Recycler Ring Beam Position Monitor

Embedded Software Description

including application in
Main Injector Beam Lines

Revised February 12, 2003
DCV

Contents

Contents

1.0 Introduction

Software for the beam position monitor front-end computers consists of the standard MOOC/Acnet package and a custom developed **Bpm** package. The MOOC/Acnet package supports common control system communication functions including:

- repetitive readings of position data,
- fast time plots of position data,
- reading of circular buffers of selected position data,
- setting and read back of selected operational parameters.

The **Bpm** package provides real-time data acquisition, operating mode coordination, and data scaling and access methods. The real-time component of this package implements the five required operational modes plus one commissioning and one diagnostic mode for the Recycler Ring:

- Background Flash mode,
- Flash mode,
- Closed Orbit (Display) mode,
- Turn-by-turn mode,
- Turn-by-turn Scan mode,
- Repetitive Flash mode (commissioning),
- BPM Delay Sweep mode (diagnostic).

An additional mode is provided to support operation in beam line configurations for the Main Injector:

- Beam Line Flash mode.

These BPM modes are realized in the software by correctly configuring the hardware subsystems and collecting digitized data within specified timing constraints.

The BPM hardware and software are modular and easily configured for operation under widely varying requirements. To illustrate this point, the Recycler BPM system has been successfully employed in the Main Injector A1, P1 and Abort beam lines. This document includes information on both Recycler and Main Injector applications of the BPM software. Notation will be provided where Main Injector beam line support differs from the Recycler modes.

2.0 Architecture

The Recycler Ring Beam Position Monitor (BPM) front-end computer consists of a three slot VME chassis containing an MVME162-512A single board computer (CPU) from Motorola Computer Group and a VIPC616 Industrial I/O Pack carrier board from SBS GreenSpring Computers. The third VME slot is occupied by a UCD signal breakout pannel. BPM application specific I/O is provided by a combination of commercial off-the-shelf and in-house developed Industrial I/O Pack modules mounted on the two VME boards. The MVME162-512A houses two IP320 multiplexed analog to digital converter

modules manufactured by Acromag, Inc. and one IPUCD TCLK/MDAT decoder developed by Fermilab and manufactured by BiRa Systems, Inc. The VIPC616 holds four IPTSG timing signal generator modules also developed by Fermilab and manufactured by BiRa Systems, Inc. The BPM front-end computer is connected through an external Transition Chassis to external BPM Analog Processing Modules. The Transition Chassis provides electrical signal conditioning and signal routing between the BPM front-end computer and the Analog Processing Modules. The Analog Processing Modules contain multiple log and difference amplifiers followed by track-and-hold stages to convert the A and B beam pickup signals into the position proportional form log A - log B.

The IPUCD and IPTSG modules decode the Recycler Ring Beam Synchronization clock (RRBS or BSYNC), the Machine Data (MDAT) system and the Tevatron Clock (TCLK) signals to provide the BPM software and hardware with Recycler timing and synchronization information. The decoded timing signals generate interrupts for the software, and are routed as electrical signals to the multiplexed analog to digital converters and to the external Analog Processing Modules. The timing signal generators tell the track-and-hold when to hold the processed position signal and then tell the multiplexed analog to digital converters to begin the conversion process. Finally interrupt driven software reads the multiplexed analog to digital converter data and applies a degree five polynomial correction to produce position values in engineering units of mm.

From the BPM software perspective (Figure 1.) the system appears to consist of two input/output hardware subsystems: a Timing Signal Generator (TSG) and a Multiplexed Analog to Digital Converter (MADC). The TSG accepts BSYNC, MDAT and external trigger signals and produces beam synchronous timing strobes that are sent to the external Analog Processing Module and to the MADC. The TSG also generates interrupts that inform the CPU of trigger activity. The MADC accepts beam position proportional analog signals from the external Analog Processing Modules and converts them into digital form for software consumption. The UCD TCLK decoder provides timing and event information for the MOOC/Acnet software package and does not directly affect the BPM software.

Recycler Ring BPM Architecture

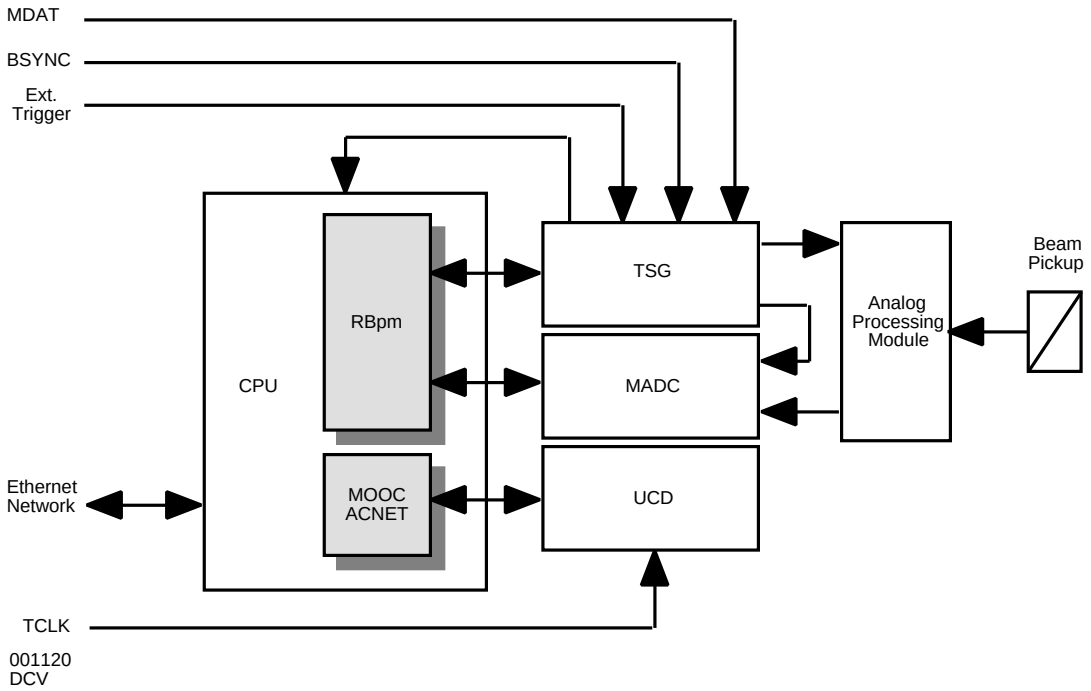

Figure 1.

