Fermilab/AD/TEV
Beams-doc-860-v33
August 16, 2005
Version 33

Tevatron Beam Position Monitor Upgrade Software Specifications for Data Acquisition

Margaret Votava, Luciano Piccoli, Dehong Zhang
Fermilab, Computing Division, CEPA

Brian Hendricks
Fermilab, Accelerator Division, Accelerator Controls Department

Jim Steimel
Fermilab, Accelerator Division, Tevatron Department

Abstract

This document contains the specifications for the BPM upgrade data acquisition software. Expected operating modes and interactions to the BPM hardware are described. Data structures for communication with the online software via ACNET are defined. Calibration and diagnostics procedures are also specified.

Table of Content

1 Overview

In compliance with the existing Accelerator Controls software architecture, the software pieces surrounding the BPM hardware are divided into 3 layers: the data acquisition (DA) software running on the individual front-end computers in the readout crates in the service buildings (usually referred to as “houses”), user applications running on analysis nodes (Windows or Linux), and online software running on a central server (VAX) and a few DAQ engines (Sun) providing the primary (but not exclusive) bridge between user applications and the DA software. Shown in Figure 1 are the overall software structure and the various layers involved with the BPM upgrade project. This document will focus on the bottom layer, the DA software.

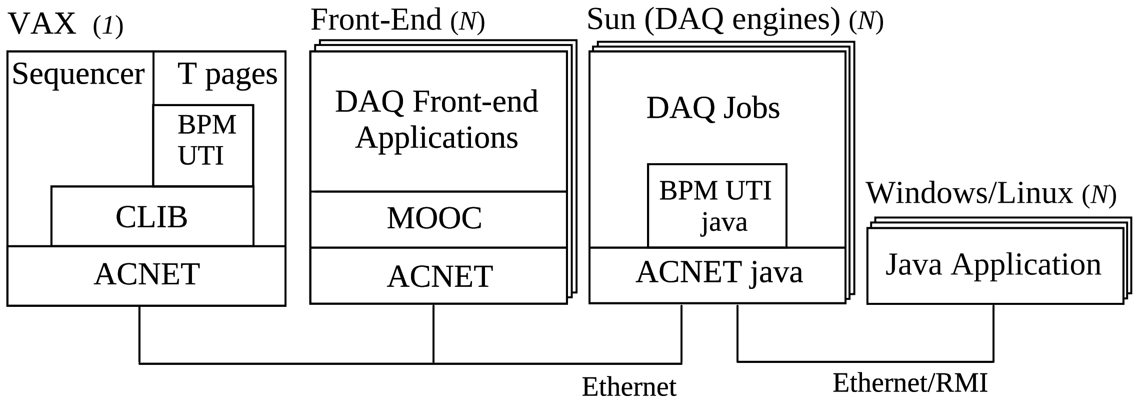


Figure 1 – Overall software architecture

In the remaining part of this section, we will briefly outline the required measurement types, the hardware configuration and the requirements on the DA software. In section 2, we will describe the controls, configurations and data buffering within one front-end crate. In section 3 we will define the interface to the online software. In sections 4 through 7 we will elaborate on calibration, debugging, alarms and monitoring.

1.1 Measurement Types

The different types of measurements to be handled by the upgraded Tevatron BPM system are summarized as follows:

- Closed Orbit Measurements. The closed orbit is a measurement of the beam position and intensity with the betatron motion, around 20 kHz as seen by the pick- ups, averaged out but including the ~ 60 Hz synchrotron motion component.
- Average Closed Orbit Measurements. The average closed orbit is a measurement of the beam position and intensity with both the betatron motion and synchrotron motion averaged out. Compare to the closed orbit measurement, this

measurement should average over a longer time in order to eliminate the beam motion due to synchrotron oscillations. In the Tevatron the synchrotron frequency is about 30 Hz at its lowest, so the beam position should be averaged over 3 synchrotron periods or for about 0.1 seconds.

- Injection Closed Orbit Measurement. This is a measurement of the closed orbit immediately after injection of beam into the Tevatron.
- Injection First Turn Measurement. This is a measurement of the beam position and intensity on the first pass after it enters the Tevatron from the injection beam lines.
- Turn By Turn (TBT) Measurement. The TBT is a sequence of 8192 beam positions and intensities measured for 8192 consecutive turns, one measurement per turn at the same relative phase. For this measurement all of the BPMs are synchronized to begin collecting data on the same turn, and there should be no missing turns.
- Asynchronous Injection Measurement. This is a failsafe mode for the situation when beam is not injected into the correct bucket, and beam fails to make many revolutions in the Tevatron, and none of the previous modes will be able to produce a reliable measurement.

It is important to note that for all variations of the TBT measurement, all BPMs should be triggered off the same bunch passing with about the same time advances.

1.2 Hardware Configuration

Based on experience from the Recycler BPM system, the upgraded Tevatron BPM readout system is designed to be a VME crate holding a crate controller with a PMCUCD daughter board, a Timing Generator Fanout (TGF) board and up to 6 ADC boards and their corresponding filter boards. There are 27 houses around the Tevatron ring, each having one BPM crate to handle up to 12 BPMs. Illustrated in Figure 2 is the hardware organization within a VME crate (see document #1070 for hardware specifications). The makes and main functions of the boards are briefly described in the following.

The TGF Board is a re-design based on a previous version being used on the Recycler ring. Its functionalities include:

- phase lock to the 53.1 MHz Tevatron RF frequency to generate the 74.3466 MHz clock signals for up to 8 ADC boards;
- decode the events transmitted through the Tevatron TCLK system to provide advanced arming signals to the crate controller so that the hardware can be configured in time for different measurement types, or present pre-defined requests for data transfers from the crate controller to the online software;
- decode the events transmitted through the Tevatron TVBS system to derive accurate timing/triggering signals to synchronize up to 8 ADC boards with respect to the different beam arrivals;

- phase lock to the 53.1 MHz Tevatron RF frequency to generate continuous, or train-of-7, diagnostic TTL pulses at 53.1 MHz on the VME backplane to feed the filter boards; The train-of-7 pulses are triggered by the Tevatron turn makers with an adjustable time delay;
- control the switching of the diagnostic signal relays on up to 8 filter boards so that the 53.1 MHz diagnostic pulses can be directed to the ADCs, or back to the BPM pick-ups, or both.

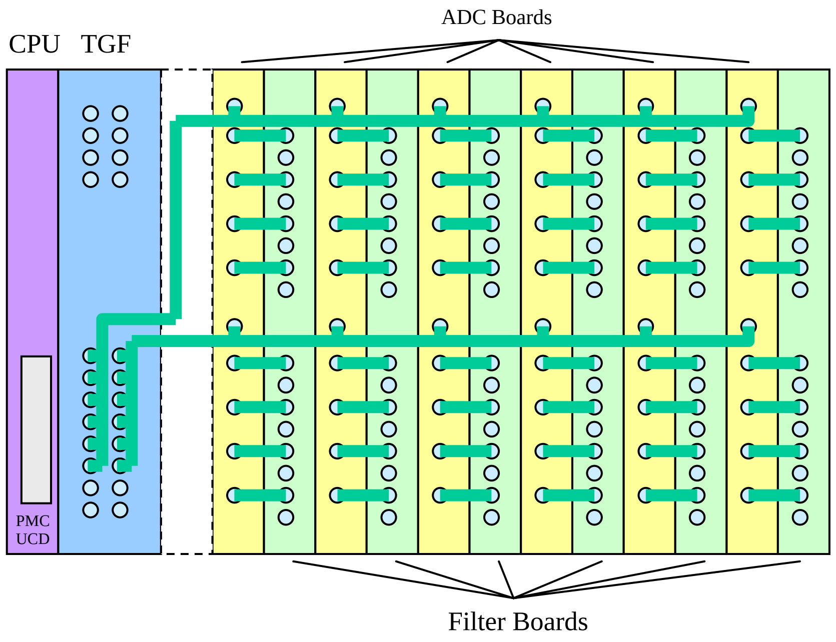


Figure 2 – Elements within a crate

The Filter Boards are also designed at Fermilab. They perform analog filtering and attenuation so that the 53.1 MHz components of the proton and antiproton signals have approximately the same dynamic range when they reach the ADCs. The filter boards also have mechanic relays to switch the diagnostic signals to different paths as mentioned above.

The ADC Boards have been chosen to be model ECDF-GC814-FV-A from the EchoTek Cooperation. Each ADC board has 8 channels that are paired together to share 4 channel-pair delays. Naturally, channels 1 and 2 are connected to digitize the proton signals from the A and B plates of a BPM pick-up with a certain channel-pair delay so that the digitization, and digital down-conversion and filtering if set up, will start just before the proton bunches arrive at this pick-up, while channels 3 and 4 digitize the antiproton signals from the same pick-up with a different channel-pair delay to account for the different arrival time of the antiproton bunches. Channels 4 through 8 repeat this pattern. The digitized outputs can be raw ADC counts or digitally down-converted and filtered to extract the strengths of the 53.1 MHz components. In the later case, the output of each channel is represented by two components: a real (I for in-phase) and an imaginary (Q for quadrature) part. The I and Q components from the 4 channels of a single BPM pick-up are then used by the DA software to calculate the proton and antiproton positions and intensities.

