Fermilab/BD/TEV
Beams-doc-860-v25
December 7, 2004
Version 25

# Tevatron Beam Position Monitor Upgrade Software Specifications for Data Acquisition

**DRAFT DRAFT DRAFT**

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 specification 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 in this document.

# 1 Overview

This note documents accumulated knowledge about the software and data formats needed for the data acquisition part of the Tevatron BPM upgrade project. The data acquisition software will run on the VME front-end computers. Figure 1 shows the software structure and the different elements involved with the BPM upgrade project.

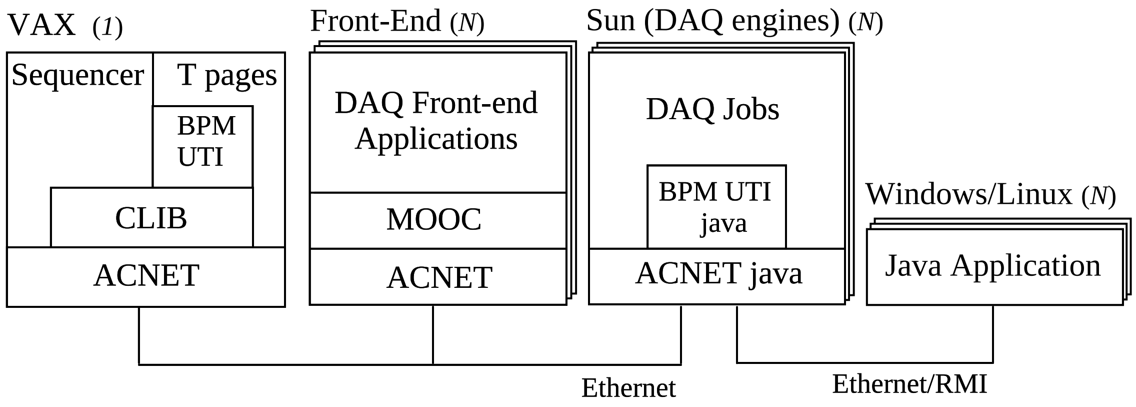

Figure 1 - Overall software architecture

This document describes the portion of code that resides on the front-end microprocessors also referred to as the Data Acquisition (DA) software. Online software resides on the VAX and DAQ engines, and forms the primary (but not exclusive) bridge between user code and BPM data.

Each front-end depicted in Figure 1 is a VME crate holding a series of BPM VME digitizing boards. One crate is referred as a ***house*** for historical reasons. There are 27 houses around the Tevatron ring, each containing one crate. A crate has up to 6 EchoTek boards (which can handle 12 BPMs. Figure 2 illustrates the organization of the cards in the VME crate (see document #1070 for hardware specifications).

The BPM digitizing boards are EchoTek module ECDF-GC814/8-XX, and each module digitizes data from 8 inputs (a BPM input is also referred as channel). A given BPM sensor generates data on 4 inputs – the A and B plates for the proton position and the A and B plates for the antiproton position. The digitized output that results from each input is raw data and each input is actually represented by two components: a real (I for in-phase) and imaginary (Q for quadrature) part. The I and Q components from the 4 inputs of a single BPM are used by the front-end to calculate the proton and antiproton position and intensity.

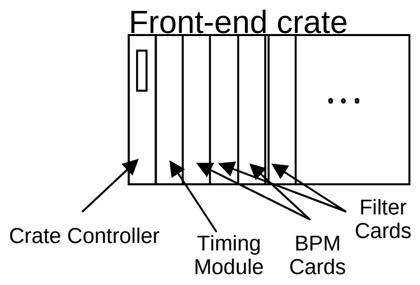

Figure 2 - Elements within a crate

The crate controller board is responsible for establishing the communication link between user applications and the EchoTek modules. All control and data transferred to/from the modules pass through the crate controller, with the exception of the diagnostic signal control relays on the filter card. Their control comes straight from the timing module.

