A HARDWARE AND SOFTWARE OVERVIEW ON THE NEW BTF TRANSVERSE PROFILE MONITOR

B.Buonomo, C. Di Giulio, L.G. Foggetta†, INFN – Laboratori Nazionali di Frascati, Frascati, Italy
P.Valente, INFN – Sezione di Roma, Rome, Italy

Abstract

In the last 11 years, the Beam-Test Facility (BTF) of the DAFNE accelerator complex, in the Frascati laboratory, has gained an important role in the EU infrastructures devoted to the development of particle detectors. The facility can provide electrons and positrons, tuning at runtime different beam parameters: energy (from about 50 MeV up to 750 MeV for e- and 540 MeV for e+), intensity (from single particle up to 1010/bunch) and pulse length (in the range 1.5–40 ns). The bunch delivery rate is up to 49 Hz (depending on the operations of the DAFNE collider) and the beam spot and divergence can be adjusted, down to sub-mm sizes and 2 mrad (downstream of the vacuum beam-pipe exit window), matching the user needs. In this paper we describe the new implementation of the secondary BTF beam transverse monitor systems based on ADVACAM FitPIX® Kit detectors, operating in bus synchronization mode externally timed to the BTF beam. Our software layout includes a data producer, a live-data display consumer, and a MEMCACHED caching server. This configuration offers to BTF users a fast and easy approach to the transverse diagnostics data using TCP/IP calls to MEMCACHED, with a user-friendly software integration of virtually any DAQ system. The possibility of sharing mixed data structures (user-generated and BTF diagnostics) allows to completely avoid the complexity of hardware synchronization of different DAQ systems.

THE DAΦNE BEAM TEST FACILITY (BTF)

The BTF (Beam Test Facility) is part of the DAΦNE accelerator complex: it is composed of a transfer line driven by a pulsed magnet allowing the diversion of electrons or positrons, usually injected into to the DAΦNE damping ring, from the high intensity LINAC towards a fully equipped experimental hall. The facility can provide runtime tuneable electrons and positrons beams in a defined range of different parameters. The beam energy can be selected from about 50 MeV up to 750 MeV, for electrons, and 540 MeV for positrons. In dedicated mode (DAΦNE off), the beam pulse length can also be adjusted in 0.5 steps from 1.5 to 40 ns. The delivery rate is depending on the DAΦNE injection frequency (25 or 50 Hz) with a duty cycle also changing according to the DAΦNE injection status, up to 49 bunches/s. Two major modes of operations are possible, depending on the user needs: high and low intensity. In the high intensity mode the LINAC beam is directly steered in the BTF hall with a fixed energy (i.e. the LINAC one, fixed to 510 MeV during the collider operations) and with reduced capability in multiplicity selection (typically from 1010 down to 104 particles/bunch). In the low intensity mode a step Copper target, allowing the selection of three different radiation lengths (1.7, 2 or 2.3 X0), is inserted in the first portion of the BTF line for intercepting the beam: this produces a secondary beam with a continuous full-span energy (from LINAC energy down to 50 MeV) and multiplicity (down to single particle/bunch), according to the setting of a 43° selecting dipole and of two sets of horizontal and vertical collimators. The typical momentum band is well below 1% down to 50 MeV. A pulsed dipole magnet at the end of the LINAC allows alternating the beam between the DAΦNE damping ring and the test beam area, thus keeping a pretty high BTF duty cycle, assuring an average of at least 20 bunches/s during the injection in DAΦNE, when BTF operates in the low intensity regime.

BTF PIXEL DETECTOR LAYOUT

A general overview of the BTF detectors and a description of the related software can be found here [1, 2].

Detector beam-testing generally require a fast transverse beam imaging during runtime, with some pre-analysis capability and, as in the BTF case, with a stable, reproducible and well-known response in the full range of energy, multiplicity and transverse dimensions of the particle beam. In addition, taking into account the BTF heavy work-cycle (beam is generally allocated in one-week slots for a minimum of 25 up to 40 weeks/years, including BTF-dedicated shifts), the equipment has to be reliable, robust, and with an easy maintenance, in order to guarantee full readiness and high-availability for the users purposes. For the same reasons, the related low-level software has to comply with these demanding requirements: not only very high reliability, but also easy and efficient integration in the vast variety of user acquisition software codes. Concerning the high level software, we decided to have the possibility of a fully featured runtime data display using LabVIEW®.

ADVACAM Silicon Pixel Detector

In [1], we described the setup based on a MEDIPIX-like silicon pixel detector with ADVACAM FitPIX® Kit electronics, at first made routinely available to the users via the basic version of the provided software, PIXET®, implemented in the BTF timing and virtual machine sub-systems. This solution well accomplished data acquisition and data-logging tasks (customizable thanks to the possibility of python scripting), but we decided to move to a completely custom low level C-code to better match our facility

†luca.foggetta@lnf.infn.it

ISBN 978-3-95450-177-9

and user requirements of a run-time data retrieval and handling of the pixel data (sharing with heterogeneous acquisition and storage systems, data display, basic analysis and beam parameters extraction, etc.), capable of sustaining the full BTF repetition rate of 50 Hz.