The Crate Controller has been decided to be model MVME2400 from Motorola. It is responsible for the communications between the online software and the TGF, filter and EchoTek ADC boards. All controls and data transfers to/from the boards are performed by the crate controller, with the exception of the controls for the diagnostic signal control relays on the filter boards. Their controls are done by the crate controller via the TGF board.

The PMCUCD daughter board is from TechnoBox Inc. It also decodes the TCLK signal and triggers the standard ACNET/MOOC software applications accordingly. Since this board belongs to the Accelerator Controls infrastructure, it will not be discussed any further in this document.

1.3 Requirements on the DA Software

In order to comply with the existing Accelerator Controls software architecture and minimize the time needed for development and debugging, the DA software must meet the following requirements:

- As of now, different measurement types require different setups in the EchoTek boards. The front-end DA software is responsible for setting up the various boards according to user requests coming from the online software.
- All communication with the front-ends is via ACNET devices. This includes data readout as well as setting acquisition and readout parameters. Internal diagnostics, however, do not have this constraint.
- Data acquisition happens asynchronously from data readout, i.e., the BPMs can be configured to take data continuously on certain triggers, but the data may not be read out until later. Not all data that is collected is read out. This implies that the DA must have buffers for each measurement type and manage readout requests from the online software.
- "Event assembly" is done by the online software, not by the DA. Therefore a given BPM crate does not need to have any knowledge about any other BPM crate(s). The data sent by the crates will however include information for having the data synchronized by the online software. That information includes time stamps and turn counts from the TGF boards.
- The front-ends should detect state changes via the state devices (in the old system, the sequencer would notify the crate controller of changes). This will help to reduce the complexity of the sequencer, allow for faster builds and testing of software changes, and push the knowledge of BPM behavior to the BPM themselves.
- The front ends software will use the Recycler BPM software and EchoTek driver as a starting point for the software design.
- The crate controller should run VxWorks.

2 Data Acquisition

The front-end data acquisition system is located between the online software and the BPM digitizing hardware (see Figure 3). Any access to information and controls on the electronics boards will pass through the front-end processor. The information includes beam positioning data, calibration data and diagnostics data among other configuration parameters.

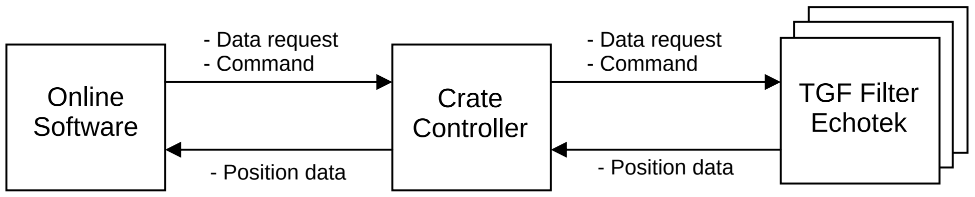


Figure 3 – Data acquisition scheme

The communication between the online software and the front-end processor is ACNET/MOOC based, while the communication between the front-end processor and other electronics boards (right side on Figure 3) happens through the VME backplane.

During a measurement, the digitized (and digitally down-converted) data is first stored in memory on the EchoTek boards and then transferred to the crate controller until all the data has been collected. The data is saved in a set of buffers in the crate controller for later readouts. The depth and number of logical buffers that reside on the crate controller is defined in document #903. The actual physical implementation of the crate controller buffers will be described in the front-end software design document.

2.1 Arm Signals

The preparation for data acquisition for the EchoTek boards is signaled by an arm event. An arm event may be a TCLK event detected by the TGF, or a state device transition. A state device transition is generated when a registered device changes its status, and a central service broadcasts the transition to predefined listeners. As an example of state device transition, the front-end can be waiting on the device V:CLDRST to switch to the "squeeze" state.

Table 1 describes a list of relevant TCLK events, most of which are for arming purposes while a few, e.g. TCLK $75, can be used to signal the crate controller to transmit certain types of data.

State transitions are not as reliable as TCLK events. For that reason they are not intended to be used as arming events.

Table 1 – TCLK and MIBS events associated with the BPM system

Event	Description	Comment
$71 TCLK	Tev:BPM Prepare for beam	Presently issued 0.01 secs after $4D. Will be issued once before shot setup.
$73 TCLK	Tev:BPM High field	Issued half way up the energy ramp. Was used to change alarm limits for BPMs. Not needed for the new system.
$74 TCLK	Tev:BPM Low field	Issued after a $4D. Was used to change alarm limits for BPMs. Not needed for new system.
$75 TCLK	Tev:BPM Write profile memory	Trigger times programmed into a CAMAC 070 Card. Used to collect profiles up the ramp.
$C1 TCLK	Tev:Tevatron reset for collider operations	Enabled by sequencer and triggered by $41. Used to reset display frame pointer.
$C2 TCLK	Tev:Prepare to accelerate collider beams.	Referenced to $41 event. Used to reset the profile frame pointer.
$78 TCLK	Tev:BPM Write display frame	Variable time and trigger. Typically set to 2.71 secs after a $4D.
$40 TCLK	Tev:Reset for pbar injection @150GeV	Start of Pbar injection from MI->Tev. Occurs about 2.7 seconds before the actual injection.
$5B TCLK	Collider pbar beam transfer trigger from MI to Tevatron	TCLK echo of MIBS $7B. MI->Tev transfer of pbars happens on the MIBS $7B which is about 2.7 seconds after the $40.
$4D TCLK	Tev:Reset proton injection @150Gev	Start of proton injection from MI->Tev. Occurs about 2.7 seconds before the actual injection.
$5C TCLK	Collider proton beam transfer trigger from MI to Tevatron	TCLK echo of MIBS $7C. MI->Tev transfer or protons happens on the MIBS $7C which is about 2.7 seconds after the $4D.
$47 TCLK	Tev:Beam has been aborted	TCLK event generated when the BSSB pulls the Tevatron abort.
$4B TCLK	Tev:Abort clean up	TCLK event used to trigger the Tev abort kickers and intentionally remove beam. (Every time beam is removed from the Tevatron either a $47 or $4B is issued.)

2.2 Timing and Trigger Signals

The DA system will be configured to operate in one of two basic running modes: repetitive and one-shot. The former will be used for closed orbit measurements, while the later will be used for the various types of TBT measurements.

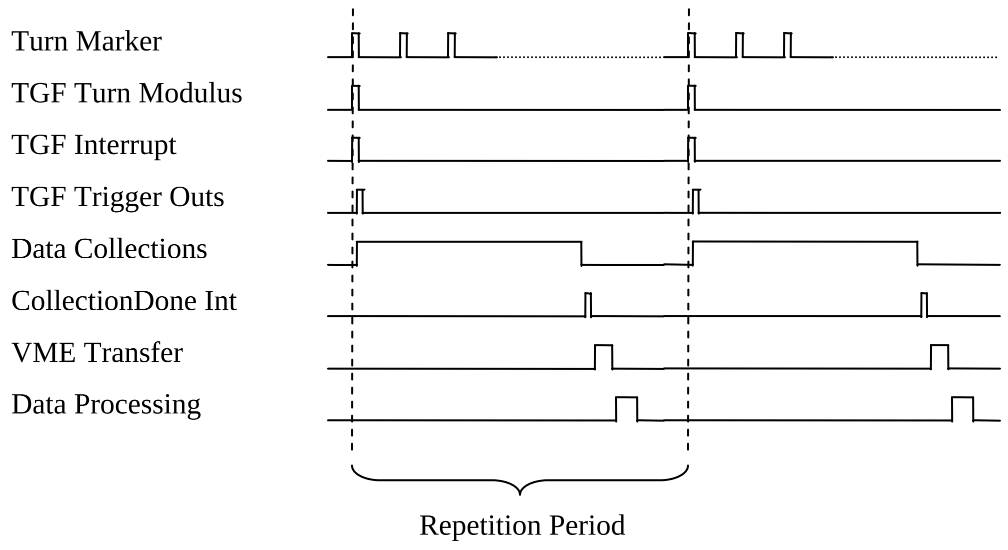


Figure 4 – Timing for closed orbit mode

For closed-orbit measurements, the front-ends do not need to be synchronized to the same turns/bunch passing, they only need to cover the same length in time. In this mode, the TGF simply decimates the Tevatron revolution/turn markers, generates an interrupt to inform the crate controller of a pending data acquisition, applies appropriate time delays to account for the different bunch arrivals at different BPM pick-ups so that data from all the BPMs have the same phase, and generates trigger (SYNC) signals for the EchoTek boards. When the pre-set number of samples (burst counts) has been collected, the first EchoTek board will issue a collection-done interrupt to notify the crate controller for data transfer. The timing relations are visualized in Figure 4.

For TBT measurements, the front-ends in all the houses will need to be synchronized to the same specific turns, so that the measurements they make correspond to the same turns. This is achieved by setting the TGF to wait for a particular "start_event". Upon the occurrence of this "start_event", the TGF generates an interrupt to inform the crate controller of a pending data acquisition, then repeats the following steps for the desired number of turns: wait for the next "turn_event", be it the turn markers, apply appropriate time delays and generate trigger signals for the EchoTek boards. When the pre-set number of triggers has been received, the first EchoTek board will issue a trigger-counter interrupt. These timing relations can be seen in Figure 5. The descriptions of the relevant TVBS events are listed in Table 2.