2.1 Timing Signal Generator Subsystem

The TSG subsystem consists of four[1] IPTSG modules (Figure 2.) each containing a single message MDAT decoder, an external trigger input, a two event BSYNC decoder, a turn counter, twenty delay timer channels and interrupt generator circuitry. The TSG sources 53 MHz synchronous acquisition strobes for the CPU, the MADC and the external Analog Processing Modules after a specified delay from a specified trigger event. During normal operation the trigger event is a specified sequence of BSYNC events, during system commissioning the triggering event comes from the external trigger input. Selection of specific BSYNC events and whether the trigger source is the turn counter, the BSYNC start event decoder or the external trigger input is under software control.

[1] Four TSG modules are used to provide the required number of timing channels within the constraint of 20 channels per Industrial I/O Pack. The use of multiple modules has implications on the TSG driver software, particularly in the flash modes, which must assure that trigger mode initialization occurs atomically for all modules between beam revolution events.

Simplified Recycler Ring BPM Timing Signal Generator

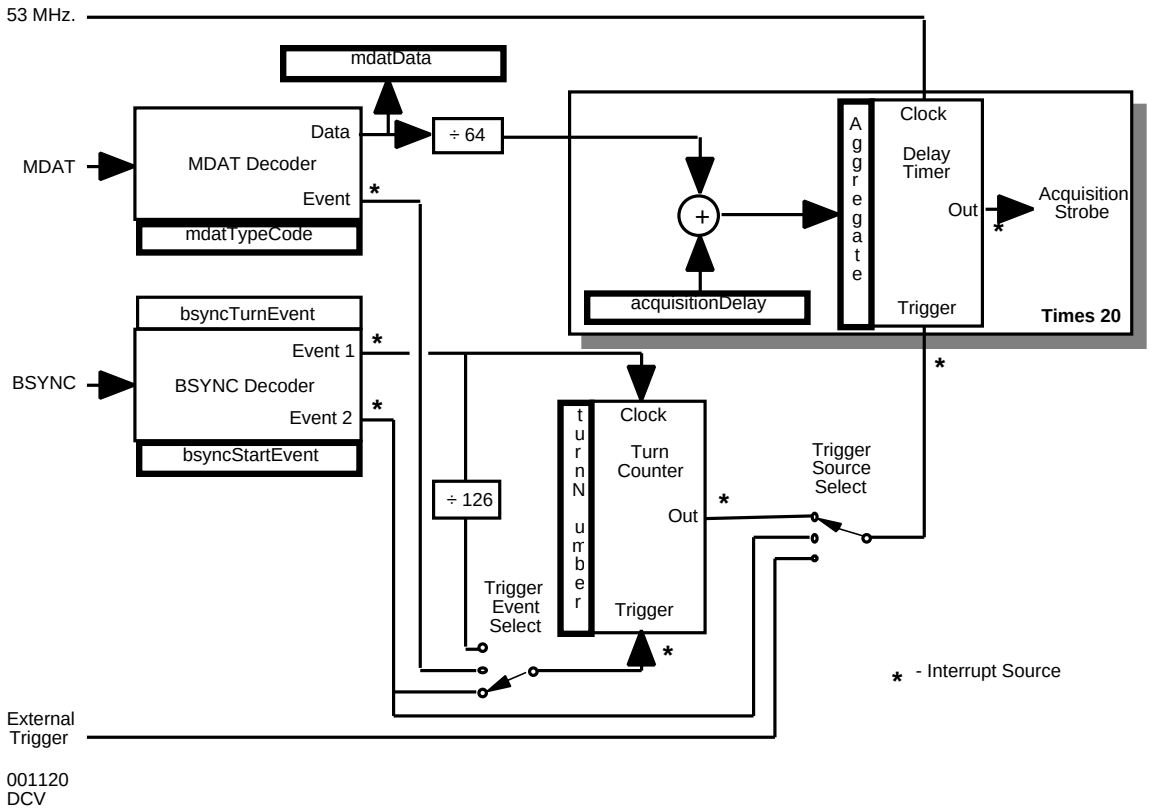

Figure 2.

To configure the TSG to produce BSYNC initiated trigger events the turn counter is programmed to generate delay timer triggers when a specified number of beam turn marker events has occurred after receipt of a specific BSYNC start event. To configure the TSG to produce externally initiated trigger events the BSYNC decoder and turn counter are disabled and the external trigger is routed directly to the twenty delay timer channels.

The generated delay between the selected trigger event and assertion of the acquisition strobes (Figure 3.) is the sum two delay parameters. The first delay, **mdatDelay**, specifies the azimuthal position of the beam to be sampled (i.e., cooled, stacked or injected) and comes from one of six possible MDAT messages containing barrier bucket position information.