FitPIX is a Medipix/Timepix detector interface with readout speed of up to 90 frames per second for single detector layer, in particular our Timepix sensors have a square pixel of 55 µm side arranged in a 256×256 pixels matrix (side length about 15 mm) for a square sensitive area of about 2 cm² and 300 µm of thickness.

Multiple FitPIX devices can be used essentially for two purposes: forming a particle tracker or as multilayer imager. The integration of a multiple sensors can be done essentially with two methods: the first by attaching up to four Timepix detector layers (single gap of 3.6mm, maximum overall longitudinal length about 20mm); the second possibility, allowing an arbitrary gap between each sensor plane, is to connect more FitPIX devices in daisy chain, with an external bus for the synchronization and master/slave trigger signals. In both cases the electronics communicates with a computer via the USB 2.0 interface for configuration, control and data retrieval [3].

BTF FITPIX Detector Usage

In order to have a completely arbitrary access to live data at the maximum frame-rate within the BTF duty cycle, we have implemented software architecture with a typical producer-consumer layout, implementing data caching on MEMCACHED(MC) server, to allow more than one different consumers at the same time (see below). This has been possible exploiting all the available programming solutions: starting with the python scripting capability of the provided software kit, down to low-level programming, which required some interaction with ADVACAM for a full exploitation of the Linux libraries. We have a fully working version of the software for the single FitPIX, the multidetector-single FitPIX, and more FitPIX devices in daisy chain. As a by-product, the same software architecture allowed us to have a similar implementation for a GEMPIX tracker [4], a four sensors detector with a software configuration similar to the four stacked layers in a single FitPIX.

We initially started with profiling a single FitPIX in different proprietary software configurations (bundle PIXETPRO software with Python scripting capabilities) with external (50Hz) trigger, and single frame acquisition vs. multiple frame acquisition with acquisition repetitions. As long as the fired pixel multiplicity is less than 10 (e.g. in a cosmic run, or in the single-particle BTF beam regime, defined as low data regime), the amount of data stays well below one MTU. In the following Tab. 1 we summarize the PIXETPRO software timing profile with a BTF external trigger at 50Hz: timing responses are reported for both Windows7 (PIXET WIN) running on a dual core, 3GHz, 8GB RAM PC with MC calls. In the PIXETPRO software, we explored the possibility to export data via Python scripting in the Windows environment (PIXET WIN Py) using the python-memcached package. In our test conditions, we were able to cope with the continuous frame rate only in multiple frame mode readout, but without actually sending the data on MC (again, via Python scripting).

In those tests, we have also measured a minimum baseline ~26ms delay before getting the first frame in each new acquisition due to DAC setup, thus degrading a little the maximum achievable rate in multi-frame acquisition during repetitions, as clearly visible in the values for single vs multiple frame acquisitions in Tab. 1. Due to the not negligible impact of this delay and in order to accomplish full rate data retrieval we had then to develop custom software with runtime extraction of each frame in multi-frame acquisition.

Table 1: Time Profiling PIXET Software with Python Scripting on MC

CODE (OS)	MC	Trigger	Trigger Type	Frame time [s]	N frames	N Rep	Acq. Rate [Hz]
PIXET(WIN) Py	Y	50 Hz	HW_Start	1.00E-05	1000	10	23.26
PIXET(WIN) Py	Y	50 Hz	HW_Start	1.00E-05	1	1000	24.61
PIXET(WIN) Py	N	50 Hz	HW_Start	1.00E-05	1000	10	48.48
PIXET(WIN) Py	N	50 Hz	HW_Start	1.00E-05	1	1000	24.90

PRODUCER SOFTWARE LAYOUT

We achieved the best performance of our code by using C/C++ compiled code in an Ubuntu 14.04LTS environment, using the multiple frame acquisition, and by interfacing to a modified API library, in strong collaboration with the ADVACAM software engineers. For a single FitPIX, a multiple frame acquisition is managed by a single thread handling the FitPIX USB interrupt. After each triggered frame a (intra-thread) function diverts the frame data on the library internal memory, calling also a call-back function. This process is automatic in multiple frame acquisition, so we limit to 1000 frames for each repetition in order to keep the memory usage below 2GB per FitPIX. The call-back function is programmable by the user code and it is used for managing the frame data in the user space. Our choice was to get directly the frame data at each interrupt: as measured in Table 3, this dump time to the user memory (USB_N, where N is the FitPIX device) used by the call-back function is normally distributed with an average value of 60µs, so we are perfectly in time for each frame interrupt (data are for double FitPIX configuration, slightly better values are obtained for single FitPIX). In order to measure

