A MODULAR VME DATA ACQUISITION SYSTEM FOR COUNTER APPLICATIONS AT THE GSI SYNCHROTRON

D. A. Liakin*, T. Hoffmann, P. Forck
Ges. f. Schwerionenforschung, 64291 Darmstadt, Germany
*ITEP, Moscow
e-mail: D.Liakin@gsi.de

Abstract

Particle counters perform the control of beam loss and slowly extracted currents at the heavy ion synchrotron (SIS) at GSI. A new VME/Lynx – PC/Linux based data acquisition system has been developed to combine the operating purposes beam loss measurement, spill analysis, spill structure measurement and matrix switching functionality in one single assembly. A detailed PC-side software description is presented in this paper. The software has been divided into time critical networking and data deploying threads and low or normal priority interface tasks to achieve best system stability. Some new abilities in the fields of data computation and presentation are described. First experiences gained while the commissioning of the system are discussed.

SYSTEM STRUCTURE

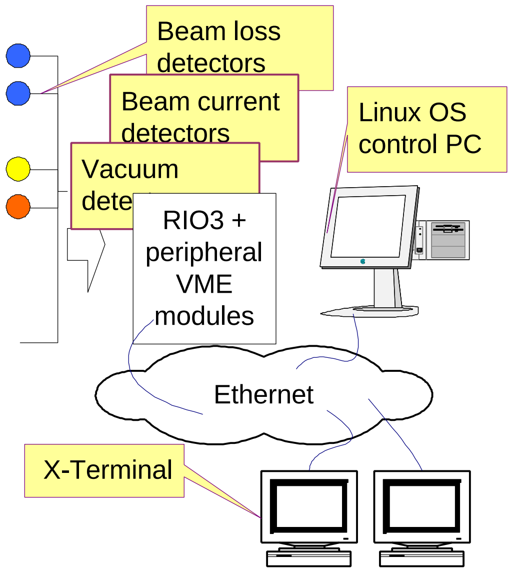

Fig. 1 Scheme of the system structure.

The structure of the new system is shown in Fig. 1. The VME data acquisition modules are controlled by a CES RIO3 processor with an installed Lynx as a real time operating system. The set of acquired data is transferred via 100MBit Ethernet connection to the control PC for being processed by different applications. Detailed description of the used hardware and LynxOS software can be found in [1].

PRINCIPLE OF OPERATION AND DATA STRUCTURE.

All detectors which are used in this system are operating in a pulse counting mode. Even if some detectors initially deal with continuous signals like currents and voltages, finally current-to-frequency or voltage-to-frequency converters are used. A high dynamic range and linearity are advantages of using modern VME counters[2] compared to other methods of data acquisition. At the same time, high timing resolution can be obtained with matched signals, therefore a fine time structure of investigated processes is also available.

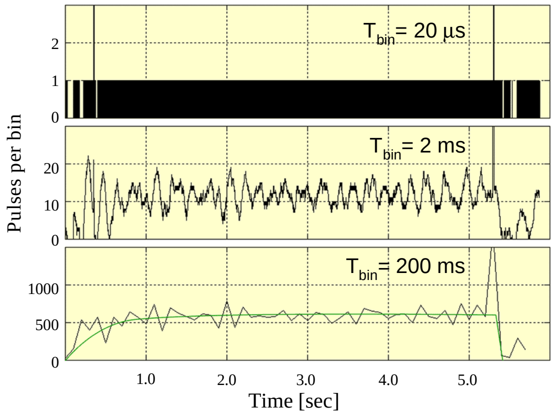

Fig. 2. The beam current in amount of pulses is shown as a function of time. The time unit is 20µs at the top, 2ms in the middle and 200ms at the bottom.

The sampling frequency may be changed according to the aims of different applications. As an example the beam current transformer signal is shown in Fig. 2. At the top one can see the full bandwidth case. In this situation the sampling frequency of 50 kHz is higher then the incoming pulse rate and only one pulse may be acquired within this 20µs interval. At the bottom the low bandwidth case is shown. Low bandwidth is acceptable for the most applications except those where special analysis is required. Spectral parameter of the Fig. 2 top signal are shown in Fig.3. A Fourier spectrum and a spectrogram[3] were prepared to investigate the fine structure of this signal. The Fourier spectrum shows a presence of a number of harmonics of a 1kHz internally generated frequency. In contrast to a simple spectrum the spectrogram illustrates the possibility to detect a more complex non-periodic system behaviour. A current-to-frequency converter is used to convert measured current values to the pulse sequence shown in Fig.2. This ‘floating’ pulse frequency cannot be detected just by Fourier spectrum observation. In addition the spectrogram gives information about the signal nature – the ‘floating’ frequency signal is clear visible in Fig. 3.

