Fermilab/BD/TEV
Beams-doc-1060-v4
April 29, 2004
Version 0.3

Tevatron Beam Position Monitor Upgrade Online Software Specification

Brian Hendricks
Fermilab, Accelerator Division, Controls Department

Abstract

This document contains the specification for the Tevatron BPM/BLM upgrade online software. Data acquisition library support and console applications are described. A methodology for phased installation of VME-based hardware is also explained.

6

Overview

The Tevatron BPM/BLM system is supported by a suite of applications for diagnosing the health of the systems as well as for displaying and archiving data retrieved from the system. In addition, there are applications that consume data from the system to perform tasks such as smoothing the Tevatron orbit. Underlying these applications is a suite of library routines. On the VAX computers, this suite is named BPMUTI, and it resides in the user library named ul_cbsaux. For Java applications, this support is provided by the package gov.fnal.controls.daq.bpm. Figure 1 shows the software structure and the different elements involved with the BPM upgrade project.
(This diagram should show multiplicity (i.e., n crates, n data acquisition engines, 1 VAX).)

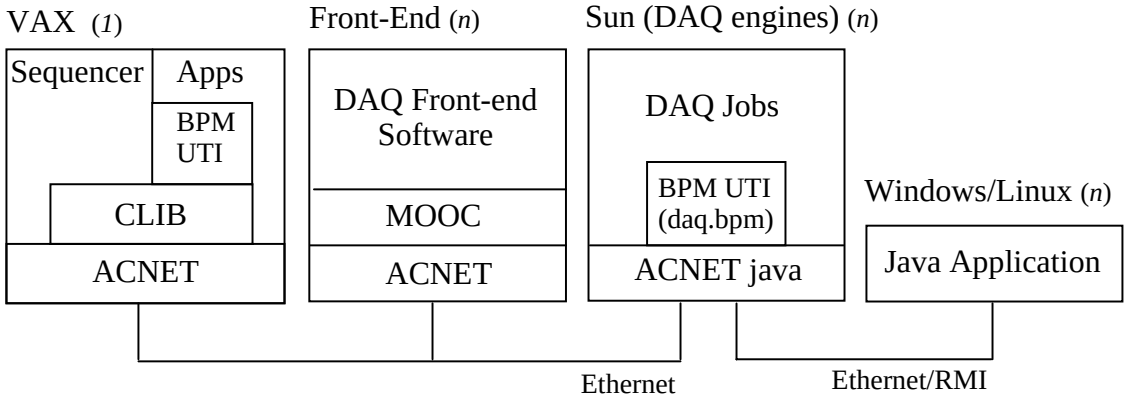

Figure 1 - Overall software architecture

Since the existing applications are built on top of library support that provides a layer of abstraction which isolates them from the specific front end data structures and since the basic functionality of the system is similar to the Multibus-based systems, these applications will continue to serve with the VME-based systems. The differences in the underlying data structures (see Beams document 860) will be handled in the libraries which will be where most of the work in the online software will be done.

This document addresses the user applications that access BPM data and the libraries that they call to perform that access. It does not cover the front end software which is described in Beams-doc-860. It also does not cover the generic ACNET control system facilities of fast time plotting and data logging. These facilities rely solely upon front end support and are therefore not pertinent to this document.

Applications

General Display Application (T39)

The application program T39 is used for the general reading, display, and saving of BPM data. It allows the user to choose the type of BPM frame which presently includes flash, display, snapshot, and profile. It also allows the user to choose a range of frame numbers. In addition to the position data, intensity or loss data can be selected for display. After the data is read, it is automatically plotted. The data is now in program memory and can be replotted using different plot setup parameters. It can also be listed in numeric form. If the data is deemed to be important, it can also be saved in a database file along with appropriate metadata describing the measurement. Files are typically saved in a circular fashion, but they can also be protected to prevent their being overwritten.

Other features of this program include the ability to configure the timing for BPM data acquisition. This program also allows display and control of which BPMs are marked out of service in the BPM configuration database. This is used to prevent other programs such as the orbit program from using detectors that are known to be malfunctioning. This application supports display and comparison of timestamps from all of the houses in order to verify the consistency of data sampling around the ring. It also supports the correlated reading and archiving of a list of other ACNET parameters which are defined by a database table. This allows the Tevatron department to monitor changes in other parameters with respect to changes in the orbit. Display and comparison of orbits saved by SDA is also supported.

