresponding to spectrum position 128 + ¼, which is the worst case between two adjacent bins. Using input peak-peak voltage ratios of 1.4, 0.65, 0.33 and 0.25 (related to one LSB), led to relative errors lower than 0.05%, 0.1%, 0.2% and 0.5% respectively. Because the absolute error is almost constant, for tune values of 0.1 and 0.5, errors can be correspondingly 2.5 times bigger and 2 times smaller than presented. These measurements have shown that it is possible to resolve signals with amplitudes smaller than the voltage corresponding to one LSB of the ADCs, thanks to the ADC noise modulation phenomenon we described in [**1**].

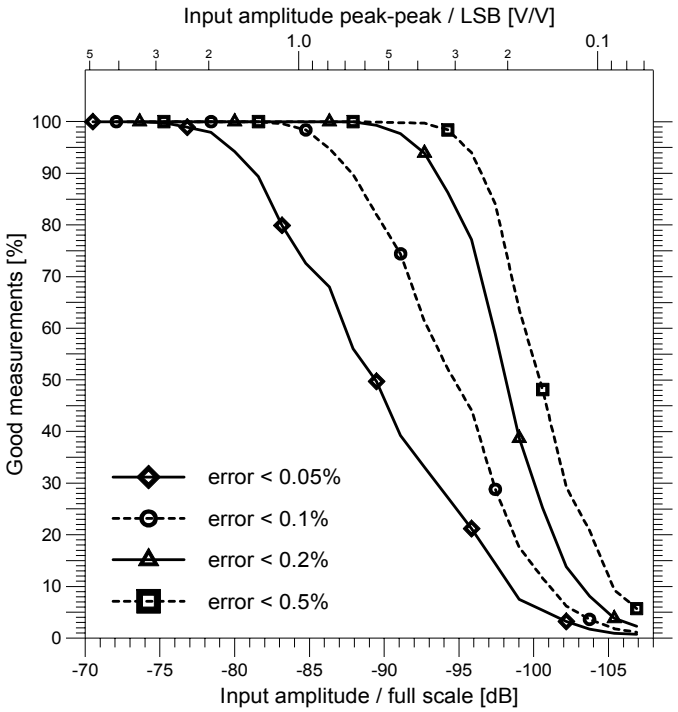

Fig. 7. Good measurement yield on 1000 measurements of 2048 points, as a function of input signal amplitude, for worst-case peak fitting around $q = 0.25$ (bin number 128+¼). Input peak-peak amplitudes: 81.5 dB, 88 dB, 94 dB and 96.5 dB below ADC full scale. Input frequency, 500 kHz; clock, 8 MHz.

Note that these results were obtained with a pure sine wave. So, in the case of a beam signal, which includes at least two components, f_β and f_{rev}, the input dynamic range could be slightly smaller.

Figure 8 shows an example of actual measurement results, during a PSB machine cycle, as presented on a workstation to the operators.