It should be noted that in a given running state of the Tevatron, the time delay from the Tevatron turn marker (TVBS $AA) as seen by the TGF till the actual bunch passing is fixed at each BPM location. In order to ensure that the ADCs are triggered at the same time relative to the actual bunch passing of the same turn, 4 layers of delays are implemented. These delays are global delay, house delay, EchoTek board delay and channel pair delay. For different states of the Tevatron, especially if the beam is injected to different buckets, these delays may need to be adjusted if the same timing relationship is required.

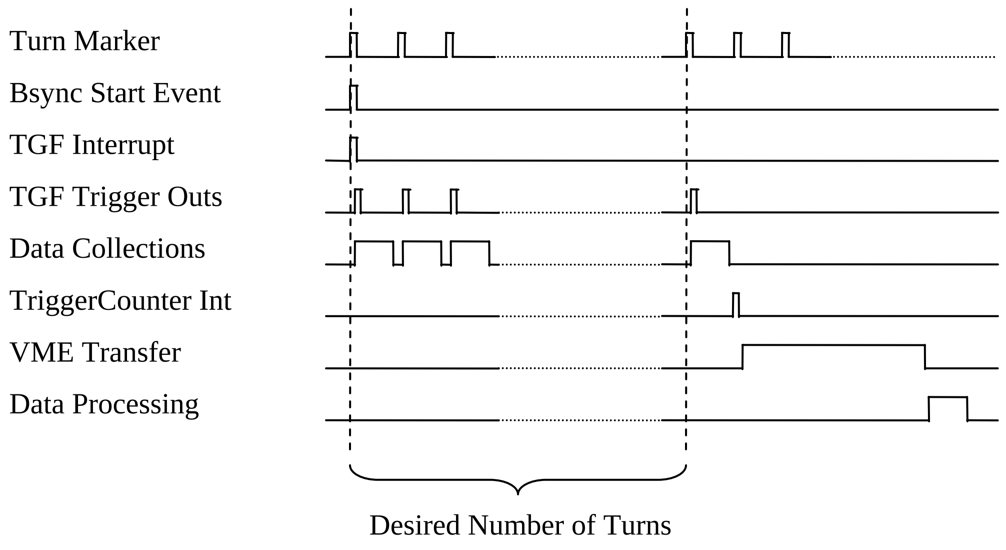


Figure 5 – Timing for TBT modes

Table 2 – TVBS events associated with the BPM system

Event	Description	Comment
$AA TVBS	Tev:Beam	
$7C TVBS	MI:Ini MI->Tev proton coli	Tevatron Beam Sync event at injection. Derived from the MIBS $7C.
$DA TVBS	Trigger TBT data collection.	Tevatron Beam Sync used to trigger TBT data collection. Used to synchronize all BPMs to the same turn.

2.3 Data Acquisition Modes

Corresponding to the various measurement types it needs to handle, the BPM data acquisition system has a few different operation modes. These modes are mutually exclusive due to the need for specific configurations of the triggering and filtering in the EchoTek boards for a specific mode. These modes are:

- Idle
- Closed Orbit
- Turn by Turn
- First Turn
- Asynchronous Injection

- Calibration
- Diagnostic

where Idle and Closed Orbit are two default modes. When not in any of the other modes, which default mode the DA system should enter is dictated by the Tevatron state device V:TEVBPM – Tevatron BPM Mode. The OOC (??) will set this device to “closed orbit” after some fixed delay from a proton injection event (TCLK $4D), will change it to “no beam” after a Tevatron clean-up event (TCLK $4B) to signal the Idle mode.

These modes are elaborated in the following.

2.3.1 Idle Mode

The Idle mode is the “no-function” mode of the BPM operation that occurs when there is no beam in the Tevatron. During this mode, all operational data buffers are frozen, and the TGF should be set to wait for TCLK $4D. Upon the detection of TCLK $4D, the TGF interrupts the crate controller and the system should transit into the First Turn mode.

2.3.2 Closed Orbit Mode

This is the default mode of operation when there is beam in the Tevatron and the data acquisition system is not requested to take any other types of measurements.

2.3.2.1 EchoTek Setup

In closed orbit mode, the EchoTek boards will be set up to operate in the split I-Q mode. Two of the 4 sub-channels of a GC4016 chip process the in-phase and quadrature signals in parallel, and their outputs are merged into one stream of interlaced I’s and Q’s, 16 bits each. The CIC is set to decimate by 1024 making the total decimation rate to be 4096. This corresponds to a PFIR output data rate of about 18 kHz while the re-sampler is by-passed. Both CFIR and PFIR are set to take rolling averages, with 9 and 23 tabs respectively. Combining the CIC, CFIR and PFIR, the overall 3 dB bandwidth is about 700 Hz (see Figure 6), which is dominated by the PFIR.

The GC4016 has a latency of 7 PFIR output data cycles before any data reaches the memory and one additional PFIR output data cycle delay for every two PFIR taps before the PFIR filter output settles. For the present configuration, the output stabilizes after 20 output clock cycles. The system is currently set to take 24 PFIR outputs and skip the first 23 to minimize the data volume to be transferred over the VME backplane. The total data acquisition time is about 1.3 ms per trigger.

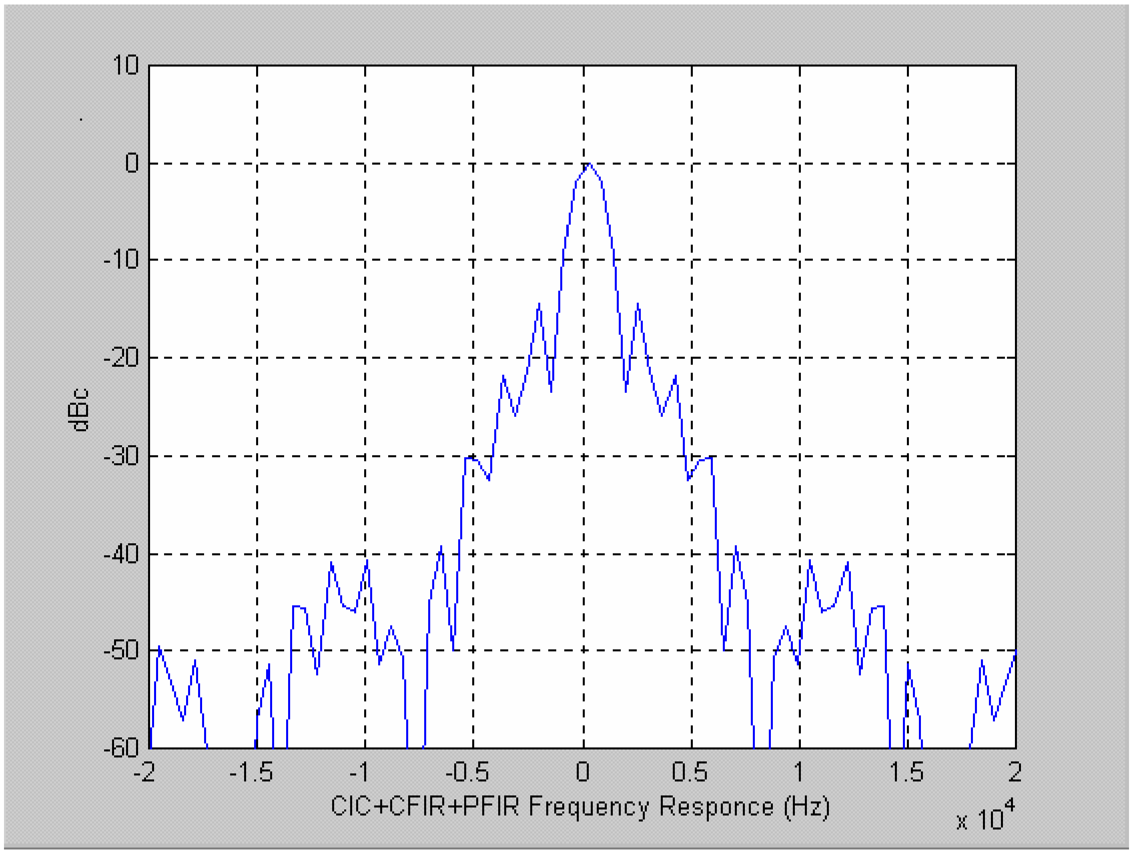


Figure 6 – GC4016 frequency response for the closed orbit mode

2.3.2.2 Triggering

The measurements are triggered at a 500 Hz rate. This rate is sourced by the TGF board through a decimate-by-*N* counter to down-sample the 47 kHz Tevatron turn marker (TVBS $AA). The value of *N*, and thus the trigger rate, can be adjusted through the turn modulus/decimation register on the TGF board. At 500 Hz, the DA system has about 200 us idle time between triggers.

2.3.2.3 Data Processing

After the EchoTek boards have completed data acquisition, the crate controller will read out, through VME single reads, one I-Q pair (the 24th PFIR burst) from each channel, 4 pairs for each BPM. The proton contaminations on the antiproton signal will then be removed from the raw antiproton I's and Q's (A and B plates) according to:

$$I'_{Pb} = I_{Pb} - C_I \times I_P + C_Q \times Q_P$$

$$Q'_{Pb} = Q_{Pb} - C_I \times Q_P - C_Q \times I_P$$

where subscripts *P* and *Pb* denote proton and antiproton signals respectively, C_I and C_Q are the in-phase and quadrature components of the coupling coefficient.