Recycler Ring BPM Acquisition Timing

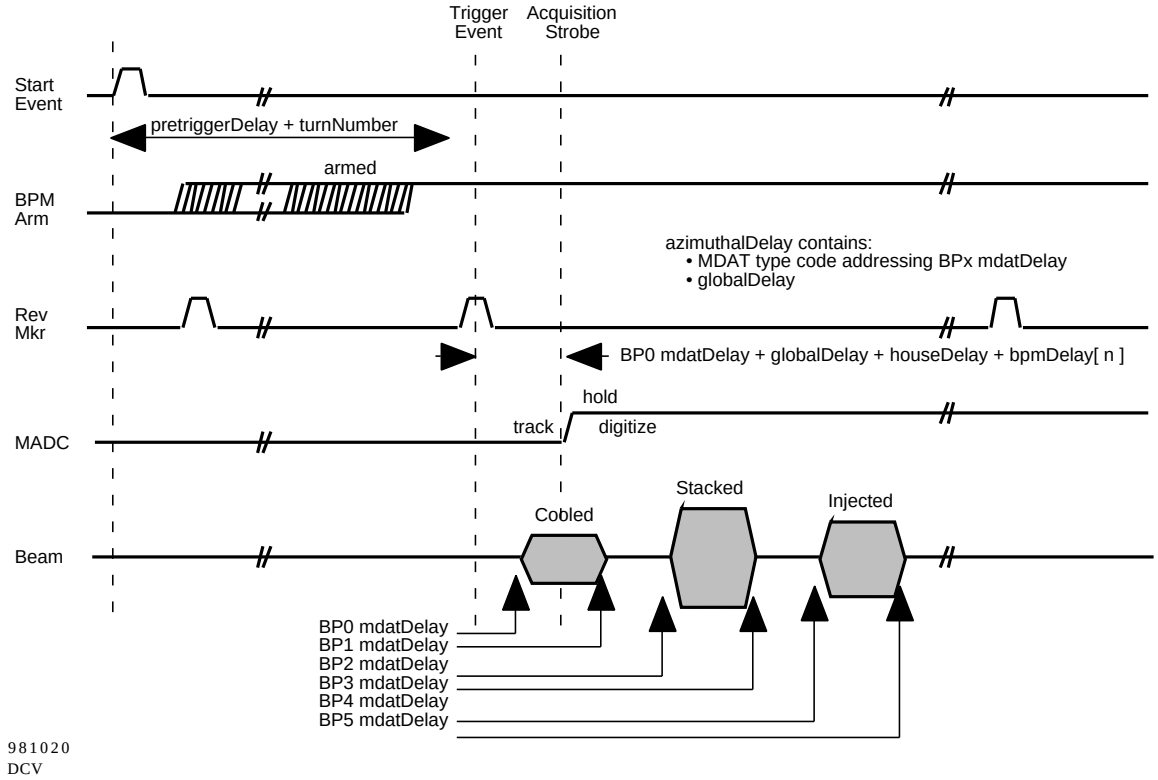

Figure 3.

The barrier bucket position is expressed in 53 MHz cycles and is encoded into 16 bit unsigned integers with a full scale value of 588 * 64 cycles. This encoding scheme provides subbucket resolution and provides for scaling to integer cycles through a simple shift operation. The second delay, **acquisitionDelay** , is the software generated sum of three system delay terms:

1 - **globalDelay** , is the same for all BPM systems and is adjusted to position the data sample time between the barrier buckets delineated by the MDAT messages,
2 - **houseDelay** , varies from house to house and compensates for different relative BSYNC turn marker to beam times around the ring,
3 - **bpmDelay,** varies from BPM to BPM and compensates for differences in signal cable timing.

Channel to channel variation in BPM signal timing dictates that there be one **acquisitionDelay** register per channel. The content of each **acquisitionDelay** register is added by the TSG hardware to the single scaled **mdatDelay** value to produce the individual delay timer **aggregate** values. The **mdatTypeCode** register and the **acquisitionDelay** registers are under software control, the **aggregate** register can be read by software but not written.

2.2 Multiplexed Analog to Digital Converter Subsystem

The MADC subsystem consists of two[2] IP320 MADC modules with forty channels each. Each MADC module consists of a forty channel analog multiplexer followed by a sample and hold and a twelve bit analog to digital converter capable of digitizing at a 100 kHz rate. The ADC conversion can be triggered externally by the TSG or under software control. Channel values can be read singly, sequentially or randomly by the CPU.

3.0 Operating Modes

The **Bpm** software package supports BPM system initialization and implements five operating modes one commissioning mode and one diagnostic mode for the Recycler, and one operating mode for the Main Injector. Initialization occurs at power-up or upon hardware reset and initializes all hardware subsystems and software data structures. The Background Flash mode is the normal operating mode for the Recycler system, regularly taking flash-type measurements to provide (nearly) continuous fast time plotting of the beam position. For the Recycler the Flash, Closed Orbit, Turn-by-turn and Turn-by-turn Scan modes all produce one-shot position measurements with specified timing characteristics. The Repetitive Flash mode is used for commissioning and the BPM Delay Sweep Mode is a diagnostic that aids in determining correct delay settings for individual BPM channels. In the Main Injector beam line application the code initializes and transitions into the Beam Line Repetitive Flash mode where it remains indefinitely.

3.1 Recycler - Background Flash Mode

The Background Flash mode is the default or background operational mode of the system in the Recycler. The BPM begins operating in this mode at power-up and returns to this mode whenever it is not not processing one of the other modes. The Background Flash mode may be restarted upon receipt of a command from the control system. The algorithm acquires data at 720 Hz, synchronous with MDAT messaging, from all horizontal and vertical channel pairs until restarted.