This program will need to be modified to support all new BPM frame types including injection and slow snapshot. The timing window will also need to be modified to reflect changes in the timing for the VME-based systems. It will also need to support all changes to the supported metadata saved with data files.

Turn by Turn Application (T42)

This program supports the reading, displaying, and saving of BPM turn by turn data. This data is primarily taken during study periods to measure various beam parameters. This program can read data from every detector in the Tevatron and save it to flat files that are web accessible so that other programs can import the data and manipulate it.

At the present time, it appears that this program will need very few modifications due to the VME-based BPM systems.

Sequencer

The Sequencer is the general application that coordinates and controls the accelerator complex especially during critical times such as performing a Collider shot. Along with other systems, the Sequencer interacts to some degree with the BPM system. It sets timing and mode parameters so that the BPM system performs in a desired way.

With the VME-based system, the Sequencer command to set the particle and bunch modes (BPM_MODE) will no longer be needed. This information, plus much more, will be transmitted to the BPM systems via state devices. The command to set profile trigger times (SET_070) should remain unchanged unless a different mechanism for triggering the profile frame triggers is devised. The Sequencer scripts will have to be modified by the Tevatron Department personnel to make sure that the proper state values are asserted at the proper times and that any new timing information is incorporated.

BPM System Diagnostics and Calibration

This is a new program which must be written to support the VME-based BPM systems. It will need to access and display system diagnostic information produced by the front ends. This information will allow the user to judge the health of the various systems. This program will also support executing and reporting the results from tests to be run in the front end environment.

This program will also be used to calibrate and update the BPM scaling constants contained in the front ends.

BPM Beam Diagnostics (T41)

This program utilizes beam measurements to determine if the BPM system is functioning properly. It measures the orbit using the latest snapshot. Then it creates a one bump and makes another orbit measurement. It compares the differences between the two orbits with what the predicted differences would be. It can then detect BPMs which have their polarity switched, aren't responding correctly, or aren't responding at all.

This program should be unaffected by the installation of the VME-based systems. All necessary changes will take place at the library layer.

BPM Profile Data Extractor (C45)

This program extracts data for individual detectors from a set of profile frames and displays it.

This program should be unaffected by the installation of the VME-based systems. All necessary changes will take place at the library layer.

SDA BPM Extractor (BPM Viewer)

This is a Java program that extracts BPM data from SDA and displays it. The user is allowed to select from a set of shots and from a list of cases.

This program should be unaffected by the installation of the VME-based systems. All necessary changes will take place at the library layer.

BPM/BLM Time Slice Display

This program displays BLM data for specified detectors over time using the snapshot buffer. This program is typically used to see the evolution of losses preceding a beam abort in order to understand its cause.

This program will need to be greatly expanded to support the display of BPM data as well as BLM data and to support profile frames in addition to snapshot frames. This program will also have the capability of saving data in files. The program should be able to handle file data equally a well as live data. The plot feature should be enhanced to support a cursor which could interrogate the contents of the plot and to support the plotting of certain key clock events. Plotting of abort concentrator information might also be added in the future. A feature to perform an FFT on the data should be included to look for patterns in the signals.

Tevatron Orbit Program (TOP) (C50)

TOP uses the data from snapshot or profile frames either live or from BPM data files in order to smooth the Tevatron orbit. It can also save BPM data used in a smooth in a file.

Unless the use of new frame types is desired, this program should not have to be changed other than to support changes to the metadata saved with the BPM data files.

Tevatron Orbit Closure (T117)

This program compares certain beam positions from the first turn flash frame data with the first closed orbit measurement data. It uses this comparison to see how well beam is being injected onto the closed orbit. If there are position or angle errors, the program will change the settings of correctors to attempt to fix the problem.

The only change that I can see at this time to this program is to use the new injection closed orbit frame for its closed orbit data.

Library Support

Overview

The bulk of the work in providing online software support for the VME-based BPM systems will take place in the BPM libraries. The largest library is called BPMUTI and it provides support for accessing BPM/BLM data by programs written in C/C++. There is also a package named daq.bpm which provides support for programs written in Java. This package also provides the underlying BPM support for the OSDA (Offline Sequenced Data Acquisition) library.