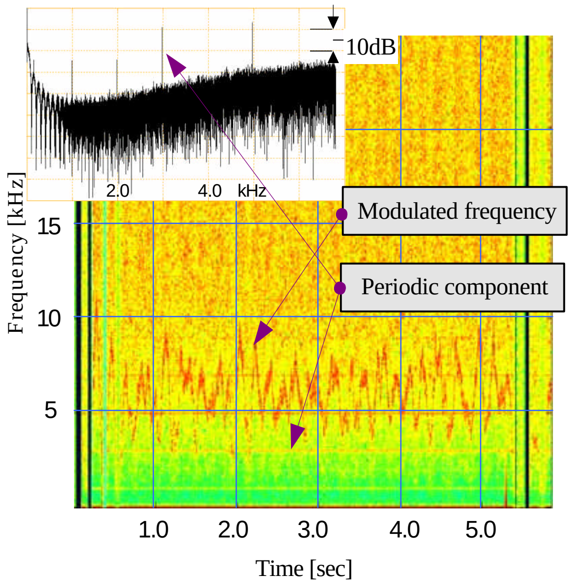

Fig. 3. The spectrum (on left top corner) and a spectrogram of signal shown in Fig. 2.

In this case the detected ‘details’ do not belong to the input signal, but to the internal transformer frequency. They reflect features of the used electronic devices. The incoming signal has a spectrum limited with a frequency about 100 Hz, so the original data is easily filtered from the complex mixture of frequencies shown in Fig. 3.

DATA DISTRIBUTION AND PROCESS SYNCHRONIZATION

An extendable set of applications may be used for online data acquisition control and evaluation on the Linux PC. A special data redistribution and process synchronization mechanism has been developed to provide the possibility to use the same data within different applications simultaneously. Due to the large data amounts which have to be transmitted and due to all applications which are running on the same PC, it is not reasonable to transfer the data to each application separately and to use multiple sockets for the data reception. Instead of this, the experimental data is routed to a single task which is not equipped with complex visualisation possibilities or high quality data evaluation algorithms. It provides a high reliable input data stream processing and an initial data manipulation. This hidden task consists of a high priority thread which is working in very close interaction with the Linux operation system. Fig. 4. shows the main structure of the data distribution. Incoming data passes through numerous network layers provided by the network equipment and OS software to the preliminary data stream interpreter. A high priority interpreter thread analyses incoming data and interprets the type and amount of data. If this data is suitable for evaluation, analysis or presentation it is then accommodated in a dedicated shared memory segment. Corresponding event and semaphore mechanisms are used to grant and synchronize the access to the stored data from other applications. The allowed access time is limited due to reliability requirements. Thus the performance of the applications using shared objects are protected for being slowed down. Usually only the required part of the data set is copied to their internal memory for future evaluation.

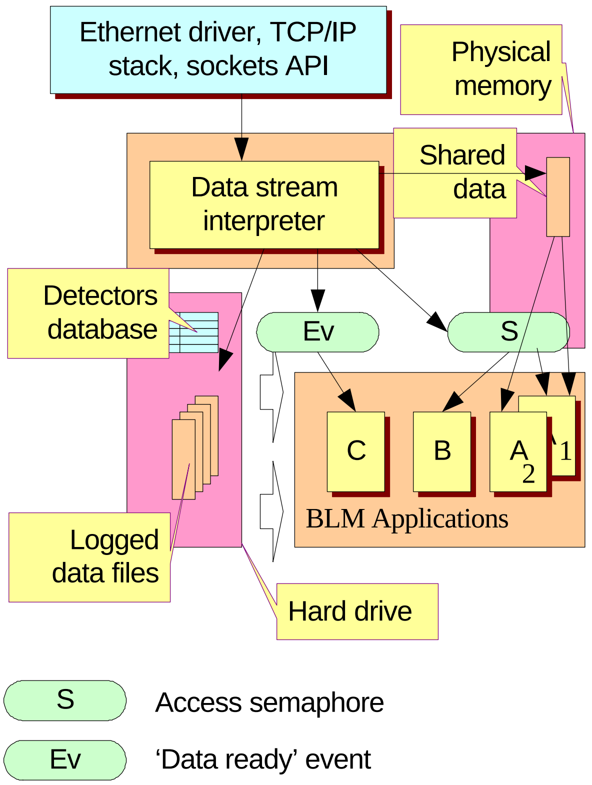

Fig. 4. A special task is used to accept and interpret the input data stream to redistribute data between different applications.

DATA TRANSMISSION

The hardware is divided into the VME front-end and the PC back-end. Currently the remote VME processor has a direct 100Mbit connection to the control PC. It includes the risk of losing a part of the data due to the public character of this network media. Priority was put on safe program operation instead of an absolute errorless data transfer. The transceiver does not verify the correct data transmission - if the network is busy and the internal buffer is full the next set of data will not be sent and data will be lost. The advantage of this situation is that limited front-end processor resources will never be locked by incoming data. From the other side, the receiver has no mechanism to force the transmitter to repeat the last transfer, so only local error handling operations on the control PC are possible and are required to handle data transport problems.

DATA PRESENTATION

Several types of presentation tools are implemented at the moment. A table form data presentation has been realised to display the counter end values after each cycle. This kind of visualisation (Fig. 5a) has proofed its value and is compatible with the GSI crossbar system. Data logging graphs of selected detectors are suitable for an all area cycle by cycle beam loss and current monitoring (Fig. 5b). A highly resolved spill structure analysis tool delivers detailed information (Fig. 5c) within a single cycle. This type of data acquisition may be used in offline and online mode and is successfully used with e.g. beam loss observation.

