Fermilab/BD/TEV
Beams-doc-860-v25
June 30, 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

Abstract

This document contains the specification for the BPM/BLM upgrade data acquisition software. Expected operating modes and interactions to the BPM/BLM 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.

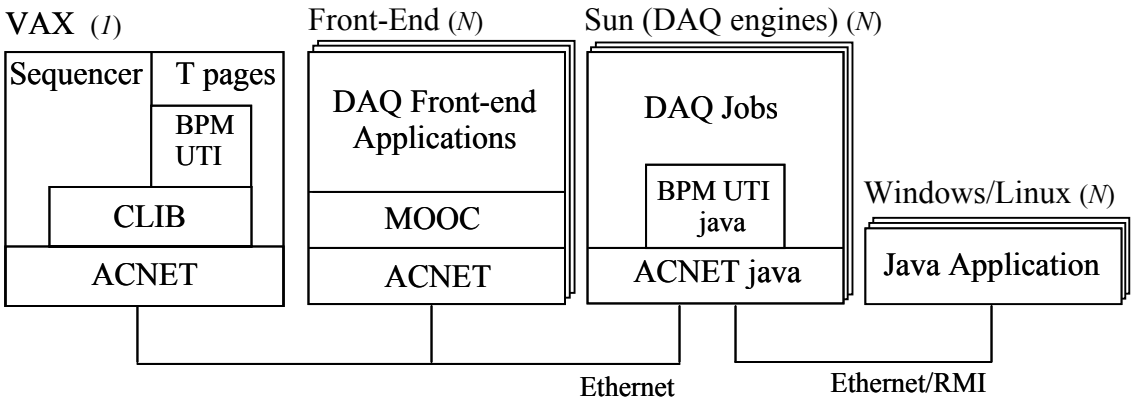

Figure 1 - Overall software architecture

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 and BLM digitizing hardware. One crate may also be 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) and 2 BLM chassis, containing 12 channels each. 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/BLM 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.

The BLM hardware is described in document #764. Notice on figure 2 that the BLM hardware is not connected to the VME back plane. The communication with the crate controller is done via a digital PMC module.

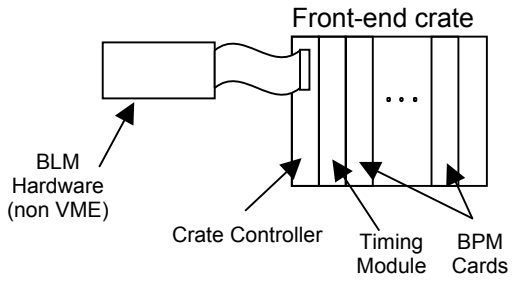

Figure 2 - Elements within a crate

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.

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 and BLM 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.
- Guarantee time to arm for a TBT request to < 100 msec.

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.

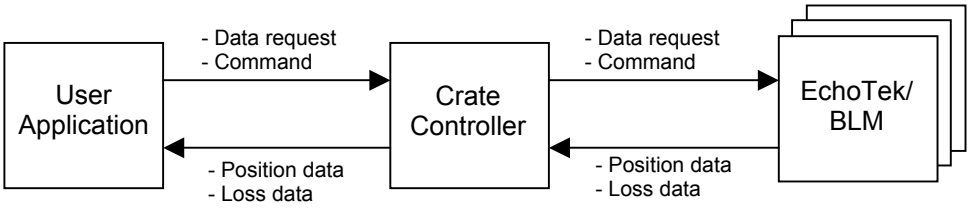

Figure 3 - Data acquisition scheme

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. For the BLMs, data is exchanged through a digital I/O module on a PMC board.

Data coming from the BPM and BLM 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/BLM values based on

a specific TCLK signal (e.g. TCLK $75). The crate controller does not receive BSYNCH 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. 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.
$76 TCLK	Tev:BLM Write display frame	Variable time and trigger.
$77 TCLK	Tev:BPM Flash trigger	Delayed from the MIBS injection event. Delay is in units of TREV (including fractions of a revolution.)
$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	TCLK echo of MIBS $7C.

	transfer trigger from MI to Tevatron	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.)
$7C? TBS	MI:Ini MI->Tev proton coli	Tevatron Beam Sync event at injection. Similar to the MIBS $7C.
$77? TBS	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 (aka Frame data). 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 10 synchrotron periods or for about 0.3 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 all 8192 positions. A more sophisticated (but as yet undetermined) algorithm might be used.
- Single Turn Measurement. This is the measurement of the beam position and intensity on a single pass of the beam.