BackgroundFlash(azimuthalDelay)

azimuthalDelay - specifies the base delay between the BSYNC turn marker event and assertion of the hold command to the external Analog Processing Modules. It is a 32 bit integer consisting of the encoded combination of the MDAT type code of the MDAT message containing **mdatDelay**, and the global delay term **globalDelay**. The MDAT type code is right justified in the upper 16 bit word and the global delay is right justified in the lower 16 bit word:

azimuthalDelay = MDAT type code << 16 | global delay.

---

[2] Two MADC modules are used to provide eighty channels of position data grouped as forty horizontal and 40 vertical channels. This configuration is convenient in supporting fast single channel pair acquisition in the turn-by-turn modes.

The MDAT type code is a unitless MDAT message identifier with a valid range of 0 to 255. The global delay is in units of 53 MHz cycles with a valid range of 0 to 588.

Upon entering this mode the software sets the TSG **acquisitionDelay** registers and configures the TSG to respond to the first BSYNC 0xC0 turn marker event after every MDAT message specified in the **azimuthalDelay** parameter. TSG configuration is processed synchronously with the BSYNC turn marker to guarantee initial synchronization of the four TSG modules. This is an important consideration in this mode since there is no explicit start event with which to synchronize the TSGs. The TSG response to each MDAT message/turn marker event combination is to generate an analog processing module hold signal and an interrupt for the data acquisition software. No MADC acquisition trigger is produced. For each of these (720 Hz) interrupts the software acquires data from all forty channel pairs. Once every 720 interrupts the software also calibrates the MADC for gain and offset. If a 720 Hz interrupt is not received within one second of initialization a time-out error is generated.

The channel data acquired by this mode are available individually for regular control system requests (e.g., fast time plots).

3.2 Recycler - Flash Mode

The Flash mode is only entered upon receipt of a command from the control system. This is not a persistent operating mode, it processes a single flash measurement and restores the Background Flash mode when completed. The algorithm acquires data from all horizontal and vertical channel pairs for a single turn.

Flash(azimuthalDelay, startEvent, turnNumber)

azimuthalDelay - specifies the base delay between the BSYNC turn marker event and assertion of the hold command to the external Analog Processing Modules. It is a 32 bit integer consisting of the encoded combination of the MDAT type code of the MDAT message containing **mdatDelay**, and the global delay term **globalDelay**. The MDAT type code is right justified in the upper 16 bit word and the global delay is right justified in the lower 16 bit word:

azimuthalDelay = MDAT type code << 16 | global delay.

The MDAT type code is a unitless MDAT message identifier with a valid range of 0 to 255. The global delay is in units of 53 MHz cycles with a valid range of 0 to 588.

startEvent - specifies the BSYNC event # to be used as a start trigger for data acquisition. It is a 32 bit integer with units of event# and a valid range of 0 to 255.

turnNumber - specifies the number of the turn to be collected. It is a 32 bit integer with units of turn# and a valid range of 1 to 65,535.

Upon entering this mode the software sets the TSG **acquisitionDelay** registers, configures the TSG MDAT decoder to receive the message specified in

the **azimuthalDelay** parameter and configures the TSG to respond to the nth BSYNC 0xC0 turn marker event after the next BSYNC **startEvent** event, where n is the **turnNumber** parameter described above. The TSG response to the selected turn marker event is to generate an analog processing module hold signal and an interrupt for the data acquisition software. No MADC acquisition trigger is produced. Upon receiving the interrupt the software acquires data from all forty channel pairs and calibrates the MADC for gain and offset. If the start event specified by **startEvent** is not received within two minutes a time-out error is generated.

The data acquired by this mode are held for control system requests until the next Flash request is processed. A circular buffer containing the most recent 100 flash samples provides historical position information for control system application program requests. The circular buffer information is stored in battery backed up RAM memory so that it will survive power and reset cycles of the CPU.

3.3 Recycler - Closed Orbit Mode

The Closed Orbit mode is only entered upon receipt of a command from the control system. This is not a persistent operating mode, it processes a single closed orbit measurement and restores the Background Flash mode when completed. The algorithm takes a specified number of samples from all horizontal and vertical channel pairs at approximately 720 Hz synchronous with MDAT messaging. The measurement is triggered by the 0xDA BSYNC event which synchronizes the beginning of the measurement across all BPM computer locations.

The measurement produces averaged orbit values as well as the RMS value of the AC component of the orbit. The Closed Orbit evaluation proceeds as follows:

Closed Orbit:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

, and

Closed Orbit AC component RMS:

$$\sigma = \left(\frac{1}{N}\sum_{i=1}^{N} x_i^2 - \overline{x}^2\right)^{1/2}$$

where N is the number of requested samples.

ClosedOrbit(azimuthalDelay, numSamples)

azimuthalDelay - specifies the base delay between the BSYNC turn marker event and assertion of the hold command to the external Analog Processing Modules. It is a 32 bit integer consisting of the encoded combination of the MDAT type code of the MDAT message containing

mdatDelay, and the global delay term **globalDelay**. The MDAT type code is right justified in the upper 16 bit word and the global delay is right justified in the lower 16 bit word:

azimuthalDelay = MDAT type code << 16 | global delay.

The MDAT type code is a unitless MDAT message identifier with a valid range of 0 to 255. The global delay is in units of 53 MHz cycles with a valid range of 0 to 588.

numSamples - specifies the number of turns to be collected and averaged (typically 20-40). It is a 32 bit integer with units of turns and a valid range of 1 to 128.

Upon entering this mode the software waits for the 0xDA BSYNC event, restarts the background flash mode with the new **azimuthalDelay** parameter value and simply prosesses **numSamples** background flash measurements. When averaging and RMS calculations are completed the software restarts the background flash operation with its original azimuthal delay values. If the 0xDA BSYNC event is not received within two minutes a time-out error is generated.