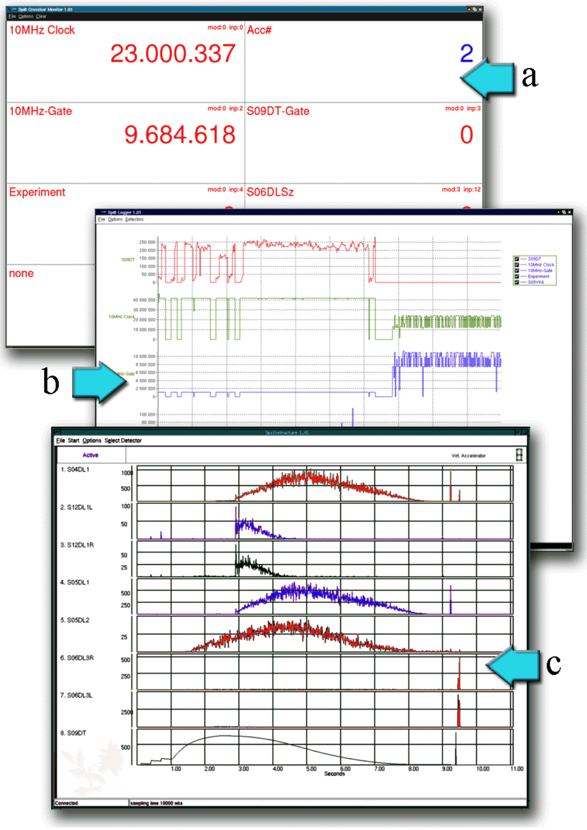

Fig. 5. Examples of application layouts. The crossbar application (a), the history logging and detailed process structure presentation (c) are shown.

SYSTEM PERFORMANCE

Currently four VME multi-channel scaler modules SIS3801, 32 channels each, are used. Every scaler input can be enabled or disabled independently. As default a sampling rate of 1 kHz is used with all 128 channels enabled. For high resolution measurements sampling rates of up to 500 kHz may be set. For this case the amount of active channels has to be reduced. A slow VME bus read/write cycle (1µs for a single 32 bit data read operation) of the CES RIO3 processor is limiting the performance.

MEMORY LEAK PROBLEM

Due to the permanent operation the undesirable memory leak problem gains in importance. Special attention has been paid to detect its presence and then to investigate sources and methods to avoid it. As a first step all applications and subroutines were checked for memory leaks. It turned out that it is difficult to observe the memory distribution in multitasking multithread systems like Linux or Window. The special Borland Kylix[4] memory reallocation mechanism also makes this task uncomfortable.

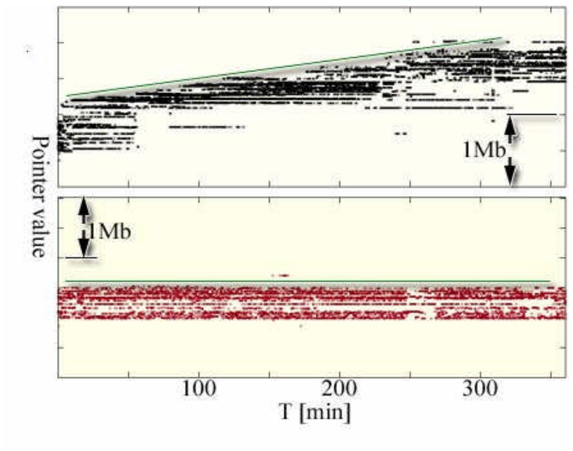

Fig. 6. The probe variable allocation pointer value behavior before (on top) and after optimization.

A simple dynamic variable pointer value was recorded over a long time period to check memory usage in our case. The result is shown in Fig. 6. Though the source of this memory leakage was still undefined after this operation while at the same time the leak itself was obvious. Advanced investigations showed that a huge amount of memory was occupied by processes of allocating and de-allocating memory space somewhere inside of a graphical library. Some leaks were found after detailed observation of the library source code. Others disappeared after changing the state of some objects from dynamic to static. As a result of all leakage corrections the pointer value seems to be stable (Fig. 6, bottom). We have found no reasons to build special tools to control precisely the memory allocation. Instead of this, the probe variable is being observed for restarting the program by itself in case of a significant process memory growth.

REFERENCES

[1] T. Hoffmann, D. A. Liakin and P. Forck "A Fast VME Data Acquisition System for Spill Analysis and Beam Loss Measurement," 10th BEAM INSTRUMENTATION WORKSHOP 2002, Upton, N.Y. (USA), 2002, pp. 329-336.

[2] http://www.struck.de/

[3] Oppenheim, A.V., and R.W. Schafer, "Discrete-Time Signal Processing", Prentice-Hall, Englewood Cliffs, N.J., 1989, pp. 713-718.

[4] http://www.borland.com/