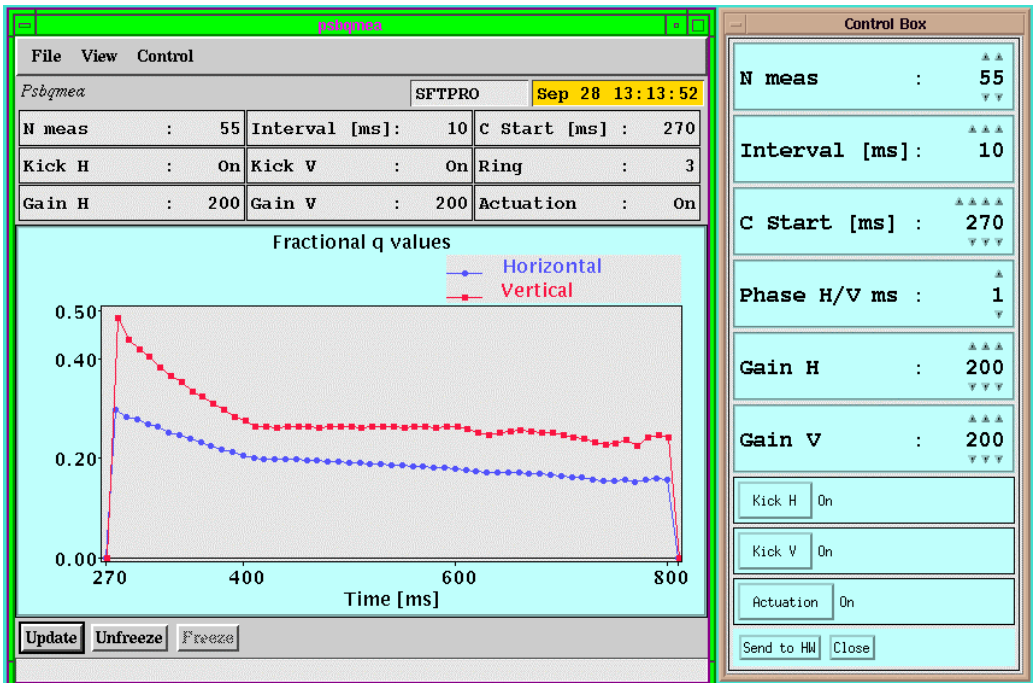

Fig. 8. PSB Q-Measurement, as displayed in the Main Control Room by the Application Program.

6. Conclusions

The PSB tune measurement software has been developed to fit to new hardware. The sampling frequency has been increased from $2 \times f_{rev}$ to $4 \times f_{rev}$ without sacrificing tune measurement resolution, because the FFT analysis is done now using 2048 points instead of 1024, respecting the same timing constraints **[9]**. To achieve this, it was necessary to introduce real data extraction from complex FFT calculation. The system gives good results, even with input signal amplitudes smaller than one LSB of the ADC, and the peak search method includes a threshold mechanism that blocks bad results when the input signal is too small or too noisy.

The improved performance of this new system allows successful measurements of rather low-intensity beams, despite the modest kicker voltage that is available now (500 V). An imminent kicker upgrade should yield still better results.

The current software is more reliable, thanks to a time-out mechanism. Error handling and diagnostic facilities have also been introduced and, at every machine cycle, the DSP informs the DSC about exceptional software status. Since the ADC overflow information is delivered to the DSC, it can be used to automatically control the gain of the analogue system channels.

Finally, this software can also be used for the tune measurement system of the 26 GeV PS, equipped with hardware similar to that of the PSB.

7. Acknowledgements

We are indebted to all those who were involved in the system software development, especially to E.T. d'Amico, A. Chapman-Hatchett, A. Gagnaire, M. Le Gras and N. de Metz-Noblat.

8. References

[1] M. Gasior, J.L. Gonzalez, *New Hardware of the Tune Measurement System for the Proton Synchrotron Booster Accelerator*. CERN/PS/BD Note 99-10, September 1999.

[2] Loughborough Sound Images plc, *VME Floating-Point DSP Board.* Technical Reference Manual. Version 3.00, June 1994.

[3] J.L. Gonzalez, J.T. Pons, *Burst and RF Trigger Generator Modules (VMOD-BURST and VMOD-RFTRIG)*. CERN/PS/BD Note 98-09 (Tech), July 1998.

[4] Intermetrics Microsystems Software Inc., *Inter Tools – 96002 C Compiler/Assembler User's Manual*. Version 3.7, January 1991.

[5] Intermetrics Microsystems Software Inc., *Inter Tools – 96002 C Compiler/Assembler Reference Manual*. Version 1.6, May 1991.

[6] F.J. Harris, *On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform*, Proceedings of the IEEE, Vol. 66, N° 1, pp. 51-83, January 1978

[7] M. Le Gras, J. Tedesco, *Application typique du protocole pour l'instrumentation*. CERN/PS/BD/Note 93-02, May 1993.