The position and RMS data acquired by this mode are held for control system application program requests until the next Closed Orbit request is processed. A circular buffer containing the most recent 100 averaged closed orbit samples provides historical orbit information for control system application program requests. The RMS data are not retained in a circular buffer and must be requested before another closed orbit measurement is made or they will be lost. The circular buffer information is stored in battery backed up RAM memory so that it will survive power and reset cycles of the CPU.

3.4 Recycler - Turn-by-turn Mode

The Turn-by-turn mode is only entered upon receipt of a command from the control system. This is not a persistent operating mode, it processes a single turn-by-turn measurement and restores the Background Flash mode when completed. The algorithm acquires data from a single horizontal and vertical channel pair over consecutive turn samples.

TurnByTurn(azimuthalDelay, startEvent, beginTurn, numTurns, horizChannel, vertChannel)

azimuthalDelay - specifies the base delay between the BSYNC turn marker event and assertion of the hold command to the external Analog Processing Modules. It is a 32 bit integer consisting of the encoded combination of the MDAT type code of the MDAT message containing **mdatDelay**, and the global delay term **globalDelay**. The MDAT type code is right justified in the upper 16 bit word and the global delay is right justified in the lower 16 bit word:

azimuthalDelay = MDAT type code << 16 | global delay.

The MDAT type code is a unitless MDAT message identifier with a valid range of 0 to 255. The global delay is in units of 53 MHz cycles with a valid range of 0 to 588.

startEvent - specifies the BSYNC event # to be used as a start trigger for data acquisition. It is a 32 bit integer with units of event# and a valid range of 0 to 255.

beginTurn - specifies the number of the first turn to be acquired. It is a 32 bit integer with units of turn# and a valid range of 1 to 127.

numTurns - specifies the number of turns to be collected. It is a 32 bit integer with units of turns and a valid range of 1 to 1024.

horizChannel - specifies the horizontal channel number for the horizontal/vertical pair to be acquired. It is a 32 bit integer with units of channel# and a valid range of 0 to 39.

vertChannel - specifies the vertical channel number for the horizontal/vertical pair to be acquired It is a 32 bit integer with units of channel# and a valid range of 0 to 39.

Upon entering this mode the software sets the TSG **acquisitionDelay** registers, configures the TSG MDAT decoder to receive the message specified in the **azimuthalDelay** parameter and configures the TSG to respond to the nth and all subsequent BSYNC 0xC0 turn marker events after the next **startEvent** BSYNC event, where n is the **beginTurn** parameter described above. The TSG response is to generate an interrupt for the data acquisition software upon receipt of **startEvent**, and generate analog processing module hold signals followed by MADC acquisition triggers after each turn marker. Upon receiving the start event interrupt the software enters a tight loop in which it acquires data from channels **horizChannel** and **vertChannel** on each beam revolution. After **numTurns** turns have been collected the software disables the TSG's turn marker response. If the event specified by **startEvent** is not received within two minutes a time-out error is generated.

The data acquired by this mode are held for control system application program requests until the next Turn-by-turn request is processed. A circular buffer containing the most recent 100 turn-by-turn samples provides historical position information for control system application program requests. The circular buffer information is stored in battery backed up RAM memory so that it will survive power and reset cycles of the CPU.

3.5 Recycler - Turn-by-turn Scan Mode

The Turn-by-turn Scan mode is only entered upon receipt of a command from the control system. This is not a persistent operating mode, it processes a single turn-by-turn scan measurement and restores the Background Flash mode when completed. The algorithm sequentially executes the turn-by-turn measurement described above for each of the forty BPM channel pairs.

TurnByTurnScan(azimuthalDelay, pingEvent, pingSpacing, beginTurn, numTurns)

azimuthalDelay - specifies the base delay between the BSYNC turn marker event and assertion of the hold command to the external Analog Processing Modules. It is a 32 bit integer consisting of the encoded

combination of the MDAT type code of the MDAT message containing **mdatDelay**, and the global delay term **globalDelay**. The MDAT type code is right justified in the upper 16 bit word and the global delay is right justified in the lower 16 bit word:

azimuthalDelay = MDAT type code << 16 | global delay.

The MDAT type code is a unitless MDAT message identifier with a valid range of 0 to 255. The global delay is in units of 53 MHz cycles with a valid range of 0 to 588.

pingEvent - specifies the BSYNC event # to be used as a start trigger for each channel pair's data acquisition. It is a 32 bit integer with units of event# and a valid range of 0 to 255.

pingSpacing - specifies the (approximate -- intended) time between ping events. It is a 32 bit integer with units of milliseconds and a valid range of 0 to 65535.

beginTurn - specifies the number of the first turn to be acquired. It is a 32 bit integer with units of turn# and a valid range of 1 to 127.

numTurns - specifies the number of turns to be collected. It is a 32 bit integer with units of turns and a valid range of 1 to 1024.

This mode requires no special initial hardware configuration. The algorithm simply calls **TurnByTurn(azimuthalDelay, pingEvent**, **beginTurn**, **numTurns,** scanChannel, scanChannel **)** inside an indexed loop, where scanChannel is the loop index. If **pingSpacing** is greater than sixty-seven milliseconds the software will restart the background flash for an appropriate period between scan iterations. This action permits the background flash to provide data for fast time plots with minimal interruption.

The 40 scan data samples acquired by this mode are held for control system application program requests until the next Turn-by-turn Scan request is processed.

3.6 Recycler - BPM Delay Sweep Mode

The BPM Delay Sweep mode is only entered upon receipt of a command from the control system. This is not a persistent operating mode, it processes a single BPM delay sweep measurement and restores the Background Flash mode when completed. The algorithm sweeps the acquisition delay value from zero to 1024 taking averaged turn-by-turn type measurements at each point.