The BPM libraries provide a layer of abstraction between application programs and raw BPM data supplied by front end computers. Applications first establish what BPM system (Tevatron, Recycler, etc.) that they want to interact with. After that has been accomplished, generic routines for reading position, intensity, or loss data can be called. There are also routines for saving data, plotting data, retrieving configuration information, retrieving diagnostic information, and even setting configuration information. The important thing to note is that it doesn't matter which BPM system you want to interact with. You always use the same routines. The difference is in the private underlying routines that the library routines call. Each BPM system has routines to read and scale the data coming from the front ends and return the scaled data to the higher level routines. The primary work will be to write these new routines. There will also need to be support added for new frame types including injection and slow abort.

A detail that must be handled by the library routines is the support of returned raw data. The routine for returning position data supports the retrieval of raw data. The Tevatron support, however, does not presently honor this request. This routine will be modified to return all four I and Q values for each detector if raw data is requested.

One of the aspects of this project is that it will be a phased implementation with a mix of Multibus-based and VME-based systems coexisting for some months. This will be handled by the BPM libraries through the use of configuration database tables. There are

several tables presently in use by the BPM library. The BPM house configuration table contains a field labeled version. The value of the version field will be used to indicate whether a given house, for example A1, is a VME-based (version 1) or a Multibus-based (version 0) BPM and process it accordingly. The date that the version number for a given house was changed will also be inserted into the table to enable SDA applications to properly reconstruct data from the interim period. This will be transparent to the upper level library routines and the applications that call them. If one of the new frame types is requested, there will be errors returned from Multibus-based houses. There is presently no provision for converting detectors one at a time.

Existing Library Routines

A brief description of each library routine follows. A detailed description of these routines including lists of arguments and returned values can be found at **http://adcon.fnal.gov/www/controls/user_libraries/ul_cbsaux/oneline.html.**

General Purpose Library Routines

bpm_machine_c - sets a particular machine for BPM data acquisition
bpm_data_source_c - sets the data acquisition source for BPM data

BPM Data Retrieval Library Routines

bpm_get_data_c - gets BPM data for the machine last requested by bpm_machine_c
(accesses TEVATRON_BPM_FRAME_DATA structures)
(must be modified to optionally return I and Q data for Tevatron)
(may be modified to return metadata)
bpm_get_intensity_c - gets BPM beam intensities for current machine
(accesses TEVATRON_BPM_FRAME_DATA structures)
bpm_get_tbt_data_c - returns BPM turn by turn positions and timestamps
(accesses TEVATRON_BPM_TBT_DATA structures)
(should this be modified to return I and Q data?)
(must be modified to support the selection of particle type)
bpm_get_tbt_intensity_c - returns BPM turn by turn intensities and timestamps
(accesses TEVATRON_BPM_TBT_DATA structures)
(should this be modified to return I and Q data?)
(must be modified to support the selection of particle type)
bpm_get_tbt_house_data_c - returns BPM turn by turn positions and timestamps for an entire house
(should this be modified to return I and Q data?)
(must be modified to support the selection of particle type)
bpm_is_tbt_trigger_enabled - determines if BPM turn by turn triggering is enabled
(Is this still needed?)

bpm_get_tbt_timing_c - returns BPM turn by turn timing
bpm_get_global_tbt_timing_c - returns BPM turn by turn timing
bpm_read_tbt_mode_c - reads BPM turn by turn data acquisition mode
(Is this still needed?)
bpm_get_threshold_c - returns BPM sample threshold levels
bpm_get_calibration_spec_c - reads calibration mode data
(new) bpm_get_raw_tbt_data_c - returns BPM turn by turn I and Q values and timestamps
(new) bpm_get_time_slice_data_c – returns BPM data for a single detector over multiple samples
(accesses TEVATRON_BPM_TIME_SLICE_DATA structures)
(new) blm_get_time_slice_data_c – returns BLM data for a single detector over multiple samples
(accesses TEVATRON_BLM_TIME_SLICE_DATA structures)
(new) bpm_get_metadata_c – returns metadata from a BPM measurement
(new) bpm_get_status_c - returns overall front-end, BPM and BLM status