The modulus of each channel (*M*) is then calculated:

$$M = G \times (\sqrt{I \times I + Q \times Q} - M_0)$$

where G and M_0 are the gain and offset of the electronics. Finally the beam positions ($D_{P,Pb}$) and intensities ($S_{P,Pb}$) can be determined according to:

$$D = g \times (M_A - M_B)/(M_A + M_B) - D_M$$

$$S = M_A + M_B$$

where g is a scale factor to convert the unit-less quantity to millimeters (nominally 26 mm), D_M is the mechanical offset that was surveyed relative to the BPM's electrical center before the BPM was installed in the ring.

For each BPM, the 2 positions ($D_{P,Pb}$) and 2 intensities ($S_{P,Pb}$) are put into a data structure together with the 4 raw I and Q pairs (see data structure section). This data structure is then stored in a 1024 point deep circular buffer, the Fast Abort buffer, and propagates to the other buffers.

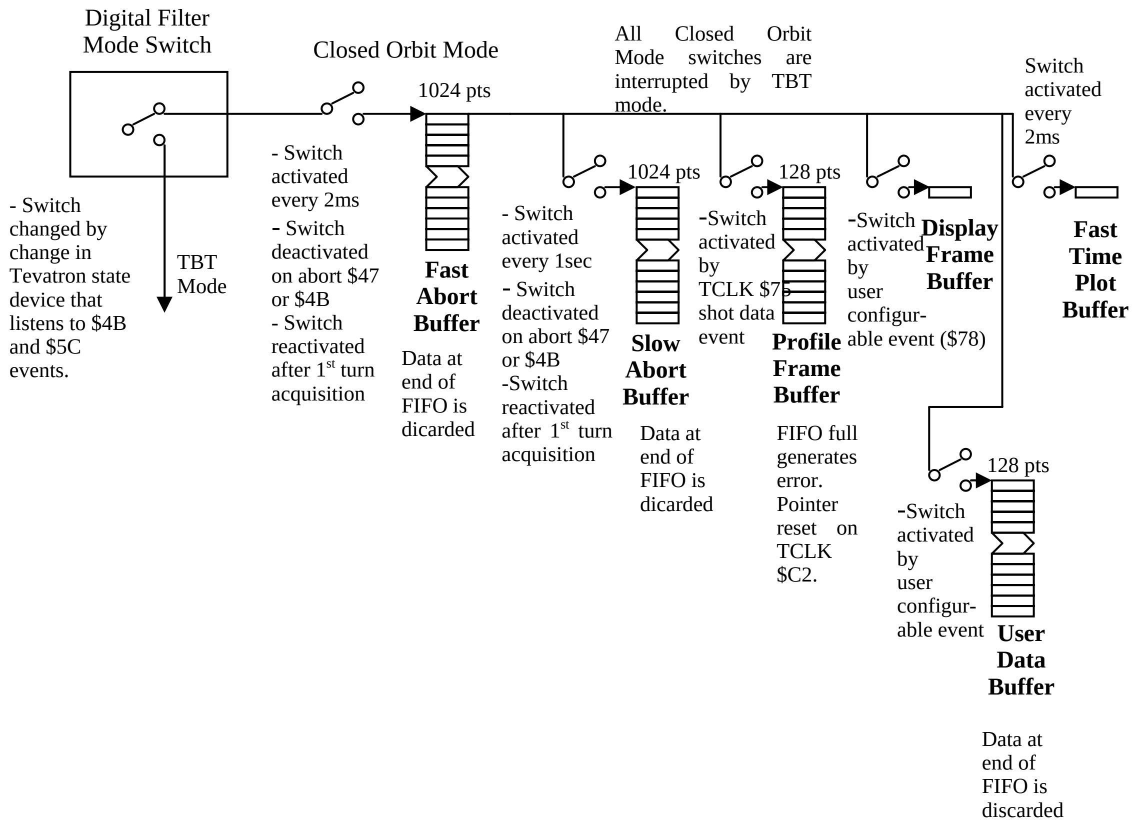
2.3.2.4 Data Buffer Organization

In closed orbit mode, the DA software maintains the following data buffers:

- Fast Abort buffer – circular buffer, 1024 points deep. One element in the fast abort buffer is referred to as a frame**.** The crate controller takes measurements from all BPMs every 2 milliseconds, and puts them into the fast abort buffer. The process stops on Tevatron aborts (TCLK $47 or $4B), and re-starts automatically after the Injection mode.
- Slow Abort buffer – circular buffer, 1024 points deep. Data is copied from the newest frame in the fast abort buffer every 500 readings where the 500 is configurable.
- Profile Frame buffer – FIFO, 128 points deep. Data is copied from the latest frame in the fast abort buffer every time a TCLK $75 (shot data event) is received, i.e. each index in the profile frame corresponds to a particular shot data event. If the profile frame buffer overflows, new values will be discarded and an alarm condition will be flagged -- only one alarm will be sent per crate. If a $75 event arrives when in a mode other than Closed Orbit, this particular profile frame needs to be filled, maybe with a bad status.
- Display Frame buffer – single frame. Data is copied from the latest frame in the fast abort buffer every time a TCLK $78 (display event) is received.
- Fast Time Plot (FTP) buffer(s) – data is copied from the latest frame in the fast abort buffer every 2 milliseconds. There is no depth to this buffer, but since fast time devices plot only single values, it will be a series of devices to plot:
    - A particular BPM's proton position
    - A particular BPM's antiproton position
    - A particular BPM's proton intensity
    - A particular BPM's antiproton intensity

- A particular input's I value
- A particular input's Q value
- Sum signal for each channel (magnitude of A + magnitude of B)

- Snapshot Data request – the most recent frame in the fast abort buffer is copied to the request's buffer.



Digital Filter Mode Switch
Closed Orbit Mode
1024 pts
All Closed Orbit Mode switches are interrupted by TBT mode.
Switch activated every 2ms
- Switch changed by change in Tevatron state device that listens to $4B and $5C events.
TBT Mode
- Switch activated every 2ms
- Switch deactivated on abort $47 or $4B
- Switch reactivated after 1st turn acquisition
Fast Abort Buffer
Data at end of FIFO is dicarded
- Switch activated every 1sec
- Switch deactivated on abort $47 or $4B
-Switch reactivated after 1st turn acquisition
1024 pts
Slow Abort Buffer
Data at end of FIFO is dicarded
-Switch activated by TCLK $75 shot data event
128 pts
Profile Frame Buffer
FIFO full generates error. Pointer reset on TCLK $C2.
-Switch activated by user configur-able event ($78)
Display Frame Buffer
Fast Time Plot Buffer
128 pts
-Switch activated by user configur-able event
User Data Buffer
Data at end of FIFO is discarded

Figure 7 – Buffer diagram for closed orbit measurements.

When a TCLK $47 (abort) or TCLK $4B (no beam left in machine) is received, the DA system should:

- Freeze data in the slow abort buffers, the profile frame buffer, and the display frame buffer.
- Fill the FTP buffers with a status indicating no beam.
- Acquire a few more measurements and fill the fast abort buffer, then freeze it.
- Change the mode of operation to the Idle mode.

2.3.3 Turn by Turn

During turn-by-turn operation, the BPM system samples the position of the beam at every BPM, at every turn for 8192 turns. Data from these arrays of measurements is analyzed to determine Tevatron lattice information such as beta functions, phase advance, local coupling, etc. Normally, for turn-by-turn measurements, there is only one proton bunch in the machine, and some kicker magnet is fired synchronously with the trigger of the turn-by-turn measurement.

In turn-by-turn mode, the EchoTek boards will also be running in the split I-Q fashion. The total decimation rate is however changed to 16 for a wide bandwidth. The corresponding data output rate is about 4.65 MHz. Both CFIR and PFIR are again set to take rolling averages, but with 11 and 13 tabs now. The combined overall 3 dB bandwidth is about 360 kHz (see Figure 8), which is again dominated by the PFIR.