BpmDelaySweep(azimuthalDelay, startEvent, beginTurn, numTurns, horizChannel, vertChannel)

azimuthalDelay - specifies the base delay between the BSYNC turn marker event and assertion of the hold command to the external Analog Processing Modules. Unlike other modes the MDAT type code is not included in **azimuthalDelay.** It is a 32 bit integer containing the global delay term **globalDelay** which is in units of 53 MHz cycles with a valid range of 0 to 588.

startEvent - specifies the BSYNC event # to be used as a start trigger for data acquisition. It is a 32 bit integer with units of event# and a valid range of 0 to 255.

beginTurn - specifies the number of the first turn to be acquired. It is a 32 bit integer with units of turn# and a valid range of 1 to 127.

numTurns - specifies the number of turns to be averaged at each delay measurement point. It is a 32 bit integer with units of turns and a valid range of 1 to 10.

horizChannel - specifies the horizontal channel number for the horizontal/vertical pair to be acquired. It is a 32 bit integer with units of channel# and a valid range of 0 to 39.

vertChannel - specifies the vertical channel number for the horizontal/vertical pair to be acquired It is a 32 bit integer with units of channel# and a valid range of 0 to 39.

Upon entering this mode the software sets the TSG **acquisitionDelay** registers, disables the TSG MDAT decoder and configures the TSG to respond to the nth and all subsequent BSYNC 0xC0 turn marker events after the next **startEvent** BSYNC event, where n is the **beginTurn** parameter described above. The TSG response is to generate an interrupt for the data acquisition software upon receipt of **startEvent**, and generate analog processing module hold signals followed by MADC acquisition triggers after each turn marker. Upon receiving the start event interrupt the software enters a tight loop in which it sweeps the TSG acquisitionDelay register value from zero to 1024 averaging **numTurns** samples at each step from channels **horizChannel** and **vertChannel**. After all 1024 data points have been collected the software disables the TSG's turn marker response. If the start event specified by **startEvent** is not received within two minutes a time-out error is generated.

The data acquired by this mode are held for control system application program requests until the next BPM Delay Sweep request is processed.

3.7 Main Injector - Beam Line Repetitive Flash Mode

The Beam Line Repetitive Flash mode is the default or background operational mode of the system in the Main Injector beam lines. The BPM begins operating in this mode immediately after power-up and remains in this mode indefinitely. The only way to restart the Beam Line Repetitive Flash mode is to reboot the BPM computer. Since this is the only mode for beam line BPMs it is generally not called like the Recycler modes, but its interface has the following form:

BeamLineFlash(startEvent, intensityDiscFlag)

startEvent - specifies the BSYNC event # to be used as a start trigger for data acquisition. It is a 32 bit integer with units of event# and a valid range of 0 to 255.

intensityDiscFlag - specifies whether intensity discrimination processing should be activated. It is a 32 bit unitless integer with zero meaning intensity discrimination disabled and non-zero meaning intensity discrimination enabled.

Upon entering this mode the software sets the TSG **acquisitionDelay** registers, disables the TSG MDAT decoder and configures the TSG to respond to the start event specified in the **startEvent** parameter. If the triggerSource engineering parameter is set to external mode by the initialization code the external trigger is enabled thereby overriding the **startEvent** parameter. The TSG response to the selected start event is to generate an analog processing module hold signal and an interrupt for the data acquisition software. No MADC acquisition trigger is produced. Upon receiving the interrupt the software acquires data from all forty channel pairs and calibrates the MADC for gain and offset. After the data are stored in the circular buffer the measurement is rearmed and the BPM awaits the next trigger.

The data acquired by this mode are held for control system requests until the next trigger is processed. A circular buffer containing the most recent 100 flash data samples provides historical position information for control system application program requests. The circular buffer information is stored in battery backed up RAM memory so that it will survive power and reset cycles of the CPU.

4.0 Parameters

Operating mode parameters are stored in battery backed up RAM (BBRAM) on the BPM's CPU module. By placing the parameters in BBRAM the BPM can return to its previous operating mode after power and reboot cycles.

4.1 Operating Mode Control - Physics Parameter

ACNET device R:BPxMC.

Selection of BPM operating mode is accomplished by setting the Operating Mode data type which is an array device with from two to seven 32 bit integer values:

[modeSelector, parameter1, ..., parameter6].

The first value is the Mode Selector:

modeSelector - the desired operating mode as follows:

0 - Abort,
1 - Background Flash (default mode),
2 - Flash,
3 - Closed Orbit,
4 - Turn-by-turn,
5 - Turn-by-turn Scan,
6 - BPM Delay Sweep.

The remaining six values represent the selected mode's parameters in the order defined in sections 3.1 to 3.6 above. Setting mode 0, Abort, will attempt to abort any active measurement. The measurement to be aborted must be in a state of "waiting for trigger" to be aborted, a triggered measurement will run to completion. Setting mode 1, Background Flash mode, is used to restart the background flash activity with new parameter values. Setting any of the other mode selector values causes a one-shot measurement to be made in the associated mode. After the measurement is completed the software will return to the Background Flash mode. For consistency the software requires that all seven values be set on any mode request -- values unused by any given mode must be set to zero.

By way of example, to start a Closed Orbit acquisition one would set the Operating Mode data type as:

[3, 0x005500aa, 20, 0, 0, 0, 0].

Here, mode 3 -- Closed Orbit requires two parameters: **azimuthalDelay** , and **numSamples** . The remaining array values will be unused by the software.