BPM Control Library Routines

bpm_enable_tbt_trigger_c - enables/disables BPM turn by turn triggering
(Is this still needed?)
bpm_set_tbt_timing_c - sets BPM turn by turn timing for a single house
bpm_set_global_tbt_timing_c - sets BPM turn by turn timing for an entire machine
bpm_set_tbt_mode_c - sets the BPM turn by turn acquisition mode
(Is this still needed?)
bpm_set_threshold_c - sets BPM sample threshold levels
bpm_set_calibration_spec_c - sets calibration mode
(new) bpm_set_diagnostic_mode_c – enables or disables diagnostic mode

BPM Configuration Information Library Routines

bpm_query - returns the number of BPM position values for current machine
bpm_get_names - returns names of BPMs in same order bpm_get_data_c returns values
bpm_get_intensity_names - returns names of BPM intensity monitors
bpm_get_tbt_names - returns valid BPM names for reading turn by turn data
bpm_get_locations - gets BPM detector locations for current machine
bpm_get_intensity_locations - gets BPM intensity detector locations for current machine
bpm_get_offsets - gets BPM detector offsets for current machine
bpm_get_detector_status - gets BPM detector status values for the current machine
bpm_set_detector_status_c - sets BPM detector status values for the current machine
bpm_crate_info_c - returns BPM crate information
bpm_to_crate_and_channel_c - returns BPM detector crate and channel values

bpm_get_crates_and_channels - gets BPM detector crate and channel values for the current machine
bpm_get_tbt_crates_and_channels - gets BPM turn by turn detector crate and channel values for the current machine

Save File Library Routines

bpm_save_data - saves BPM data for the current machine
(must be modified to handle additional metadata)
bpm_directory_info - returns an array of BPM file directory information
(must be modified to handle additional metadata)
bpm_file_menu_c - displays a menu of BPM save files for the current machine
bpm_master_file_menu_c - displays a menu of BPM master save files
bpm_get_default_save_file - returns the default BPM save file for the current machine
bpm_get_save_file - returns the currently selected BPM save file for the current machine
bpm_set_save_file - sets the active BPM data save file
bpm_rename_file_c - renames a BPM file for the selected machine
bpm_protect_file_c - changes the protection for a BPM file for current machine
bpm_set_reference_file_c - changes the reference file status for a BPM file for current machine

BPM Miscellaneous Library Routines

bpm_map_from_model_c - copies array of lattice values into BPM data order
bpm_plot_data - plots BPM data
(must be modified to handle additional metadata)
bpm_interrogate_plot - enable interrogation of existing BPM plots

BLM Library Routines

blm_get_data_c - gets BLM data for machine last requested by bpm_machine_c
blm_get_locations - gets BLM locations for current machine
blm_get_names - returns names of BLMs in same order blm_get_data_c returns values
blm_query - returns number of beam loss monitors (BLMs) per frame for machine

Java Classes

There are five classes and two interfaces that support accessing BPM data. They are found in the package gov.fnal.controls.daq.bpm and are documented at http://www-bd.fnal.gov/javadoc/build/api/gov/fnal/controls/daq/bpm/package-summary.html.

TevBPMItem – Tevatron data request object
TevBPMScaledItem – Tevatron scaled data request object
BPMDataObject – raw data object
TevBPMScaledDataObject – Tevatron scaled data object
BPMScaler – scales raw data into positions

Configuration Database Tables

The BPM library routines are dependent upon a set of database tables that configure their functionality. There is a system specification table which names and describes each BPM system and provides some information about the overall capabilities of the system. There is also a field that describes the configuration of each BPM crate or house. There are three more tables that give configuration information for individual position, intensity, and loss readbacks respectively. There is also a set of database tables that provide the BPM file system.

This upgrade will require both the addition of fields to some of the above tables as well as the insertion of new values in existing fields as systems are converted from Multibus-based to VME-based.

As mentioned earlier, one of the key components of the phased installation of VME-based crates is the version field in the crate configuration table. A field will be added to indicate the date that the version number was changed to enable SDA software to reconstruct orbits during the transition period. This table also contains the device mapping for each crate. This upgrade will require that fields be added to the crate configuration table to support both new types of orbits as well as for raw (I and Q) data. Also, as crates are converted, new values for the existing device fields in this table will have to be entered.

It appears that a new table will need to be created to assist SDA routines in reconstructing the configuration of the system for a given point in time. This table will contain a timestamp and corresponding version values for each house.