ISBN 978-3-95450-177-9

the performance the custom software, for each of the FitPIX configurations, can also perform the following tasks:

- Set the MC pushing attributes for each FitPIX
- Detect the configuration and setup the active FitPIX, especially in defining the type of triggering and synchronization for more FitPIX (if any)
- Create one multiple frame acquisition thread for each active FitPIX. Each thread is sensitive to an internal interrupt for each new available frame, releasing a fast call-back function in order to download frame data
- The call-back function extracts each frame in the multi-frame acquisition, and picks up a nanosecond timer value for the frame interrupt
- Acquire frame data and header, prepares and compress the data payload
- Export processed data on Memcached server

The software starts by command line, where some parameters overwrite the configuration file, to maintain the possibility of simple batch procedure.

Table 2: Time Profiling BTF Compiled Code

Trigger	Trigger FitPIX	Frame time [s]	N frames	Rep	MC	Sparse file	Acq. Rate [Hz]
50 Hz	HW_Start	1.00E-05	1000	10	N	N	**49.97**
50 Hz	HW_Start	1.00E-05	1	1000	N	N	**25.03**
50 Hz	HW_Start	1.00E-05	1000	10	Y	N	**31.00**
50 Hz	HW_Start	1.00E-05	1	1000	Y	N	**24.92**
50 Hz	HW_Start	1.00E-05	1000	10	Y	Y	**49.98**
50 Hz	HW_Start	1.00E-05	1	1000	Y	Y	**25.21**
Auto	PXC_TRG_NO	1.00E-05	1000	10	Y	Y	**41.80**
Auto	PXC_TRG_NO	1.00E-05	1	1000	Y	Y	**26.40**
Auto	PXC_TRG_NO	1.00E-05	1000	10	Y	Y	**78.64**
Auto	PXC_TRG_NO	1.00E-05	1	1000	Y	Y	**38.60**

Processed Data Structure

We have optimized the data structure in order to deliver the minimum necessary information for each frame in the different use-cases. We have then implemented three possible data structures:

- array = send all the matrix data (65536 pixel value data, indirect pixel indexing),
- sparse mode 2dim array = a variable size bi-dimensional array [2,N]. Each slice is the couple of pixel data: pixel number and pixel value
- ultra sparse mode array = each array element stores the over threshold pixel number

All of these cases are headed by the following values:

- Case and FitPIX configuration identifier;
- Time tag (resolution 0.1 ms);
- number of fired pixel;
- frame number;

The data types are in dependence of the used FitPIX configuration: for single layer FitPIX (as imaging detector) all of the values are unsigned short int, with the header expressed in 6 array elements. For multiple-layers FitPIX we use unsigned int, especially for low data regime, with five header elements (still six for compatibility, one being spare). The FitPIX configuration file and data are tagged with the serial identifier of the chip, unique id for a multiple FitPIX configuration.

MEMCACHED

MEMCACHED(MC) [5] is an Ethernet based in-memory key-value store for arbitrary data (strings, objects). In the BTF network environment, it is installed in native version on a virtual machine, running Linux (2.6.18-308.11.1.el5xen quad-cored), with 2GB RAM.

The producer software pushes the processed FitPIX data in these key-value couples on the MC server, where the key name has the FitPIX detector unique identifier, protecting so the source. The producer and the consumer C/C++ codes use standard MC calls provided by the POSIX, thread-safe libmemcached [6] library. For the consumer software, written in LabVIEW, we use our custom MC API's. This choice has been driven by the high reliability obtained in a heavy data load environments of the DAΦNE Control System, BTF [1, 2] and the very good performance obtained within the !CHAOS project [7, 8], where this configuration has been tested via multiple, concurrent producer and consumer calls.

Overall Timing Result Discussion

We profiled the call-back from its interrupt-starting to the data pushing on MC (Table 3 and 4), for a double FitPIX configuration both for the two data regime. Normally, we stay within 1ms for low data one, as we need. Table 2 is a summary for the overall acquisition rate of our software, where the MC column tells if we are pushing (or Not) to MEMCACHED and the sparse file column tells if we

ISBN 978-3-95450-177-9

push the full matrix (sparse file equal to N) or we perform shrinking below one MTU per frame. Again is clear the bottleneck of the MC pushing in full matrix regime, however still within or close to the BTF Ethernet band-width. Another important achievement has been the excellent stability of the producer, running without any problem for months, while providing data to different (BTF or users) consumers.