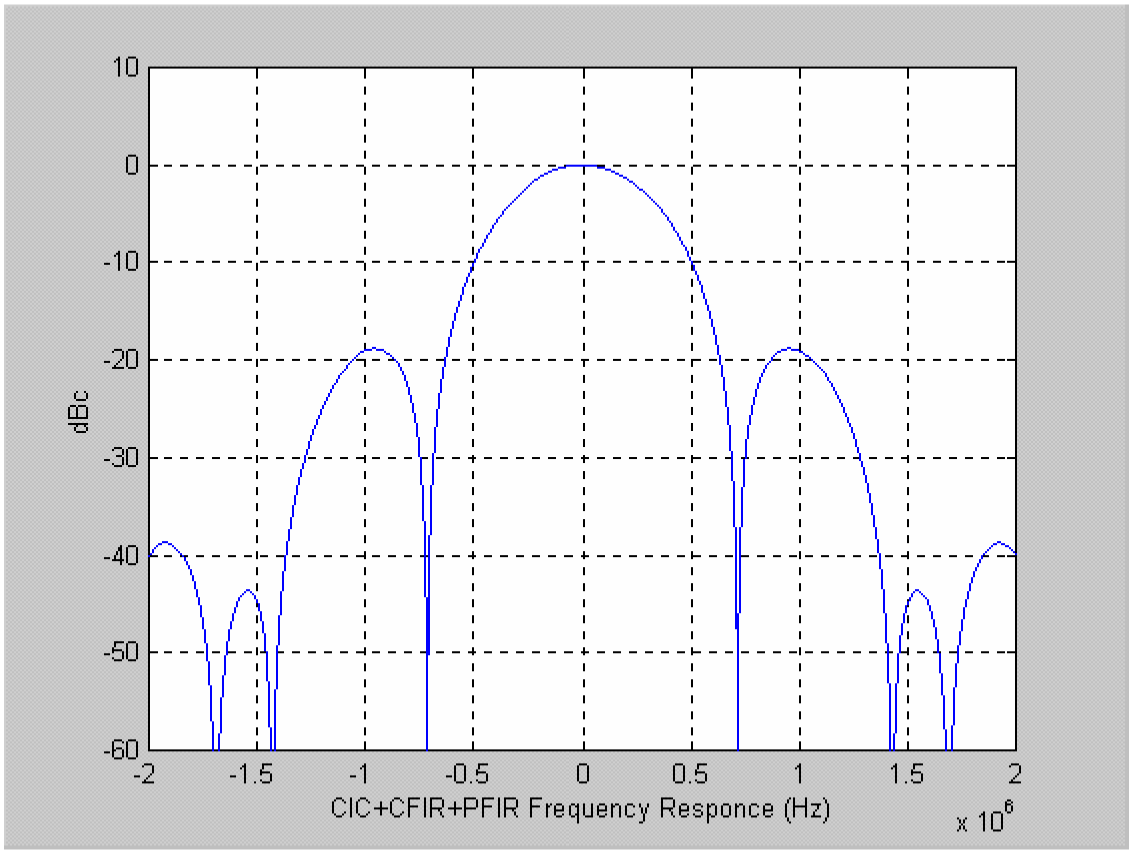


Figure 8 – GC4016 frequency response for the turn-by-turn mode

For this configuration, the output stabilizes after 14 output clock ticks, the system will therefore be set to take 16 PFIR outputs and skip the first 15 for every turn.

The turn-by-turn mode can occur at any user specified arm and start events. For example it can be armed by a TCLK event $77, started by BSYNC event $DA which is derived from the TCLK $77, and triggered by the Tevatron turn markers (BSYNC event $AA). The time advance from the arm event to the start event must be long enough to allow time for the system to change mode before triggering.

Upon the successful completion of a turn-by-turn data acquisition, the data will be transferred to the crate controller through DMA. The positions and intensities of the proton bunch at each BPM location will be calculated in the same fashion as in the closed orbit mode. The data structure will then be stored in a 8192 point deep TBT buffer for later access.

After a turn-by-turn measurement, it is very important that the DA system change to closed orbit mode as soon as possible. Due to the large data volume, there is inevitably a dead time of about ?? ms. Studies are being carried out to see whether: a) the FPGA firmware can be modified to accommodate multiple operation modes in the same time so that there is no need for mode change at all, for example two of the 4 sub-channels run in turn-by-turn mode, while the other 2 sub-channels run in the closed orbit mode; or b) the electronics can run in turn-by-turn mode in a repetitive fashion and the crate controller calculates the closed orbit by averaging, so that a mode change would only involve changing the number of turns to trigger and probably changing the time delays with the TGF.

2.3.4 First Turn

The first turn mode is a special case of the turn-by-turn mode. It is intended to capture the position of the beam at each BPM location as the beam travels its first revolution around the ring immediately after injection, and obtain the closed orbit while the injection bumps are still active. These position information is used to diagnose errors in injection position and angle in order to tune the magnets in the injection beam lines.

Since the injection bump magnets are only activated briefly, it is not feasible to take a one-turn turn-by-turn measurement, then re-configure the electronics and switch to closed orbit mode to catch the injection closed orbit. Instead, the DA system will take a normal 8192-turn turn-by-turn measurement, retrieve the first turn position, and obtain the injection closed orbit by simply averaging over all elements in the injection turn-by-turn buffer that have intensity above a predefined threshold.

The first turn mode is armed if the TGF receives a proton injection event (TCLK $4D) and the Tevatron state device indicates “no beam”. The start event for the TGF is TVBS $7C which corresponds to the kickers being fired for main injector to Tevatron transfer. The TCLK $4D occurs about 2.5 seconds before TVBS $7C, allowing ample time for the DA system to configure the electronics.

The first turn measurement will be stored in the First Turn buffer for later readout.

2.3.5 Asynchronous Injection

In this mode the EchoTek boards should be configured to take 8192 raw data samples for each channel. At a sampling frequency of 74.3466 MHz, these samples can cover around 70 revolutions.

The TGF should be set to trigger at the first Tevatron turn marker after a specific start event. The global (coarse) delay should be used to make sure the 70 revolution window covers the first turn. Fine (short time) delays like house delays, EchoTek board delays and EchoTek channel pair delays can be used but not required.

The raw data can be used online to determine the aggregate time delays for each BPM location through certain pattern recognition. It can also be used offline to calculate the beam positions and intensities so as to help diagnosing the injection beam lines.

2.3.6 Calibration Mode

The calibration mode is basically a normal run mode where data collected is tagged as calibration data. These data specially marked is used on the offline processing for defining the calibration constants to be used by the front-end when calculating the position of the beam.

The user via online software sets the calibration mode. That information is passed to the front-end DAQ system through ACNET variables. All data read out from EchoTek cards are tagged as calibration data, until the calibration mode is disabled.

2.3.7 Diagnostic Mode

The first level of diagnostics for the system involves verifying the signal path of the detectors, cables, filters, and A/D converter. The tools for performing this diagnostic are the diagnostic signal and the filter card relays. The diagnostic signal is a 53MHz TTL waveform that is distributed to every channel on the filter card. There are two filter card relays per channel. These relays can be configured for four possible states: diagnostic signal deactivated, diagnostic signal direct to EchoTek input through attenuator and filter, diagnostic signal direct to detector, and diagnostic signal to both EchoTek input and tunnel. Sending the signal in both directions will reduce its intensity compared to the direct connections.

There are three modes of EchoTek operation required to take advantage of the test signal diagnostics. First, the modules can be set up in raw A/D mode. This mode bypasses the Greychip and routes the A/D data directly into memory. The memory values can be inspected to insure that some reasonable signal is getting into the EchoTek module with the diagnostic signals activated. Second, the modules can be set up in standard closed orbit mode. This mode can be used when the signal is passed through the detector before reaching and EchoTek input. Third, the modules can be set up in a reduced gain closed orbit mode. This mode is required for when the diagnostic signal goes directly through the attenuator and filter and into the EchoTek module. The continuous diagnostic signal will have a much higher fundamental frequency component than the beam, and the standard closed orbit configuration will saturate in the Greychip processing.

There are four different measurements that can be made to verify the operation of a channel. First, isolate the channel from the pickup and the diagnostic signal and verify a zero amplitude result in standard closed orbit configuration. Second, switch the diagnostic signal into the input channel directly in reduced gain closed orbit configuration. Verify proper operation of filter, attenuator, and A/D. Third, activate diagnostic signal into input channel and detector cable to verify operation of A/D with different amplitude signal. This operation also requires the reduced gain closed orbit configuration. Fourth, connect input channel directly to tunnel and connect opposing channel diagnostic switch (channel connected to same BPM plate but opposite side) to tunnel cable only. Verify A/D with different amplitude signal and verify cable, detector, attenuator, and filter chain integrity.

The diagnostic application program should have the ability to control the filter board switches individually. To simplify the configuration to the user, there should be four possible relay configurations: diagnostic off, diagnostic to EchoTek only, diagnostic to tunnel only, diagnostic to EchoTek and tunnel. The application should also allow the tests above to be generated automatically for a single house upon a user request. Only the magnitudes of the different channel measurements need to be recorded. These measurements are compared to baseline measurements for possible variations that are out of tolerance. Those measurements that are out of tolerance by more than 1% in magnitude are flagged as potentially requiring a new calibration. The user also has the option of saving the results as the new baseline.

The next level of diagnostics for the system is monitoring and control of the VME crate. We will take advantage of the RS232 interface on the crate to control and monitor the voltage levels and fan speeds on each of the crates. An Ethernet to RS232 adapter will be utilized as the communication bridge to the internal crate monitoring and control. The adapter will be powered independently from the crate to allow a remote user to power down the crate without losing communication to the adapter. The remote control of the crates will include the ability to perform a SYSRST on the backplane, a SYSFAIL on the backplane, and power down the crate.