The position configuration table will need new fields to support raw turn by turn data retrieval. This table will also need to have its current device field values updated as crates are converted.

The loss configuration table will not need any new fields at this time, but its device field values will have to be updated.

The BPM file system database tables will need new fields to support new metadata to be acquired and saved with each file. They will also have to be updated to store raw I and Q data in addition to the present data types.

Front End Data Structures

The description of the front end data structures that the BPM library routines will operate on is contained in Beams-doc-860.

ACNET Device Names

In the ACNET device name listed below, lower case letters indicate fields that are variable. A field of "hh" indicates that that field will be comprised of a two character house name such as "A1". A field of "nnn" indicates the three character name of a detector location starting with the sector such as "A11". A single character "p" indicates a plane field of either "H" or "V".

T:hhBPFL – BPM flash frame device
(TEVATRON_BPM_FRAME_DATA structure)
T:hhBLFL – BLM flash frame device
(TEVATRON_BLM_FRAME_DATA structure)
T:hhBPIF – BPM injection frame device (closed orbit)
(TEVATRON_BPM_FRAME_DATA structure)
T:hhBLIF – BLM injection frame device
(TEVATRON_BLM_FRAME_DATA structure)
T:hhBPDF – BPM display frame device (closed orbit)
(TEVATRON_BPM_FRAME_DATA structure)
T:hhBLDF – BLM display frame device
(TEVATRON_BLM_FRAME_DATA structure)
T:hhBPPR – BPM profile frame device (closed orbit)
(TEVATRON_BPM_FRAME_DATA structure)
T:hhBLPR – BLM profile frame device
(TEVATRON_BLM_FRAME_DATA structure)
T:hhBPFA – BPM fast abort frame device (closed orbit)
(TEVATRON_BPM_FRAME_DATA structure)
T:hhBLFA – BLM fast abort frame device
(TEVATRON_BLM_FRAME_DATA structure)
T:hhBPSA – BPM slow abort frame device (closed orbit)
(TEVATRON_BPM_FRAME_DATA structure)
T:hhBLSA – BLM slow abort frame device
(TEVATRON_BLM_FRAME_DATA structure)
T:hhBPUF – BPM user frame device (closed orbit)
(TEVATRON_BPM_FRAME_DATA structure)
T:hhBLUF – BLM user frame device
(TEVATRON_BLM_FRAME_DATA structure)
T:pTTnnn – turn by turn data
(TEVATRON_BPM_TBT_DATA structure)
T:pPSnnn – BPM profile time slice data
(TEVATRON_BPM_TIME_SLICE_DATA structure)

T:LPSnnn – BLM profile time slice data
(TEVATRON_BLM_TIME_SLICE_DATA structure)
T:pFSnnn – BPM fast abort time slice data
(TEVATRON_BPM_TIME_SLICE_DATA structure)
T:LFSnnn – BLM fast abort time slice data
(TEVATRON_BLM_TIME_SLICE_DATA structure)
T:pSSnnn – BPM slow abort time slice data
(TEVATRON_BPM_TIME_SLICE_DATA structure)
T:LSSnnn – BLM slow abort time slice data
(TEVATRON_BLM_TIME_SLICE_DATA structure)
T:pPnnn – fast time plot proton BPM device
T:pPnnnI – fast time plot proton BPM I device
T:pPnnnQ – fast time plot proton BPM Q device
T:pAnnn – fast time plot antiproton BPM device
T:pAnnnI – fast time plot antiproton BPM I device
T:pAnnnQ – fast time plot antiproton BPM Q device
T:LMnnn – fast time plot BLM device

Appendix

Current BPM data structures

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;
```

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;
```

BLM 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;
```

Change Log

Version	Issue Date	Description of Change
0.3	04/15/04	Minor updates and added signoff section

TBPMrevdoc1060.doc

Concurrence

Following persons reviewed and concurred with the content of this document.

Steve Wolbers, Project Manager (date) 4/29/04

Bob Webber, Deputy Project Manager (date) 4/29

Jim Steimel, Technical Coordinator (date) 4/29/04

Vince Pavlicek, Subsystem manager (date)

Margaret Votava, Subsystem manager (date) 4/29/04

Robert Kutschke, Subsystem manager (date) 4/29/04

Mike Martens (date) 4/29/04

Bakul Banerjee Page 2 4/26/2004