4.2 Start Trigger Source (External/Internal) - Engineering Parameter

ACNET device R:BPxSTS - READ ONLY.

Selection of the start trigger external/BSYNC source is accomplished by setting the Start Trigger Source data type which is a 32 bit integer with two allowable values:

triggerSource - the desired start trigger source as follows:
zero - internal BSYNC/MDAT events (default mode)
one - external input

4.3 Beam Mode (Proton/Antiproton) - Physics Parameter

ACNET device R:BPxMOD.

The BPM specific acquisition trigger delay, **bpmDelay** , is context sensitive with four possible values indexed by the type of beam in the machine (proton/antiproton) and the operating mode (flash/turn-by-turn) of the BPM software. The flash/turn-by-turn selector is implicit in the **operatingMode** command. Selection of the proton/antiproton beam mode is accomplished by setting the Beam Mode data type which is a 32 bit integer with two allowable values:

beamMode - the desired beam type as follows:
zero - Antiproton (default mode)
one - Proton

4.4 House Delay - Engineering Parameter

ACNET device R:BPxHSD - READ ONLY.

The house specific acquisition trigger delay term, **houseDelay**, varies from house to house and compensates for different relative BSYNC turn marker-to-beam times around the ring. Specification of the house specific trigger delay term is accomplished by setting the House Delay data type which is an array consisting of two thirty-two bit integers with units of 53 MHz cycles and a valid range of 0 to 2047. The single dimensional array:

long int houseDelay[2];,

is indexed as:

houseDelay[beamMode],

where:

beamMode - beam type as follows:
zero - Antiproton
one - Proton

4.5 BPM Delay - Engineering Parameter

ACNET device R:BPxAFD - Antiproton Flash Delays - READ ONLY.
ACNET device R:BPxATD - Antiproton Turn-by-turn Delays - READ ONLY.
ACNET device R:BPxPFD - Proton Flash Delays - READ ONLY.
ACNET device R:BPxPTD - Proton Turn-by-turn Delays - READ ONLY.

The BPM specific acquisition trigger delay term, **bpmDelay**, varies from BPM to BPM and compensates for differences in signal cable timing. Specification of the BPM specific trigger delay term is accomplished by setting the BPM Delay data type which is an array consisting of 320 sixteen bit unsigned integers with units of 53 MHz cycles and a valid range of 0 to 2047. The 320 integers form a four dimensional array:

unsigned short int bpmDelay[2] [2] [2] [40];,

indexed as:

bpmDelay[beamMode][delayMode][plane][channel],

where:

beamMode - beam type as follows:
zero - Antiproton
one - Proton
delayMode - operating delay mode as follows:
zero - Flash (background, flash or closed orbit)
one - Turn-by-turn (turn-by-turn or turn-by-turn scan)
plane - BPM orientation plane as follows:
zero - Horizontal
one - Vertical
channel - BPM channel in the range 0..39.

The four dimensional array has been divided into four one dimensional arrays to simplify access by control system application programs. The beamMode and

delayMode indices described above have been removed resulting in four arrays. Each array contains the BPM delay terms for eighty channels (forty horizontal channels followed by forty vertical channels). The four arrays are addressed through the four devices indicated above. Length and offset (both in bytes) may be used to specify delays for individual channels or ranges of channels.

4.6 Pre Trigger Delay - Engineering Parameter

ACNET device R:BPxPTD - READ ONLY.

The Flash, Turn-by-turn and Turn-by-turn Scan modes require a BSYNC start event that signals an upcoming data acquisition event (e.g., injection, extraction or ping). These start events are encoded onto the BSYNC system a programmed (but stable) number of turns before the associated data acquisition event happens to allow for arming of the BPM acquisition system. Selection of the pre trigger delay is accomplished by setting the Pretrigger Delay data type which is a 32 bit integer with units of turns and a valid range of 0 to 63.

4.7 Channel Scaling Coefficients - Engineering Parameter

ACNET device R:BPxSCL - READ ONLY.

The raw position proportional MADC data are converted into calibrated and scaled form with engineering units of mm through application of a fifth-order polynomial. The coefficients are stored in a three dimensional array:

float mmCoefficients[2][40][6];,

is indexed as:

houseDelay[plane][channel][coefficient],

where:

plane - beam type as follows:
- zero - horizontal
- one - vertical

channel - BPM channel in the range 0 to 39

coefficient - polynomial coefficients as follows:
- [0] = x^0, [1] = x^1...

The three dimensional array has been flattened into a single array containing the coefficients for eighty channels (forty horizontal channels followed by forty vertical channels) to simplify access by control system application programs. The single array is addressed through the four array devices indicated above. Length and offset (both in bytes) may be used to specify coefficients for individual channels or ranges of channels.

4.8 Background Flash Azimuthal Delay - MDAT Type Code - Physics Parameter

ACNET device R:BPxADT - MDAT Type Code term.

The azimuthal delay MDAT type code and global delay terms for the regular background flash activity are made available as individual parameters which are monitored for change by the **_BkgFlashControl()** task. When either of these parameters, or the beam mode parameter, changes the **_BkgFlashControl()** task automatically restarts the background flash with the new values. Control of the two terms is accomplished by setting the Background Flash Azimuthal Delay data types which are 32 bit integers. The MDAT type code term is a unitless MDAT message identifier with a valid range of 0 to 255. The global delay term is in units of 53 MHz cycles with a valid range of 0 to 588.

4.9 Background Flash Azimuthal Delay - Global Delay - Physics Parameter

ACNET device R:BPxADG - Global Delay term.

See section 4.8 above.

4.10 BSYNC Turn Event - Engineering Parameter

ACNET device R:BPxTRN - READ ONLY.