## 1.1 Requirements

Because of the code base of existing applications, the BPM replacement system must continue to support the existing architecture. This implies:

- From the online application perspective, all communication with the front-ends is via ACNET devices. This includes data readout as well as setting readout and acquisition parameters. Internal diagnostics, however, do not have this constraint.
- Data collection happens asynchronously from data readout, i.e., the BPMs can be configured to take data continuously on certain triggers, but the data is not read out until later. Not all data that is collected is read out. This implies that the DA must manage readout requests from the online software.
- "Event assembly" is done by the online software, not the by DA. Therefore a given BPM house does not need to have any knowledge about any other BPM house(s). The data sent by the houses will however include ways for having the data synchronized by the online software. That will be provided through time stamps and/or turn counts from the timing boards.
- Embedded boards will run VxWorks
- There is one VME processor in each crate running the front-end DA, which will be responsible for handling multiple BPM cards.

## 1.2 Goals

- 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 BPM changes, and push the knowledge of BPM behavior to the BPMs themselves.
- The front ends software will use the Recycler BPM software and EchoTek drivers as a starting point for the software design.

# 2 Data Acquisition

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

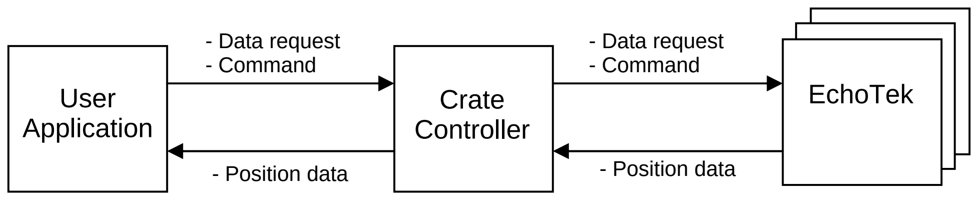

Figure 3 - Data acquisition scheme

The communication between the user applications and the front-end card is ACNET/MOOC based. The SSDN data structures are defined in the following sections and contain data defined by the Tevatron BPM Upgrade Requirements document (#554). The communication between the front-end and the EchoTek (right side on Figure 3) modules happens through the VME backplane.

Data coming from the BPM digitizers are stored into a set of buffers in the crate controller. The depth and number of logical buffers that reside on the crate controller is defined in document #903. At least some of the buffers, most notable the turn by turn data, is first stored in memory on the EchoTek boards and then transferred to the crate controller on demand, as the processor/backplane does not have the bandwidth to acquire it in real time. The actual physical implementation of the crate controller buffers will be described in the front-end software design document.

## *2.1 Clock Signals*

There are two TCLK decoders in a given house – one PMCUCD and one IPUCD. The ACNET/MOOC software infrastructure and applications are triggered by TCLK signals that are decoded by the PMCUCD card. The other card will be discussed in subsequent sections. The crate controller can be programmed to read out BPM values based on a specific TCLK signal (e.g. TCLK $75). The crate controller does not receive TVBS (Tevatron Beam Synch) events.

The preparation for data acquisition from the EchoTek cards is signaled by an arm event. An arm event may be triggered by a TCLK event, or by 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.

State transitions are not as reliable as TCLK events. For that reason they are not intended to be used as triggers. Table 1 describes a list of relevant TCLK triggers.

Table 1 TCLK, MIBS, and TBS clock events associated with the BPM system

| Event | Description | Comment |
|---|---|---|
| $71 TCLK | Tev:BPM Prepare for beam | Presently issued 0.01 secs after $4D.<br>Will be issued once before shot setup. |
| $73 TCLK | Tev:BPM High field | Issued half way up the energy ramp.<br>Was used to change alarm limits for BPMs.<br>Not needed for the new system. |
| $74 TCLK | Tev:BPM Low field | Issued after a $4D.<br>Was used to change alarm limits for BPMs.<br>Not needed for new system. |
| $75 TCLK | Tev:BPM Write profile memory | Trigger times programmed into a CAMAC 070 Card.<br>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.<br>Typically set to 2.71 secs after a $4D. |
| $40 TCLK | Tev:Reset for pbar injection @150Gev | Start of Pbar injection from MI->Tev.<br>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.<br>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.<br>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.<br>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.<br>(Every time beam is removed from the Tevatron either a $47 or $4B is issued.) |

| | | |
|---|---|---|
| $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. |