[8] S. Johnston, *Real-Time Program for the PS FFT Q-Measurement*. CERN/PS/BD/Note 94-02, January 1994.

[9] A. Chapman-Hatchett, V. Chohan, E.T. d'Amico, *Tune measurement for the CERN Proton Synchrotron Booster rings using DSP in VME.* CERN/PS 99-031 (BD). Particle Accelerator Conference, New York, USA, March 29-April 2, 1999.

[10] G.R.L. Sohie, W. Chen, *Implementation of Fast Fourier Transforms on Motorola's Digital Signal Processors*. APR4/D, Rev. 3, Motorola Inc., 1993.

Appendix A

Calculation method of the spectrum of 2*N* point real-valued signal using a standard *N* point complex FFT routine

Let us consider a real-valued data array $r(n)$, which contains the samples of a signal. Assume also that $n = 0, 1... 2N–1$, where N is an even number. Then, the Discrete Fourier Transform (DFT) $R(k)$ of the signal $r(n)$ is:

$$R(k) = \sum_{n=0}^{2N-1} r(n) \exp\left(- \mathrm{j} \frac{2\pi}{2N} nk \right) \tag{A-1}$$

where $k = 0, 1... 2N–1$.

This equation can be expanded, separating odd and even elements of $r(n)$:

$$\begin{aligned} R(k) &= \sum_{n=0}^{2N-1} r(n) \exp\left(- \mathrm{j} \frac{2\pi}{2N} nk \right) \\ &= \sum_{n=0}^{N-1} r(2n) \exp\left(- \mathrm{j} \frac{2\pi}{2N} 2nk \right) + \sum_{n=0}^{N-1} r(2n+1) \exp\left(- \mathrm{j} \frac{2\pi}{2N} (2n+1)k \right) \\ &= \sum_{n=0}^{N-1} r(2n) \exp\left(- \mathrm{j} \frac{2\pi}{N} nk \right) + \exp\left(- \mathrm{j} \frac{2\pi}{2N} k \right) \sum_{n=0}^{N-1} r(2n+1) \exp\left(- \mathrm{j} \frac{2\pi}{N} nk \right) \\ &= \sum_{m=0}^{N-1} r2(m) \exp\left(- \mathrm{j} \frac{2\pi}{N} mk \right) + \exp\left(- \mathrm{j} \frac{2\pi}{2N} k \right) \sum_{m=0}^{N-1} r1(m) \exp\left(- \mathrm{j} \frac{2\pi}{N} mk \right) \end{aligned} \tag{A-2}$$

where $r2(m)$ and $r1(m)$ are N-element arrays, which contain respectively the even and odd successive elements of $r(n)$:

$$\begin{aligned} r2(m) &= r(2n) \\ r1(m) &= r(2n+1) \end{aligned} \tag{A-3}$$

where $m = 0, 1... N–1$.

Combining equation (A-2) with the definition of the DFT in equation (A-1) leads to:

$$R(k) = R2(k) + \exp\left(- \mathrm{j} \frac{2\pi}{2N} k \right) R1(k) \tag{A-4}$$

where $R2(k)$ and $R1(k)$ are the DFTs of $r2(m)$ and $r1(m)$ respectively.

Equation (A-4) shows that the spectrum $R(k)$ of a signal $r(n)$ can be derived from spectrums $R2(k)$ and $R1(k)$, which correspond to its even and odd parts $r2(m)$ and $r1(m)$ respectively. Usually, the exponential terms are referred to as "*twiddle factors*".

Now, let us construct a complex array $c(n)$, with $r2(n)$ as the real part and $r1(n)$ as the imaginary part:

$$c(n) = r2(n) + \mathrm{j} r1(n) \tag{A-5}$$

where $n = 0, 1... N–1$.