- 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.
- Turn Data (or snapshot) is the instantaneous single value (i.e., not averaged over n readings) of the sensors once the trigger is received.

It is important to note that for turn measurements, all BPMs are triggered off the same bunch. However, for closed orbit measurements, the specific bunch used in the averaging can be different across BPMs.

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:

- Normal operation.
- Injection
- Turn by Turn
- Diagnostic
- Calibration

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

2.3.1 Normal operation

This is the default mode of the data acquisition system, and controls the filling of several different data buffers. These are:

- 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 *n* readings where *n* 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 Normal Operation, 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 (single turn) 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 normal operation mode will

- Freeze data in the fast and slow abort buffers, the profile frame buffer, and the display buffer and slow abort 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
- Disable all data acquisition modes until the next prepare for beam TCLK event is detected.
- Do nothing to the turn by turn buffer data

Upon receipt of a TCLK $71 (prepare for beam), if the DA is running, any acquisition mode is halted. If there is no DA active the fast and slow abort buffers are cleared, the profile frame buffer cleared and the pointer is reset to zero, injection mode is enabled, and normal operation mode (re)started. When restarting in normal operation mode, the DA will flag each data point with a low intensity threshold status until real beam is actually detected. The TBT buffer is not cleared on the prepare for beam event.

2.3.2 Injection

Injection mode has the highest priority of all data acquisition modes and will supersede any other acquisition mode in progress. It is a specialized mode that needs to take turn by turn data followed quickly by a closed orbit measurement. It happens automatically on receipt of a TCLK $4D (proton injection event). This might be extended in the future to occur on a $40 (pbar injection event) as well. There is a 2.7 second delay between the TCLK $4D event and the trigger to actually take the data (BSYNC injection event). The actual mode switch will occur somewhere in this interval instead of the beginning to allow normal mode monitoring to run as long as possible.

Shortly after the TCLK $4D is received, all other modes will halt and the EchoTek modules will arm themselves waiting for the corresponding trigger, $5C (injection event main injector/tev BSYNC). Once a turn by turn measurement of 8K consecutive turns has been completed, the crate controller will then calculate a closed orbit measurement using the data from the turn by turn buffer. The crate controller knows that the readout is complete after receiving an interrupt from the timing card signaling *one* of the EchoTek cards is ready to be read out.

Once the calculation is complete, the mode automatically returns to normal operation. If the corresponding BSYNC does not occur in a specified amount of time, an error status will be returned and the normal operation mode restarted.

Injection mode needs to be disabled and enabled again on request because there are times when closed orbit data is preferred during proton injection.

Note that for a given BPM, the turn by turn buffer on the EchoTek board will have the data from the first turn in a fixed slot in the array, but not necessarily the first element in the array (e.g., 3). This needs to be configurable on a BPM by BPM basis.

2.3.3 Turn by Turn

Turn by turn mode is very similar to injection mode in that an 8K consecutive turns are taken when a turn by turn request is made. A turn by turn request comes through an ACNET SETDAT request, and contains any user specification for the measurement.

The turn by turn mode is armed with a TCLK event $77 and the data collection is started on a TeV BSYNC (event $77). The crate controller is informed that the measurement is complete by an interrupt from the timing system (same as the injection mode).

Note that for a given BPM, the turn by turn buffer on the EchoTek board will have the data from the first turn in a fixed slot in the array, but not necessarily the first element in the array (e.g., 3). This needs to be configurable on a BPM by BPM basis.

2.3.4 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.

2.3.5 Diagnostic Mode – we don't know what this means yet

When there is no beam in the machine, the crate controller can be configured to set up one BPM channel to receive a 53MHz tone while remaining channels in the BPM can be digitized and analyzed (can be readout as Turn-by-Turn data with header set to diagnostics). The controller should be able to handle this type of request for one BPM in the system.

2.4 BLM Data Acquisition Modes

The BLMs on the other hand are much simpler. There is no need for mode switching for BLMs, and there are no turn-by-turn requirements. Internal software timers periodically trigger the BLM read out. The trigger periodicity is given by the BLM integration time, which according to the document #764, is a 63 ms time constant.

Every 63 ms a new BLM channel value is read out and stored in an internal buffer, dimensioned to hold enough data for a given time frame. The online software can request data from the front-end DA BLM buffer. Those requests specify a range, which can vary from a single entry or to the whole buffer. The entire buffer may be requested in events such a Tevatron abort.

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/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)

2.6 State Diagram

The crate controller's only interest in the Tevatron state devices is to include various pieces of information from them 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.