## 2.2 BPM Measurement types

The EchoTek cards can be configured to return different types of measurements. The front-end DA software is responsible for setting up the cards according to user requests coming from the online software. The different measurement types to be handled by the software are:

- Closed Orbit Measurements. The Closed Orbit is a measurement of the average position of the beam with the betatron motion averaged out. This is the position of the beam as measured by the EchoTek boards while making closed orbit measurements. For the Closed Orbit Measurements the BPM signal is filtered in order to remove the betatron motion, but the synchrotron motion is not filtered out.
- Average Closed Orbit Measurements. The Average Closed Orbit is a measurement of the average position of the beam with both the betatron motion and synchrotron motion averaged out. This is different from the Closed Orbit Measurement since the beam position is essentially “averaged” 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. The injection closed orbit is a closed orbit measurement of the beam immediately after injection of beam into the Tevatron. The closed orbit is determined from the 8192 measurements taken during the injection TBT. A simple algorithm for determining the Injection Closed Orbit is to take the average of 100 positions. A more sophisticated (but as yet undetermined) algorithm might be used.
- Injection First Turn Measurement. This is the measurement of the beam position and intensity on a single pass of the beam as it enters the Tevatron from the injection beam lines.
- Turn by Turn (TBT) Measurement. The TBT Measurement is a sequence of 8192 single turn measurements. For this measurement all of the BPMs are synchronized to begin collecting the single turn data on the same turn, and will measure the position of the beam on 8192 consecutive turns without missing any turns.
- Asynchronous Injection Measurment. This is the failsafe injection position measurement mode. If beam is not injected into the correct bucket, and beam fails to make many revolutions in the Tevatron, none of the previous measurement types will produce a reliable position. This measurement mode is

wideband and covers all of the beam revolution through multiple revolutions. Data is analyzed off-line to determine injection positions.

**It is important to note that for every measurement except for Asynchronous Injection Measurment, all BPMs are triggered off the same bunch. However, for closed orbit measurements, one trigger causes the measurement to take data for multiple bunches/multiple revolutions.**

The data acquisition system must support buffers of *N* events for these data types. For turn buffers, the arrays are of N consecutive measurements (i.e., Turn by Turn). On the other hand, closed orbit buffers are built by *N* triggers of a specific type.

The measurement types require different setups in the EchoTek modules, and, therefore have a time penalty associated with them. In general, one needs to switch data acquisition modes, to acquire a different type of measurement.

## *2.3 BPM Data Acquisition Modes*

There are different modes of BPM data acquisition. These modes are mutually exclusive due to the necessity to configure the filters and timing in the EchoTek modules for a specific mode. The modes are:

- Closed Orbit.
- Idle
- Injection
- Turn by Turn
- Asynchronous Injection
- Diagnostic
- Calibration

The modes are described in more detail in the following sections.

### 2.3.1 Closed Orbit operation

This is one of the default modes of the data acquisition system (the other being the Idle mode) and controls the filling of several different data buffers.

#### 2.3.1.1 EchoTek Setup and Timing

The EchoTek card is set up in split I-Q mode for this configuration. Two of the Graychip channels are processing the in-phase and quadrature signals in parallel and combining them into one output. The total decimation rate is set to 4096. This corresponds to a PFIR output data rate of about 18kHz. The PFIR dictates the bandwidth of the system in this configuration, and it is set up as a rolling 23 point average. Combined with the decimation rate, this gives a 3dB final bandwidth of about 700Hz. The Graychip 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 this configuration, the output stabilizes after 20 output clock ticks. The system is currently set to burst 24 points for a total processing time of 1.3ms per trigger.