We will show that it is possible to calculate the spectrum $R(k)$ of the $2N$ real-valued array $r(n)$ from the N-point spectrum $C(k)$ of the complex array $c(n)$. In our system, $C(k)$ results from a 1024-point complex FFT.

First, we need to be reminded of two properties of the DFT:

- Linearity: $a\,x1(n) + b\,x2(n) \Leftrightarrow a\,X1(k) + b\,X2(k)$
- For real time functions $x(n)$, the real part of $X(k)$ is an even function and its imaginary part is an odd function: $X(k) = X^*(-k)$, where X^* stands for the complex conjugate of X.

The second property means that the N-point spectrum $X(k)$ is completely characterized when it is calculated on half of the fundamental interval, i.e., k ranging from 0 to $(N/2) - 1$.

According to the DFT linearity, the first half of the spectrum, i.e., k from 0 to $(N/2) - 1$, $C(k)$ can be written as:

$$\begin{aligned} C(k) &= C_r(k) + \mathrm{j}\,C_i(k) \\ &= DFT\big(c(n)\big) = DFT\big(r2(n) + \mathrm{j}\,r1(n)\big) = DFT\big(r2(n)\big) + \mathrm{j}\,DFT\big(r1(n)\big) \\ &= \big[R2_r(k) + \mathrm{j}\,R2_i(k)\big] + \mathrm{j}\big[R1_r(k) + \mathrm{j}\,R1_i(k)\big] \\ &= \big[R2_r(k) - R1_i(k)\big] + \mathrm{j}\big[R2_i(k) + R1_r(k)\big] \end{aligned} \tag{A-6}$$

where $C_r(k)$, $R2_r(k)$, $R1_r(k)$, $C_i(k)$, $R2_i(k)$ and $R1_i(k)$ are respectively the real and imaginary parts of $C(k)$, $R2(k)$ and $R1(k)$.

Then, rewriting the second property as $X(N-1-k) = X^*(k)$ and applying it to $R2(k)$ and $R1(k)$, second half of $C(k)$, i.e., elements $N/2$ to $N-1$, can be expressed as a function of elements 0 to $(N/2) - 1$ of the spectrums $R2(k)$ and $R1(k)$:

$$\begin{aligned} C(N-1-k) &= C_r(N-1-k) + \mathrm{j}\,C_i(N-1-k) \\ &= \big[R2_r(N-1-k) + \mathrm{j}\,R2_i(N-1-k)\big] + \mathrm{j}\big[R1_r(N-1-k) + \mathrm{j}\,R1_i(N-1-k)\big] \\ &= \big[R2_r(k) - \mathrm{j}\,R2_i(k)\big] + \mathrm{j}\big[R1_r(k) - \mathrm{j}\,R1_i(k)\big] \\ &= \big[R2_r(k) + R1_i(k)\big] - \mathrm{j}\big[R2_i(k) - R1_r(k)\big] \end{aligned} \tag{A-7}$$

for reversed indexing and index k ranging from 0 to $(N/2) - 1$.

Equations (A-6) and (A-7) allow the first halves of spectrums $R2(k)$ and $R1(k)$ to be expressed as:

$$\begin{aligned} R2(k) &= R2_r(k) + \mathrm{j}R2_i(k) = \frac{1}{2}\Big\{\big[C_r(k) + C_r(N-1-k)\big] + \mathrm{j}\big[C_i(k) - C_i(N-1-k)\big]\Big\} \\ &= \frac{1}{2}\Big\{\big[C_r(k) + \mathrm{j}C_i(k)\big] + \big[C_r(N-1-k) - \mathrm{j}C_i(N-1-k)\big]\Big\} \\ &= \frac{1}{2}\Big\{C(k) + C^*(N-1-k)\Big\} \end{aligned} \tag{A-8}$$