Table 3: Two Detector, Low Data, Timing Profile

Low	USB_0	USB_1	MC_0	MC_1
Mean[s]	5.9E-05	5.9E-05	8.62E-04	8.9E-04
Std[s]	1.7E-05	1.7E-05	1.79E-04	1.5E-04

Table 4: Two Detector, Full Matrix, Timing Profile

Full	USB_0	USB_1	MC_0	MC_1
Mean[s]	5.8E-05	6.0E-05	2.23E-02	2.33E-02
Std[s]	1.6E-05	1.5E-05	2.09E-03	1.37E-03

OVERVIEW ON CONSUMER SOFTWARE

LabVIEW and C/Root Example Code

We have developed a LabVIEW runtime display of FitPIX data and extracted beam parameters display (such as 3-d transverse image, beam centroid and Gaussian fit data), working both as shot by shot (instantaneous) and in cumulative mode.

In addition, we have developed a combined C/Root code, released freely on [9], that fetches MC keys and integrates ROOT library to save ROOT trees, both in single and in multiple FitPIX configurations. This software makes a wide use of the information encapsulated in the data header, also allowing data integrity checking, e.g. by comparing the frame number vs. frame time. This is a simple comparison but very useful when acquiring multiple FitPIX devices: after getting the MC Key, it gives the chance to discard aged, non-synchronous data. This code is provided as an example code for FitPIX integration in users DAQ software.

Table 5: Two Detector, Low Data Regime, Consumer Timing

Frame	Max Delay [s]	framerate [Hz]	Rejected
100000	0.010	49.98	0
100000	0.005	49.95	1
100000	0.002	49.95	6
100000	0.001	49.63	486

Data Synchronization with Users DAQ

Thanks to the usage of the MC functions, the integration in any user code is now just a matter of few plain C code include and calls. Allowing the BTF users such an easy integration in any, heterogeneous DAQ software, already during the first phases of the beam-time, was indeed one of the major objectives of this work.

From the point of view of consumer (C code) data fetching, Table 5, we show that the consumer software rejection function senses a single event jitter delay of 5ms (the minimum time-tag difference between two synchronized FitPIX MC keys) after 100000 calls, mainly due to producer thread time jitter. This guarantees a complete software data alignment within the user DAQ cycle, even for the data coming for two FitPIX devices in the BTF typical tracking setup (shown in Fig. 1). Users can easily implement this integration after having set NTP synchronization on their PC, before performing one of these actions: fetching the MC keys in a separate database for a delayed offline data-matching; implementing our example code in their own DAQ cycle, when providing a trigger signal to the FitPIXs; or, if the DAQ cycle is fast enough, even without any hardware synchronization, just matching the time of the two data streams (DAQ and FitPIX).

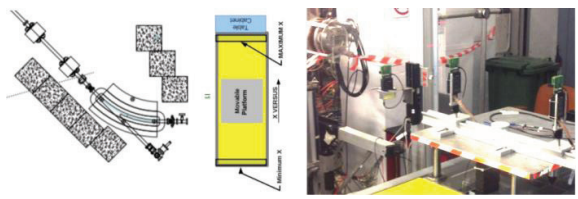

Figure 1: Three FitPIX layout in BTF.

ACKNOWLEDGEMENT

This work is supported by the Horizon 2020 project AIDA-2020, GA no. 654168.

We would like to thank Daniel Tureček and Pavel Soukup, ADVACAM, for the fruitful experience sharing in low-level libraries and hardware development. We are grateful also, among our users, especially to Francesco Renga and Giovanni Tassielli: their requirements are constantly pushing our facility for improvement.

REFERENCES

[1] L.G. Foggetta, B. Buonomo, and P. Valente, "Evolution of Diagnostics and Services of the DAÎ¦NE Beam Test Facility", in *Proc. 6th International Particle Accelerator Conference*, Richmond, VA, USA, 2015, paper MOPHA049, pp. 904-906.

[2] P. Valente et al, "Frascati Beam-Test Facility (BTF) High Resolution Beam Spot Diagnostics", in *International Beam Instrumentation Conference (IBIC16),* Barcelona, Spain, 2016, paper MOPG65, this conference.

[3] V. Kraus et al, "FITPix — fast interface for Timepix pixel detectors", Journal of Instrumentation, Volume 6, January 2011.

[4] F.Murtas et al, "Applications of triple GEM detectors beyond particle and nuclear physics", Journal of Instrumentation, Volume 9, January 2014.

[5] https://memcached.org/

[6] http://libmemcached.org/

[7] http://chaos.infn.it/

[8] A. Stecchi et al., "!CHAOS Status and Evolution", in *Proc. 6th International Particle Accelerator Conference*, Richmond, VA, USA, 2015, paper MOPHA046, pp. 894-896

[9] https://svn.lnf.infn.it/filedetails.php?repname=btf_science&path=%2Ftrunk%2FMC_btf%2FMC_read_tracker%2FreadMC_tracker_Example.cc

ISBN 978-3-95450-177-9