The position measurements are triggered at a 500Hz rate. The 500Hz clock is sourced by the trigger module. The clock is derived from TVBS $AA triggers. The trigger card uses a divide by 94 counter to convert the 47kHz $AA triggers to 500Hz triggers. Each closed orbit trigger interrupts the processor to inform the system that a measurement is taking place. The Echotek module interrupts the processor once the burst count measurement is completed, so the processor can safely retrieve for the measurement data.

After the Echotek module has completed its measurement, the processor accesses the final I-Q pair (24th point) from each channel in the crate. The raw data from each I-Q pair is stored in a 1024 point deep circular buffer. Each channel has a separate buffer. Each I-Q pair then goes through a correction process. The first correction deconvolves the proton component of the pbar signal and the pbar component of the proton signal in [or from?] the raw I-Q values. Once the proton and pbar signals have been separated, the modulus of each channel is calculated. The inputs to an Echotek module are configured so that the first two channels are connected to opposing plates on the proton end of a single BPM. The next two channels are connected to the corresponding opposing plates on the pbar end of the same BPM. The next four channels are copies of the first four channels, but from a different BPM. The beam position is calculated by dividing the difference between the modulus of two opposing plates on one end by the sum of the modulus of the two opposing plates. The ratio is multiplied by a scale factor to convert the unitless quantity to millimeters (nominally 26mm). This position is then offset by two corrections, an electrical offset that compensates for differences in signal path attenuation, and a mechanical offset that was surveyed relative to the BPMs electrical center before the BPM was installed in the ring. This final corrected position is stored in a 1024 point deep circular buffer (Fast abort buffer discussed later). Also, the sum of the two moduli is stored with the position in the same circular buffer.

#### 2.3.1.2 Data Buffer Organization

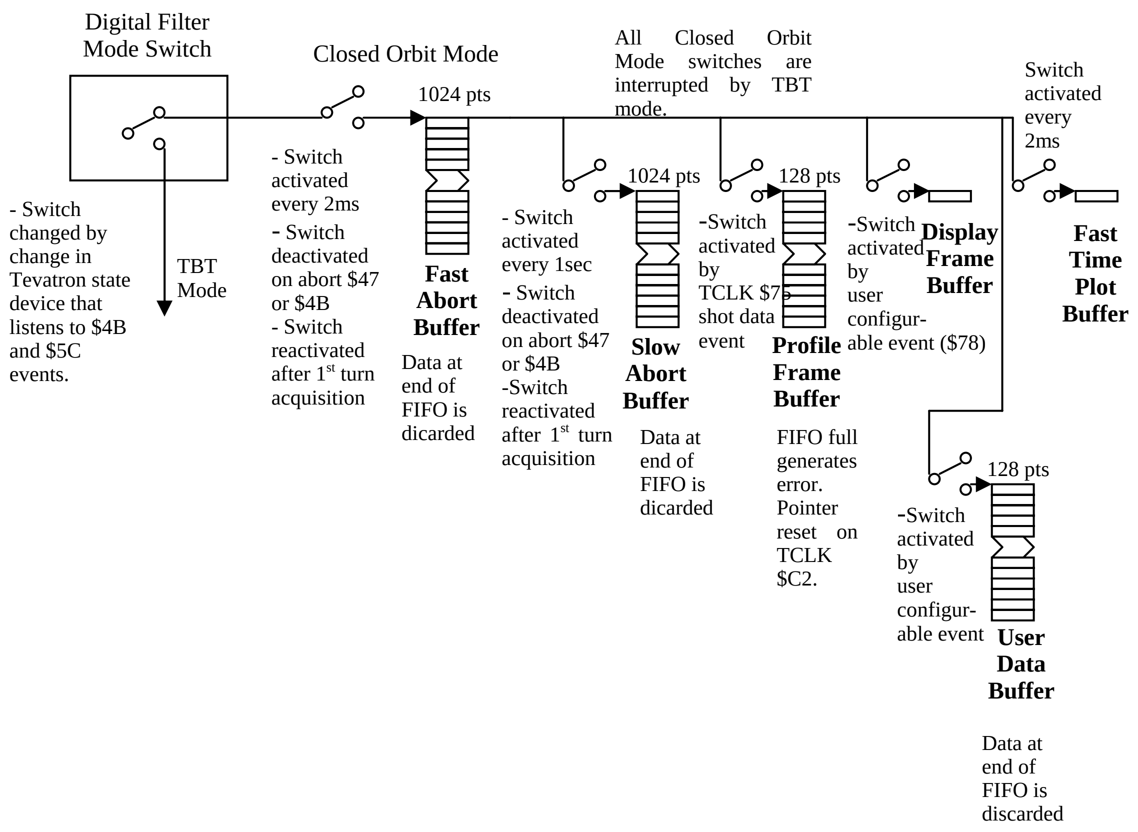
Digital Filter Mode Switch
Closed Orbit Mode
All Closed Orbit Mode switches are interrupted by TBT mode.
Switch activated every 2ms
1024 pts
- Switch changed by change in Tevatron state device that listens to $4B and $5C events.
TBT Mode
- Switch activated every 2ms
- Switch deactivated on abort $47 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 1: Buffer diagram for closed orbit measurments.