$$\begin{aligned} R1(k) &= R1_r(k) + \mathrm{j}R1_i(k) = \frac{1}{2}\Big\{\big[C_i(N-1-k) + C_i(k)\big] + \mathrm{j}\big[C_r(N-1-k) - C_r(k)\big]\Big\} \\ &= \frac{1}{2}\Big\{\mathrm{j}\big[\big(C_r(N-1-k) - \mathrm{j}C_i(N-1-k)\big)\big] - \mathrm{j}\big[C_r(k) + \mathrm{j}C_i(k)\big]\Big\} \\ &= \frac{\mathrm{j}}{2}\Big\{C^*(N-1-k) - C(k)\Big\} \end{aligned} \tag{A-9}$$

where $C^*(N-1-k)$ is the complex conjugate of $C(N-1-k)$ and k spans from 0 to $(N/2) - 1$.

Inserting equations (A-8) and (A-9) into equation (A-4) leads to the first quarter of the spectrum $R(k)$, i.e., elements 0 to $(N/2) - 1$:

$$R(k) = \frac{1}{2}\Big\{C^*(N-1-k) + C(k)\Big\} - \frac{\mathrm{j}}{2}\exp\left(-\mathrm{j}\frac{2\pi}{2N}k\right)\Big\{C(k) - C^*(N-1-k)\Big\} \tag{A-10}$$

Since the spectrums $R2(k)$ and $R1(k)$ correspond to real signals:

$$R2(N-1-k) = R2^*(k) = \frac{1}{2}\left\{ C^*(k) + C(N-1-k)\right\} \tag{A-11}$$

$$R1(N-1-k) = R1^*(k) = \frac{\mathrm{j}}{2}\left\{ C^*(k) - C(N-1-k)\right\} \tag{A-12}$$

where k ranges from 0 to $(N/2) - 1$, and spectrums $R2(k)$ and $R1(k)$ are from equations (A-8) and (A-9) respectively.

Inserting equations (A-11) and (A-12) into equation (A-4) leads to the second quarter of the spectrum $R(k)$, i.e., elements $N/2$ to $N - 1$ in reversed order:

$$R(N-1-k) = \frac{1}{2}\left\{ C^*(k) + C(N-1-k)\right\} + \frac{\mathrm{j}}{2}\exp\left(-\mathrm{j}\frac{2\pi}{2N}k\right)\left\{ C^*(k) - C(N-1-k)\right\} \tag{A-13}$$

where k spans from 0 to $(N/2) - 1$.

Because real halves of the spectrum $R(k)$ are even and imaginary halves are odd, having evaluated the first half of $R(k)$, elements N to $2N - 1$ can be calculated as follows:

$$R(2N-1-k) = R^*(k) \tag{A-14}$$

where k ranges from 0 to $N - 1$.

We have shown that, for a real signal of $2N$ points represented by the array $r(n)$, it is possible to evaluate its N-point spectrum $R(k)$ from the complex spectrum $C(k)$, which is the spectrum of the complex-valued array $c(n)$, created from $r(n)$. The first two quarters of the $R(k)$ can be calculated from equations (A-10) and (A-13) respectively and its second half from equation (A-14).

The period of the twiddle factors is equal to $2N$ in equations (A-10) and (A-13) and N in the FFT routine. So, it is necessary to create tables with sine and cosine values to calculate the final spectrum.

In the PSB tune measurement system the oversampling ratio is equal to 4, which means that the tune value is located in the first quarter of the spectrum $R(k)$, and only equation (A-10) needs to be calculated. Thus, once the equation is arranged for fast DSP processing **[10],** one spectrum point requires 4 real additions, 8 real subtractions and 8 real multiplications. Since in the DBV96 additions and subtractions are executed in parallel with multiplications, about 4000 CPU instructions are necessary to evaluate 512 points, i.e., some 0.2 ms (for a 40 MHz CPU clock and 2 clocks per instruction). For a comparison, the 1024-point complex FFT routine takes about 1.6 ms and a complex 2048-point FFT would take about 3.5 ms.