A third level of diagnostics for the system is the verification of proper events and triggers. There are four types of events that need to be monitored for diagnostics purposes: state transitions, TCLK events, the TVBS $AA marker, and the other TVBS events. The system needs a means of monitoring the order of critical events that the timing card and processor will react to. The timing card will keep a circular buffer of the last 256 events (TCLK, TVBS $7C & $DA, measurement complete, etc.) that it uses to generate interrupts or triggers, excluding TVBS $AA events. The timing card does not need to record all TCLK events, only the events that it uses to generate interrupts to the processor. The processor, in turn, will have a circular buffer of the last 256 events of interrupts that it receives from the timing card as well as any other events that it will cause a change in system configuration (change in state device V:TEVBPM, request for change into diagnostic mode, etc.). These two buffers will be accessible from the Tev BPM diagnostics application and will be used to verify that the processor is receiving the proper interrupts and that the events are happening in the proper order. The final event diagnostic will be $AA marker verification. This will be done on every turn-by-turn and first turn data acquisition. While triggering the turn-by-turn measurement, the timing card will verify that the $AA markers are being triggered every turn by comparing the time between markers with a divide by 1113 counter clocked by the 53 MHz. If a trigger is missed, the turn number that has a skip is loaded into a buffer. The data user can then reconstruct the proper sequence of data for analysis.

The fourth level of diagnostics is verification of the EchoTek and Greychip setup. The diagnostic application will be able to override the default EchoTek configurations and force it into closed orbit mode, turn-by-turn mode, or raw A/D mode. There will be a means to override the trigger card configuration as well, so that the system can be configured to trigger the EchoTek module on after an arbitrary TCLK event and delay. The software will retrieve data from all active channels of the Greychip, from up to two EchoTek channels. The data from separate Greychip channels will be organized and plotted after each trigger. The application will also have the means for saving channel data for offline analysis, either by saving to a file, or e-mailing the data to a users e-mail account.

The final level of diagnostics is the ability to halt and examine all of the processor data buffers. There will be an option on the diagnostic application page to halt data acquisition into the buffers for the EchoTek channels selected (up to two). The application will then allow comparisons between the processor buffers and any associated ACNET variables that mirror the buffers. Discrepancies will be flagged.

2.4 State Diagram

The crate controller uses the Tevatron state devices for two purposes. It uses V:TEVBPM to determine which mode of operation to return to after a reboot or a change from diagnostic mode. It also fetches information from other state devices to include in the metadata. Please refer to

http://www-bdnew.fnal.gov/tevatron/adcon/tev_states.html

for a detailed description of the Tevatron state diagram. In the old system the sequencer is responsible for informing the BPM system about a TeV state change. The new system should be able to detect such changes by itself, freeing the sequencer from this task.

The state device containing the current Tevatron state is V:CLDRST. The BPM system is interested in some of the possible states assumed by this device. The following is a list of states for V:CLDRST:

1. Proton injection porch
2. Proton injection tune up
3. Reverse injection
4. Inject protons
5. Pbar injection porch
6. Inject pbars
7. Cogging
8. Before ramp
9. Acceleration
10. Flattop
11. Squeeze
12. Remove halo
13. HEP
14. Pause HEP
15. Proton removal
16. Unsqueeze
17. Flattop2
18. Deceleration
19. Extraction porch
20. Extract pbars
21. Reset
22. Recovery
23. Ramping

Besides the device V:CLDRST, other state devices that should be checked by the BPM system are:

- V:TEVBPM – Tevatron BPM mode of operation. Use this device to determine the mode of operation for the BPM system after power-down, reboot, or diagnostics exercise.
  1. Closed orbit
  2. idle
- V:NXBNCH – Next bunch injected into Tevatron (1 - 36). Use this device to set different trigger delays for first turn injection mode if first bunch goes anywhere besides bunch 1.
- V:NXTBKT – Next bucket injected into Tevatron (1 - 1113), for metadata only.
- V:TEVMOD – Tevatron high-level mode
  1. colliding beams
  2. proton only
  3. dry squeeze
  4. ramping
  5. recovery / turn on
  6. off
- V:COALP – Proton coalescing state for meta data only
  1. coalescing off
  2. coalescing on
- V:COALA – Antiproton coalescing state for meta data only
  1. coalescing off
  2. coalescing on
- V:TVBEAM – Particles present in the Tevatron

1. no beam
2. protons
3. pbars
4. protons and pbars

- V:HELIX – Helix state for metadata only
- V:PBKTC – Number of Proton bunches for metadata only
- V:ABKTC – Number of Pbar bunches for metadata only

2.5 Alarms

In order to avoid flooding the operations alarm screen with several repetitious alarm conditions, there will be a single alarm device for a given BPM crate. The device will alarm on any condition that implies that the house is in trouble. One must then go to the diagnostics page to determine the exact source of the problem. Alarm conditions include, but are not limited to:

- Any dead channel
- Power supply problems
- Profile frame buffer overflow
- Timeouts when getting injection data
- Detection of raw signals above or below thresholds (same thresholds across the machine)

2.6 Configuration Parameters

These are some configuration parameters identified that should be used for setting up the BPM software DAQ. Changes to all parameters implemented as ACNET devices will take effect immediately, i.e., they will not be delayed and they will not synchronized across houses.

2.6.1 DAQ Parameters

Description	Scope	Type	Range
Disable automatic triggering of injection mode on TCLK $4D	System	ACNET Operator pages	State device Restore last value
Define the current mode of operation of the BPMs	House	ACNET Operator pages	DA Diagnostics Calibration
The number of fast abort buffers frames that are taken after the abort is received.	System	Compilation	
The minimum intensity threshold needed to be considered real beam	System	ACNET Engineering pages	

Number of samples used in closed orbit “averaging” in normal operation mode	System	Call argument	1, 100
Constants used to compute “averages” in normal operation mode	System	Compilation (need to keep track of versions)	
Number of samples used in closed orbit averaging in injection mode	System	ACNET Engineering pages	2-8k
Constants used to compute “averages” in injection mode	System	Compilation (need to keep track of versions)	
Number of fast abort samples needed for each slow abort sample	System	Compilation	1 - 1024
Buffer Depths Abort Buffers, Turn by Turn, Profile	System	Compilation	
TCLK needed to arm TbT request	System	Compilation/Configuration File (?)	

2.6.2 Timing Parameters

Description	Scope	Type	Range
Delay for trigging after specific TCLK	House	ACNET Engineering pages	msec
	EchoTek module		
	Channel		
Delay for arming after specific TVBS	System	ACNET Engineering pages	msec
TCLK $4D + <value> until hardware modules are switched into TBT mode.	System	Compilation/Configuration File (?)	msec
TBT readout must complete in <value>	System	Compilation	msec

2.6.3 Diagnostic Parameters

Description	Scope	When can Change	Range
Debug Level	House	ACNET Diagnostic pages	
Defines the type of trigger injecting data into the buffer	House	ACNET Diagnostic pages	Accelerator clock External clock
Defines what is the source for the buffer incoming data	House	ACNET Diagnostic pages	BPM single turn, BPM closed orbit
Software self triggering for first turn (ie, not on bsynch)	System	ACNET Operator pages	

3 Interface to Online Software

This section defines how data and commands are exchanged between the front-end DAQ software and the online software. ACNET will be the means of transportation of data and commands between the front-end DA and the online software. Commands are:

- Arm turn by turn – prepare BPMs for turn by turn data taking
- Mode Select (Diagnostics, Calibration, Data Acquisition)
- Enable/Disable injection mode
- Get BPM data

Requests may be made in parallel by disjoint applications, and some mechanism for avoiding conflicts must be designed and implemented.

These types of conflicts can specifically occur with turn by turn arm commands. When the crate controller receives multiple turn by turn arm commands, it will process only the last request received. Any turn by turn request already in progress will be aborted.

BPM data read by the online software will be organized according to the data structures defined subsequent in section 0. Online applications that make use of the old system must be changed to handle new data formats (see document #1060 – Online Software Specifications).

The supported ACNET protocols will be SETDAT, RETDAT and Fast Time Plot (FTP). The snapshot protocol (not to be confused with a BPM snapshot) will not be supported by the front-end DAQ. The front-end must be able to generate FTP data at a rate up to 500 Hz.

3.1 SSDN / ACNET Device Mapping

The following suggested SSDN numbers will be used to map to the appropriate ACNET devices. There is at least one ACNET device associated with each SSDN number.

The SSDN number is an 8 byte field split into 4 2-byte pairs. See the MOOC Front-Ends document for a detailed field description:

http://www-bd.fnal.gov/controls/micro_p/mooc_front_ends.html

For reading BPM data, this project will use the recycler BPM model of object id:

- 0x0021 for BPM retdat/setdat and
- 0x0022 for BPM FTP data
- 0x0026 for setdat
- 0x0042 for BPM FTP data
- Timing and ADC settings not yet defined.

3.1.1 SSDN Mapping for FTP devices

Each channel can have an I and Q value. Data from a pair of channels (i.e. A and B plates) can be manipulated to generate in a corresponding intensity and position. There will be a position and intensity measurement for a horizontal and vertical position for both proton and antiproton data. One EchoTek card contains eight channels, allowing 2 BPM to be read out (e.g. one horizontal and one vertical). A complete house, with 6 EchoTek boards can read out 12 BPMs, each one with 4 channels. The total channels in a house is 48, and each channel has an I and Q value associated.

The following FTP devices are associated with these 48 channels:

`0000/0022/000`**`X`**`/22`**`YY`**

where **`X`** is `0` for I, Q and the sum signal, where I is the first, Q is the second and the sum is the third element on the device; and **`YY`** is the channel number, which can vary from `00` to `2F` (47 decimal).