- Fast abort buffer - The crate controller takes measurement data from all inputs on all EchoTek modules every 2 milliseconds, and puts them into the fast abort buffer. One element in the fast abort buffer is a **frame** and the contents are described in the section on data structures. The fast abort buffer is a circular buffer with a depth of 1024 frames. Data from this fast abort buffer is then propagated to the other buffer types in normal operation mode.
- Slow abort buffer – data is copied from the newest frame in the fast abort buffer every 500 readings where the 500 is configurable. This is also a circular buffer that has a depth of 1024 frames.
- Profile frame buffer – data is copied from the latest frame in the fast abort buffer every time a TCLK $75 (shot data event) is received. The profile frame buffer is a FIFO buffer with a depth of 128 frames. If the profile frame buffer overflows, new values will be discarded and an alarm condition will be flagged. Note that indices in the profile frame have specific meaning. If a $75 event arrives when in a mode other than Closed Orbit, then this particular profile frame needs to be filled (maybe with a bad status.) Only one alarm will be sent per crate.
- Display frame buffer – data is copied from the latest frame in the fast abort buffer every time a TCLK $78 (display event) is received. This buffer is not an array, but is just a single frame.

- Fast time plot 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 BPMs proton position
    - A particular BPMs antiproton position
    - A particular BPMs proton intensity
    - A particular BPMs 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)

Requests for snapshot data return the most recent frame in the fast abort buffer.

In the event that a TCLK $47 (abort) or TCLK $4B (no beam left in machine) is received, the closed orbit mode will:

- Freeze data in the slow abort buffers, the profile frame buffer, and the display buffer.
- Fill the FTP buffers with a status indicating no beam.
- Will acquire a few more data points in *after* the fast abort buffer and then freeze it
- Change the mode of operation to the Idle mode.
- Do nothing to the turn by turn buffer data

### 2.3.2 Idle

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. The Echotek modules are programmed to be prepared for first turn injection. The first TCLK $4D detected by the system in Idle mode changes the mode of operation to the Injection mode.

### 2.3.3 First Turn

The first turn mode of operation is used to capture the immediate position of the beam at each BPM as the beam travels its first revolution around the ring immediately after injection. This position information is used to diagnose errors in injection position and angle in order to tune the magnets in the injection beam lines. The key to successful first turn acquisition is a well timed trigger and timely conversion of closed orbit data quickly after injection. The closed orbit data must be collected before there are changes to the Tevatron injection lattice. This is a relatively short period of time, because the "injection bump" magnets that are used for each injection are only activated briefly.