Are we sure we don't want to return status when somebody asks for something that shouldn't be – e.g., pbar data in a proton injection state?

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

The following states will require functionality from the BPMs:

Proton Injection Porch (1)

From the time the Tevatron is in *recovery* until it reaches either the *proton injection porch* (event $43), the BPMs are not required to be doing anything. The system can do an inventory of the current BPMs, checking whether they working or not. Problems detected must be reported to operators and data taken has to be tagged with the BPM status for later analysis.

Proton Injection Tuneup (2)

Only proton data is needed in this state. Take both single turn, in particular, first turn, as well as triggered (either on a TCLK ($2B+$4D) event or a TCLK event + delay) closed orbit data in this state. No pbar data available.

Reverse Injection (3)

Old system uses only Main Injector BPM data taking closed orbit data triggered on a TCLK. Triggered by event $2A+$5D.

Inject Protons (4)

Switch now to coalesced beam depending on the operational mode (the device **V:COALP** should be checked). SDA is collecting closed orbit data on demand. Triggered by event $2B+$4D.

Pbar Injection Porch (5)

SDA collecting proton closed orbit data on demand.

Inject Pbar (6)

Both proton and pbar data are needed here. Collecting closed orbit data on TCLK events as well as on demand. There may be unique cogging values for each BPM.

Acceleration (9)

Profiling data taken. Profiling is a closed orbit measurement taken at *n* points during the acceleration process at 100GeV, 200GeV,... , 980GeV. Right now p only – pbar would be nice. Currently generated by a TCLK event that is coming from the CAMAC timing modules.

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

V:TEVMOD – Tevatron high-level mode
1 - colliding beams: take closed orbit data
2 - proton only: take proton closed orbit data
3 - dry squeeze: take closed orbit data
4 – ramping: take closed orbit data
5 - recovery/turn on: run diagnostics
6 - off: run diagnostics when machine is 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: run diagnostics while no beam in machine
2 - protons: take proton closed orbit data
3 - pbars: take pbar closed orbit data
4 - protons and pbars: take proton and pbar closed orbit data

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 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 (?)	

3.2 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 BSYNCH	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

3.3 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 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 BPM/BLM 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 and BLM data read by the online software will be organized according to the data structures defined a 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
- 0x0041 for BLM retdat/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:

`0000/0022/000`**`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).

A house also is responsible for reading out the Beam Loss Monitor system, which will consist up to 24 channels. These channels can also be read out via FTP devices through the following configuration:

0000/0027/0000/27**YY**

where **YY** is the BLM channel number, varying from 00 to 17 (23 decimal).

The maximum number of FTP devices provided by one house is 96:

1 I/Q/Sum x 48 channels + 2 p/pbar x 12 position/intensity + 24 BLM = **96**

4.1.2 SSDN Mapping for RETDAT devices

The RETDAT protocol is used to retrieve most data read out by the BPM and BLM systems. 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)

And the following BLM data

- Abort buffer (array)
- Display buffer
- Injection turn-by-turn buffer (array)
- Profile frame buffer

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. The BLM devices will return data from all BLMs managed by the system. 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

The SSDN for the RETDAT BLM devices are:

0000/0026/000**X**/260**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 – Abort buffer
1 – Display buffer
2 – Profile buffer
3 – Injection 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:

0000/0021/0000/218**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

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 BLM Data Structures

```
typedef struct TEVATRON_BLM_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 */
    long num_detectors;       /* number of detectors present */
    float losses[24];         /* loss values in rads/sec */
    long status[24];          /* status values */
}
```

Proposed status values:

```
OK = 0
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_BLM_DATA {
    TEVATRON_BPM_HEADER       header;
    long num_frames;   /* number of frames returned */
    TEVATRON_BLM_FRAME_DATA frame_data[];
}
```

4.2.6 BLM Time Slice Data

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

```
typedef struct TEVATRON_BLM_TIME_SLICE_VALUE {
    long status;                    /* detector status */
    unsigned long milliseconds;     /* milliseconds since first frame */
    float loss;                     /* loss value in rads/sec */
    }

typedef struct TEVATRON_BLM_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_BLM_SLICE_VALUE frame_data[];
    }
```

4.2.7 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/BLM 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.

Access to BLM data and commands will happen through a PMC-DIO64 card that connects the crate controller to the BLM hardware. Commands sent and data read from the BLM hardware are defined in the documents #764. BLM buffers contain single values and are buffered on the front-end board.

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;
```