Position and Intensity values can also be accessed through FTP devices. Each house will provide at most 24 positions and 24 intensities that can be accessed via these SSDN numbers:

`0000/0022/000`**`X`**`/22`**`YY`**

where **`X`** is `1` for proton position and intensity and `2` for pbar information, where position is the first and intensity is the second element on the device; and **`YY`** is the BPM number, which can vary from `00` to `0C` (12 decimal).

The maximum number of FTP devices provided by one house is 72:

1 I/Q/Sum x 48 channels + 2 p/pbar x 12 position/intensity = **72**

3.1.2 SSDN Mapping for RETDAT devices

The RETDAT protocol is used to retrieve most data read out by the BPM system. Through RETDAT it is possible to read the following BPM data:

- Fast abort buffer (array)
- Slow abort buffer (array)
- Profile frame buffer (array)
- Display frame buffer
- Snapshot buffer (first element of the fast abort buffer)
- Average snapshot buffer (10Hz average of fast abort buffer)
- Injection turn-by-turn buffer (array)
- Injection closed orbit frame
- Turn-by-Turn buffer (array)

The BPM data will contain information from all EchoTek cards present in the system; except when handling turn-by-turn requests, which can specify a single BPM. All array buffers above can return any number of frames.

The SSDN for the RETDAT BPM devices are:

`0000/0021/000`**`X`**`/210`**`Y`**

where **`X`** is `0` for raw data (I and Q) or `1` for scaled data (position/intensity); and **`YY`** is the data being read out:

- `0` - Fast abort buffer
- `1` - Slow abort buffer
- `2` - Profile frame buffer
- `3` - Display frame buffer
- `4` - Snapshot buffer
- `5` - Average snapshot buffer
- `6` - Injection turn-by-turn buffer
- `7` - Injection closed orbit buffer
- `8` - Turn-by-turn buffer

3.1.3 SSDN Mapping for SETDAT devices

SETDAT devices are used for setting modes of operation of the BPM system and also to specify the data acquisition and change configuration values. For changing the mode of operation the following SSDN are defined:

`0000/0021/0000/218`**`X`**

where **`X`** is `0` for enabling Turn-by-Turn mode and `1` for turning on system diagnostics. The remaining SSDN are use for:

- DAQ specifications
- System configuration

3.2 Data Structures (Output Date)

The data sent from the front-end DAQ to the BPM library and/or applications is based in the following C data structures. For compatibility, the BPM libraries on the online side will be responsible for extracting subsets from these data, which are required by existing application programs.

3.2.1 Generic Headers

```
typedef struct BPM_TIME {
    ulong timestamp;          /* timestamp in seconds (GMT) */
    ulong nanoseconds;        /* nanoseconds */
}
```

```
typedef struct TRIGGER_INFO {
    long type;                /* Periodic, TCLK */
    to be defined;
}

typedef struct TEVATRON_BPM_HEADER {
    long endian_type;                 /* 0    -> little endian,
                                         else -> big endian */
    long version;                     /* data structure version */
    long status;                      /* overall status: zero = OK */
    BPM_TIME time;                    /* time stamp */
    ulong turn_number;                /* starting turn number */
    ulong num_turns;                  /* number of turns in data */
    double time_in_cycle;             /* starting time in cycle */
    long data_type;                   /* flash/snapshot/profile/TBT/etc */
    TRIGGER_INFO trigger_info;        /* trigger information */
    long data_source;                 /* 0 -> beam,
                                         1 -> calibration system,
                                         2 -> software diag,
                                         3 -> hardware diag */
    long particle_type;               /* 0 -> proton, 1 -> pbar */
    long bunch_type;                  /* 0 -> uncoalesced, 1 -> coalesced */
    long scaled_data;                 /* 0 -> raw data, n -> scaling algorithm */
    long calibration_id;              /* calibration data ID number */
}
```

```
status - returns non zero value if there is some problem at the crate level. The online
side will ignore the data.
time - should be the time stamp of the first data frame.
turn_number - comes from the timing module.

Suggestions of new fields:
algorithm_version: version of the algorithm used for calculating the closed orbit
firmware_version: version of the EchoTek firmware
```

```
typedef struct TEVATRON_BPM_STATE_DATA {
    long machine_state;               /* value of V:CLDRST */
    long helix_state;                 /* value of V:HELIX */
    long num_proton_bunches;          /* value of V:PBKTC */
    long num_pbar_bunches;            /* value of V:ABKTC */
    long bpm_state;                   /* BPM state (not yet defined) */
    unsigned long narrow_gate_mask;   /* narrow gate measurement mask */
}
```

3.2.2 BPM Non Turn By Turn

```
typedef struct TEVATRON_BPM_FRAME_DATA {
    long frame_number;        /* ordinal number in front-end */
    BPM_TIME time;            /* time stamp */
    ulong turn_number;        /* starting turn number */
    double time_in_cycle;     /* starting time in cycle */
    TEVATRON_BPM_STATE_DATA state_data;      /* machine/BPM state information */
    long num_detectors;       /* number of detectors present */
    long status[12];          /* status values */
    float positions[12];      /* position values in mm */
    float intensities[12];    /* intensity values */
}
```

For raw data requests (I and Q), the returning structure is the same, except for the positions and intensities arrays that are replaced by:

```
    long i[24];
    long q[24];
```

Proposed status values:

```
OK = 0
invalid reading = 1 (too little beam intensity?)
alarm level = 3 (if we want alarm limits)
saturated = 5
error = -1 (error reading value (hardware error?))
```

```
unequipped = -2 (channel is not in use)
```

This would be the final structure of the ACNET device for display, snapshot, profile, and flash frames:

```
typedef struct TEVATRON_BPM_ORBIT_DATA {
    TEVATRON_BPM_HEADER       header;
    long num_frames;   /* number of frames returned */
    TEVATRON_BPM_FRAME_DATA frame_data[];
}
```

3.2.3 BPM Time Slice Data

This structure is used for sending an entry (frame) from the BPM buffers.

```
typedef struct TEVATRON_BPM_TIME_SLICE_VALUE {
    long status;                     /* detector status */
    unsigned long milliseconds;      /* milliseconds since first frame */
    float position;                  /* position in mm */
    float intensity;                 /* beam intensity */
}
```

For raw data requests (I and Q), the returning structure is the same, except for the positions and intensities arrays that are replaced by:

```
    long i[2];
    long q[2];
```

```
typedef struct TEVATRON_BPM_TIME_SLICE_DATA {
    TEVATRON_BPM_HEADER header;
    TEVATRON_BPM_STATE_DATA state_data;      /* machine/BPM state information */
    Long num_frames;                 /* number of frames returned */
    TEVATRON_BPM_SLICE_VALUE frame_data[];
}
```

3.2.4 BPM Turn By Turn

```
typedef struct TEVATRON_BPM_TBT_TURN {
    ulong turn_number;        /* turn number */
    float position;           /* position in mm */
    float intensity;          /* beam intensity */
}
```

For raw data requests (I and Q), the returning structure is the same, except for the positions and intensities arrays that are replaced by:

```
    long i;
    long q;
```

This would be the final structure of the ACNET device for turn by turn data:

```
typedef struct TEVATRON_BPM_TBT_DATA {
    TEVATRON_BPM_HEADER       header;
    TEVATRON_BPM_STATE_DATA state_data;      /* machine/BPM state information */
    long status;              /* detector status */
    long num_turns;           /* number of turns returned */
    TEVATRON_BPM_TBT_TURN turn_data[];
}
```

3.2.5 Calibration Constant Data

This structure is used for reading and setting calibration constant data:

```
typedef struct TEVATRON_BPM_CALIBRATION_DATA {
    long calibration_id;             /* calibration ID number */
    long state_value;                /* corresponding BPM state value */
    long num_constants_per_detector; /* number of constants per detector */
    float constants[][12];           /* calibration constants */
}
```

4 Interface to BPM Hardware

The front-end software will access data from the BPM hardware through VME memory mapped buffers. Information about protons and pbars will be available in different addresses, and each type of particle will have at least two streams of information. One type contains the latest position information (for Turn By Turn measurements) while the other has average position information (Closed Orbit measurements).

Additionally, there will be other memory mapped regions used to pass configuration and diagnostics data from the front-end to the hardware and vice-versa.

4.1 Operation of the EchoTek ADC Boards

Each EchoTek board provides 8 receiver channels of 14-bit, 80 MHz (maximum) analog-to-digital conversion and digital processing in a single 6U VME slot. It supports 3 operating modes:

1. Gate Mode – data is collected as long as the external sync signal is active;
2. Trigger & Free Run – data collection begins at the rising edge of the external sync signal, or when the Software Sync Bit is written, and continues until the Trigger Clear bit is written;
3. Trigger & Counted Burst – A preprogrammed number of samples (burst counts) is acquired and processed with each occurrence of an external sync pulse, or writing to the software bit – This is the mode the Tevatron BPM system will be using.

In the trigger modes, the external sync signal can be delayed to each pair of channels (0 & 1, 2 & 3, 4 & 5, 6 & 7). The SYNC_DELAY registers are 12-bit counters that are clocked at the ADC sample rate.