The Greychip must operate at a high bandwidth, initially, in first turn acquisition mode. Once the data is acquired, it must switch back to closed orbit mode as quickly as possible. It is important to compare the first turn orbit to the closed orbit with the injection bumps activated. There is some question as to whether all of the BPMs can be configured to run in closed orbit mode quickly enough to measure the closed orbit before the injection bumps are deactivated. To be safe, instead of trying to change the mode of operation quickly after the first turn is acquired, the system will be placed in a first turn/turn-by-turn mode. It will continue to take wideband position measurements every turn, for 8192 points. From this data, the first turn position will be retrieved, and the data will be averaged by the processor to determine an injection closed orbit.

The EchoTek card is set up in split I-Q mode for this configuration. Two of the Greychip channels are processing the in-phase and quadrature signals in parallel and combining them into one output. The total decimation rate is set to 16. This corresponds to a data output rate of about 4.65MHz. The PFIR dictates the bandwidth of the system is this configuration, and it is set up as a rolling 13 point average. Combined with the decimation rate, this gives a 3dB final bandwidth of about 360kHz. The Greychip has a 7 output clock tick latency before any data reaches the memory. After that, every two PFIR taps requires one output clock tick before the taps contribute to the output. For this configuration, the output stabilizes after 14 output clock ticks. The system will be set to burst 20 points for a total processing time of 4.3μs per trigger. The Greychip will be programmed to only output the last point into memory to preserve the total EchoTek memory and achieve 8192 point storage before data is transferred to the processor.

The default mode of operation for the system is dictated by a Tevatron states device V:TBPMM (Tevatron BPM Mode). There are two possible states for this device: closed orbit data collection, or no beam. An OOC will drive the BPM states device. When the OOC sees a clean-up event (TCLK $4B) it will change states to no beam. It will set itself to closed orbit mode after some fixed delay from a proton injection event (TCLK $4D). The system needs time to prepare all of the EchoTek modules and trigger cards for first-turn measurement. It will arm itself for first turn measurement if it receives a $4D and the BPM states device is in no beam mode. The $4D occurs about 2.5 seconds before injection and is enough time to prepare the system for injection turn-by-turn acquisition.

Triggering is critical for accurate acquisition of first turn data. There is a Tevatron beam synch event (TVBS $7C) that corresponds to the same main injector beam sync event that fires the kickers for main injector to Tevatron transfer. This event will always occur at the same time relative to the TVBS $AA marker. It will move relative to the injected beam if the injected beam goes to different buckets in the Tevatron. The BPM trigger cards are configured to listen for the $7C event and start a train of 8192 $AA based sync triggers for the EchoTek card. We can assume that the delay timing of the $AA markers is set so that the EchoTek cards sample bunch 1 correctly every time there is a trigger. However, it is also important that the memory index that refers to the first turn of beam be consistent from house to house. The way to accomplish this is to strategically delay the $7C event on a house by house basis, so that the BPM at F0 is the first BPM to see an $AA trigger after a $7C and then sequentially from F0-downstream to F0-upstream through F, A, B, C, D, and E sectors.

After the 8192 samples have been taken, the EchoTek card interrupts the processor to tell it that it completed the measurement. It is very important that the system be returned to closed orbit mode as soon as possible. My proposal for minimizing the amount of time between first turn acquisition and closed orbit is to set up the Greychip to operate with all

4 channels in split I/Q mode. One pair of channels is set-up for closed orbit decimation, and one set is set up for turn-by-turn decimation. During closed orbit operation, the turn-by-turn channel output is deactivated. After an abort, the processor deactivates the closed orbit outputs and activates the turn-by-turn outputs. After the measurement is complete, the processor reverses the process and reactivates the trigger card for closed orbit sample rates. This keeps the integrity of the turn-by-turn data on the EchoTek memory while closed orbit measurements take place, allowing the processor to retrieve the data at its earliest convenience.

### 2.3.4 Turn by Turn

The turn-by-turn mode of operation for the Tevatron BPMs is very similar to the first turn mode of operation. The major difference between the two modes is that first turn mode occurs only when the first protons are injected, and turn-by-turn acquisition can occur at any user specified event. During turn-by-turn acquisition, the BPM system samples the position of the beam at every BPM, at every revolution for 8192 turns. Data from these arrays of points 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.