The 8 channels on each board can be independently configured to output one of 3 types of data:

1. Count Data – the on-board FPGA generates a continuous counting sequence and puts the data directly into the memory. This bypasses the whole chain of analog-

to-digital conversion and digital processing, is hence intended for checking the data handling within the board and checking the VME interactions.
2. Raw Data – the ADC counts are stored into the memory directly, bypassing the digital processing. The samples are right justified with the two least significant bits always set to zero. Every two samples are packed together to form a 32-bit word.
3. Receiver Data – the ADC counts are digitally down-converted, decimated and filtered to output an interlaced sequence of 24-bit I's and Q's. The 24-bit I's and Q's can be either truncated to 16-bit then packed into 32-bit words each consisting one I and the corresponding Q, or directly output in 32-bit words.

The memory for each channel is a 128 K x 32 bits SRAM operating in a FIFO fashion.

The procedure to initialize the boards is as follows:

1. read in all the setup files (.ini and .ch files), parse the files then create an array of "ghSetup" structures in the crate controller's memory by calling "ecdr814gcReadSetup" in the startup script;
2. install a driver for each board by calling "ecdr814gcInstall";
3. open the driver by calling "open";
4. allocate data buffers in the crate controller's memory;
5. set the buffer pointers in the driver to the buffers in the crate controller's memory by calling "ecdr814gcSetBufferBrd";
6. copy the desired setup from the "ghSetup" array to all the boards by calling "ecdr814gcCopySetupAll";
7. set the channel-pair delays by calling "ecdr814gcIoctlCopysyncDelayAll";
8. program all the Gray chips by calling "ecdr814gcProgramGrayAll";
9. set up the daq conditions by calling "ecdr814gcRdSetupAll";
10. set the trigger counters by calling "ecdr814gcSetNumTrigsAll";
11. reset the boards by calling "ecdr814gcIoctlClearAll".

Steps 4 through 11 need to be repeated when changing to a different operation mode.

After initialization, the boards need to be enabled by calling "ecdr814gcEnableSyncAll". Then the boards can be disabled and read out by calling "ecdr814gcReadAll". These two function calls form a complete daq cycle.

4.2 Operation of the Timing Generator Fanout Board

Within the Tevatron BPM system, the TGF has four function blocks: a clock generator, a TCLK decoder, a timing block and a diagnostic block. It has 2 interrupt channels each with a separate multiplexer to select the interrupt source.

The clock generation automatically starts after power-up, provided that the Tevatron RF input is present. For standalone applications, the FPGA firmware can be modified to multiply the clock output with 5/7 and route it back to the input so that the board works in a freerun fashion without the Tevatron RF input.

The TCLK decoder can be set to wait for up to 16 TCLK events, and generates a VME interrupt upon the receival of any of them. It stores the most recent event in a register for user reference. The procedure to set up the TCLK decoder is as follows:

1. install an interrupt handler for vector 198;
2. set one of the 2 IRQ Vector registers, e.g. Reg 1 at offset 0x60 for interrupt channel one, to fire at one of the 2 interrupt levels (5 & 6), e.g. 5, with an interrupt vector 198;
3. set the corresponding IRQ Source Mux register, e.g. offset 0x70, to choose the interrupt source to be TGF_SOURCE_TCLK (0x0A);
4. set the desired TCLK events in the TSG registers, offsets 0x80 through 0x9E – at power up, these registers are set to the invalid value of "0xFE";
5. enable the TCLK decoder by writing to bit 8 of the Control And Mode register;
6. enable the interrupt by writing to bit 12 of the corresponding IRQ Vector register.

The timing part uses two timebases: the Tevatron turn marker event from the BSYNC system as a coarse clock, and the 53.104696 MHz Tevatron RF clock, multiplied by 2, as a fine clock.

The start of the synchronization sequence can be triggered by a turn scaler, or by the receival of a specified "start event". The selection is made by a "start source" multiplexer, and the start can be delayed by a "pre-trigger delay" in units of "turns". Through another multiplexer, the "trigger source", the delay timer can be activated by the receival of the "start event" itself, or by the receival of the "start event" delayed by the "pre-trigger delay", or by an the external input.

When the "activate" condition is met, or external trigger received, the TGF uses the Tevatron turn maker events as references, outputs 8 sequences of synchronization signals through 8 spigots. Each synchronization sequence can have its own 0.5-bucket delay with respect to the turn makers. In closed orbit mode, one sequence consists of only one pulse; in turn-by-turn mode, it has 8192 pulses with the same 0.5-bucket delay with respect to 8192 consecutive turn makers.

The procedure to set up the timing part is as follows:

1. install interrupt handlers, e.g. vector 160 for closed orbit mode and 161 for turn-by-turn mode;
2. set the Turn Event register;
3. for closed orbit mode, set the Turn Scaler/Modulus register; for turn-by-turn mode, set the Start Event register;
4. set the Gate Count register to 1 for closed orbit mode, 8192 for turn-by-turn mode;
5. set the other IRQ Vector register, e.g. Reg 2 at offset 0x62 for interrupt channel two, to fire at the other interrupt level, e.g. 6, with an interrupt vector 160, interrupt auto-clear "false" for closed orbit mode and interrupt vector 161,

interrupt auto-clear "true" for turn-by-turn mode – when the auto-clear is "true", the interrupt will disable itself after the first occurence;

6. set the other IRQ Source Mux register, e.g. offset 0x72, to choose the interrupt source to be TGF_SOURCE_PERIODIC (9) for closed orbit mode and to be TGF_SOURCE_START (1) for turn-by-turn mode;
7. write to the Control and Mode register to: enable the BSYNC decoder, the pre-trigger delay counter and the delay timer, set the Start Source to be Turn Scaler/Modulus for closed orbit mode and Start Event for turn-by-turn mode, set the Trigger Source to be Pre-Trigger Delay, and set the Single Strobe bit;
8. enable the interrupt by writing to bit 12 of the corresponding IRQ Vector register.

At present, the TGF is configured to output only one sync pulse for each closed orbit cycle. It is possible to output multiple sync pulses if the Gate Count register is set to a value more than 1. This way, a closed orbit measurement can be derived from a few turns of turn-by-turn data in the front-end processor, and the closed orbit mode and the turn-by-turn mode only differ in the start event and the gate count. This should eliminate the need to re-load the Echotek boards when switching modes, and hence significantly simplify the DA package.

5 Calibration

The front-end DAQ is able to return raw data as well as calculated beam position to the online applications. Raw data does not require any processing on the front-end whereas calculated beam position requires the use of calibration constants.

Calibration constants are defined by offline processing. Data used for calibration will be collected from the front-end systems, running on calibration mode, and will have its data type marked as calibration data.

At startup time the front-end downloads current calibration constants. The calibration constants are retrieved by the front-end system via ACNET variables. The use of ACNET insures that the front-end automatically receives the latest calibration set.

The calibration set used by the front-end is identified by a database ID. This ID should be included in the metadata that is returned when calculated data is requested by an online application.

The system should be able to handle different calibration constants for different machine states (e.g. 150 GeV, 980 GeV). The Tevatron state device (V:CLDRST) may be used to define what calibration set should be used.

6 Diagnostics, Test Suite, and Simulation

6.1 Diagnostics

The DAQ software will provide diagnostics data via ACNET devices. It will provide means for operators or programs to detect a bad or misbehaving BPM.

Some diagnostics operations follow:

- Generate close orbit: return known closed orbit values.
- Generate turn by turn: return known values for a turn-by-turn measurement.
- Generate single turn: return known values for a single turn measurement.
- Check BPM hardware: run test procedures in the BPM hardware (one or all the BPM cards) – if supported by the hardware.
- Get buffers: return current contents of all (or selected) data buffers.

6.2 Self-Testing Procedures

The front-end DAQ should be able to perform tests on itself and on the associated BPM hardware. Results from self-tests should be available to user applications.

Hardware tests will be performed if supported, i.e., the hardware should have the capability of receiving triggers from the front-end and generate data for self-tests.

Software self-testing will be used for validating the data path from the time data is read out from the BPM until it is ready to be read via ACNET devices.

7 Monitoring

The front-end DAQ should periodically send status and statistics messages to a monitor, via ACNET devices. There should be a central monitoring application that receives data from all BPM front-ends and points out BPMs that have problems.

Data from the front-end include:

- Buffer usage
- Up time
- Available memory
- Status of processes
- Number of requests
- Tevatron status

8 Appendix

8.1 Current BPM Data Structures

8.1.1 BPM Turn By Turn

```
#define HOUSE_CHANNELS    12

typedef struct BPM_FLASH_DATA {
    char positions[HOUSE_CHANNELS];/* raw position data (ADC counts) */
    uchar intensities[HOUSE_CHANNELS]; /* raw intensity data */
    ushort valid;           /* valid data bits */
    char timestamp[3];      /* base timestamp in inverted byte order */
    uchar timeoff;          /* BCD encoded timestamp offset */
} BPM_FLASH_DATA;
```

8.1.2 BPM Closed Orbit

```
typedef struct BPM_ORBIT_DATA {
    char positions[HOUSE_CHANNELS]; /* raw position data(ADC counts) */
    ushort valid;        /* valid data bits (bit #7 indicates alarm */
                         /*    limits used - 0-low, 1-high) */
    uchar abort;         /* abort status bits */
    uchar alarm_abort;   /* alarm status in low nibble and */
                         /*    abort status in high nibble */
    uchar alarm;         /* alarm status bits */
    char timestamp[3];   /* timestamp in inverted byte order * 1000.0 */
} BPM_ORBIT_DATA;
```