The EchoTek card is configured in the exact same way for turn-by-turn mode and first turn mode. The mode is armed by a TCLK event $77. A separate Tevatron beam synch event (TVBS $DA), that is derived from a delayed TCLK $77, triggers the system to start taking data on the next turn marker. The delay must be set long enough to allow time for the system to change modes before triggering. The $DA trigger should be timed the same way as the $7C trigger in the first turn acquisition mode. This ensures consistent correlation between data arrays of different houses.

### 2.3.5 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.

### 2.3.6 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. The application for controlling and monitoring the crate will be T?? (title?).

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 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/BLM house. 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)

## 2.5 State Diagram

The crate controller uses the Tevatron state devices for two purposes. It uses **V:TBPMM** to determine which mode of operation to return to after a reboot or a change from diagnostic mode. Also, it includes various pieces of information from other states devices in the metadata that is returned. Please refer to

URL: 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 independently, freeing the sequencer from the task of informing the BPMs about TeV state changes.

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
14 - HEP
15 - Pause HEP
16 - Proton removal
17 - Unsqueeze
18 - Flattop2
19 - Deceleration
20 - Extraction porch
21 - Extract pbars
22 - Reset
23 - Recovery
24 - Ramping

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

**V:TBPMM** – 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** – Pbar coalescing state for metadata 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

## 3 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.

## 3.1 DAQ Parameters

| Description | Scope | Type | Range |
|---|---|---|---|
| Disable automatic triggering of injection mode on TCLK $4D | System | ACNET<br>Operator pages | State device<br>Restore last value |
| Define the current mode of operation of the BPMs | House | ACNET<br>Operator pages | DA<br>Diagnostics<br>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<br>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<br>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<br>Abort Buffers, Turn by Turn, Profile | System | Compilation | |
| TCLK needed to arm TbT request | System | Compilation/Configuration File (?) | |

## 3.2 Timing Parameters

| Description | Scope | Type | Range |
|---|---|---|---|
| Delay for trigging after specific TCLK | House | ACNET<br>Engineering pages | msec |
| | EchoTek module | | |
| | Channel | | |
| Delay for arming after specific TVBS | System | ACNET<br>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 |

## 3.3 Diagnostic Parameters

| Description | Scope | When can Change | Range |
|---|---|---|---|
| Debug Level | House | ACNET<br>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 or BLM |
| Software self triggering for first turn (ie, not on bsynch) | System | ACNET Operator pages | |

# 4 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 BPMdata

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 4.2. 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 500Hz.

## 4.1 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 URL for a detailed field description,

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

For reading BPM and BLM 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.

### 4.1.1 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:

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

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:

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

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**

### 4.1.2 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:

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

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

### 4.1.3 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:

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

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

## 4.2 Data Structures (Output Data)

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.

### 4.2.1 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 */
}
```

### 4.2.2 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[];
}
```

### 4.2.3 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[];
}
```

### 4.2.4 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[];
}
```

### 4.2.5 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 */
    }
```

# 5 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 containing 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. A complete specification of the communication channels will require interaction and agreement between the BPM FPGA code and the front-end software. During the software development, data from the EchoTek board may be simulated using the agreed interface.

# 6 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.

# 7 Diagnostics, test suite, and simulation

## *7.1 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.
- Check BLM hardware: run test procedures in the BLM hardware (one or all the BLM cards) – if supported by the hardware.
- Get buffers: return current contents of all (or selected) data buffers.

## *7.2 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.

## 8 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

# 9 Appendix

## 9.1 Current BPM data structures

### 9.1.1 BPM Single Turn (Flash)

```
#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;
```

### 9.1.2 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;
```

### 9.1.3 BPM Data Structure

```
typedef struct BLM_DATA {
    uchar raw_losses[HOUSE_CHANNELS];     /* raw loss data */
    uchar status;      /* BLM status */
    char timestamp[3]; /* timestamp in inverted byte order * 1000.0 */
} BLM_DATA;
```