All BPM modes require a clock that signals beam revolutions. These turn events are encoded onto the BSYNC system. Selection of the turn event marker is accomplished by setting the BSYNC Turn Event data type which is a 32 bit integer with units of turns and a valid range of 0 to 63.

4.11 Main Injector - Beam Intensity Threshold - Physics Parameter

ACNET device I:BPxBIT.

The beam line BPMs have intensity channels which provide for the possibility of doing intensity descrimination for position data. If intensity descrimination is enabled at start up the first intensity is checked against the intensity threshold parameter value to determine if any data should be collected for any given trigger. If the first channel is over the threshold collection will proceed and each individual channel will be compared against the threshold. The results of the individual threshold check are placed in the **BlFlashData** structure as Boolean flags. The Beam Intensity Threshold data type is a 32 bit float with units of particles per bunch. No bounds checking is done on this parameter.

5.0 Data

5.1 Operating Mode Status

ACNET device R:BPxMS.

The status of the most recent operating mode command is available through the Operating Mode Status data type which is a 32 bit integer consisting of the encoded combination of the measurement status and the mode selector from the most recent Operating Mode Control parameter setting (see 4.1 above). The

measurement status is right justified in the upper signed 16 bit word and the operating mode selector is right justified in the lower 16 bit word:

operatingModeStatus = measurement status << 16 | mode selector.

The status portion of the word is unitless with six possible numeric value ranges:

- negative values indicate an error condition with the following status:
  - -1 - background flash MDAT synchronization interrupt failed,
  - -2 - background flash first 720 Hz MDAT interrupt failed,
  - -3 - measurement interrupt wait timed out,
  - -4 - closed orbit synchronization trigger timed out,
  - -5 - BSYNC turn event not detected,
  - -6 - overflow of one or more channel's acquisition delay,
  - -101 - initialization of horizontal MADC failed,
  - -102 - initialization of vertical MADC failed,
  - -103 - initialization of TSG #0 failed,
  - -104 - initialization of TSG #1 failed,
  - -105 - initialization of TSG #2 failed,
  - -106 - initialization of TSG #3 failed,
  - -107 - initialization of time stamp clock failed,
  - -125 - interrupter TSG module masked out of initialization,
  - -150 - unable to create a VxWorks semaphore,
  - -151 - unable to create a VxWorks interrupt service routine,
  - -152 - unable to create a VxWorks reboot handler,
  - -153 - unable to read engineering parameter file,
  - -154- unable to write engineering parameter file
  - -201 - unable to create background flash control task,
  - -202 - unable to create repetitive flash control task
  - -512 - measurement aborted.
- 32767 indicates that the BPM software is initializing.
- 32766 indicates that the BPM has received an Operating Mode Control command and is awaiting a start event to trigger the requested operation.
- 32765 indicates that a trigger has been received and measurement is in progress.
- positive values in the range 32764-1 indicate the number of measurements remaining to complete the requested operation -- positive values will count down toward zero as the specified operation progresses.
- zero indicates that the requested operation was successfully completed.

The mode selector portion of the word is unitless with a valid range of 0 to 6. Mode zero indicates that the BPM software has not completed initialization and is not actively making measurements. Modes one through six are identical in meaning to the **modeSelector** value of the **operatingMode** data type described in section 4.1. Note that when **operatingModeStatus** is numerically equal to the **modeSelector** of the most recent **operatingMode** parameter setting the BPM software has processed the command and the requested operation has been completed.

By way of example, to start a Closed Orbit acquisition one could set the Operating Mode data type by:

```
INT32 modeSelector = 3;
INT32 azimuthalDelay = 0x00800005;
INT32 numSamples = 100;
SetMode( modeSelector, azimuthalDelay, numSamples, 0, 0, 0, 0 ).
```

To check the operation's status one could use:

```
INT32 status = GetOperatingModeStatus();
if ( status == 0 ) {
  // BPM is still initializing
} else if ( status < 0 ) {
  // some type of error
} else if ( status == modeSelector ) {
  // measurement is completed
} else {
  remainingMeasurements = status >> 16;
  if ( remainingMeasurements == 32767 ) {
    // BPM is waiting for synchronization trigger event
  } else {
    // BPM is counting down to completion of operation
  }
}
```

5.2 Beam Position Data

Position data collected in each of the BPM operating modes are stored in battery backed up random access memory (BBRAM) located on the BPM's CPU module. By placing the data in BBRAM the BPM can retain circular buffer data across power and reboot cycles. Operating mode parameters, status information and scaling factors are stored along with the raw position data. Beam positions in engineering units are evaluated by access method functions as they are requested. Thus, the raw data stored in memory are calibrated and scaled only when needed by the control system.

Two "raw data structure" data types: **RawFlash** and **RawTbT** are defined for storing the data collected during the various modes of operation. For Background Flash mode one **RawFlash** structure is updated as new measurements are taken at a 720 Hz rate. For Flash, Closed Orbit and Beam Line Repetitive Flash modes the associated **RawFlash** structures are placed into circular buffers which provide historical information. For Turn-by-turn mode a **RawTbT** structure is also placed into a circular buffer. Turn-by-turn Scan mode **RawTbT** structures are placed into a buffer with one buffer element for each channel pair in the scan. Due to its sheer volume the Turn-by-turn Scan data are not stored in BBRAM and will be lost after power cycles. It *should* remain available after reboot cycles however.

5.2.1 Background Flash Data

As mentioned above, the Background Flash mode data are stored in the form of a **RawFlash** structure. This data may be requested in whole or as individual channel position data.

5.2.1.1 Most Recent Background Flash Structure

ACNET device R:BPxBFV.

The **RawFlash** background flash data are transformed into a **FlashData** structure and made available to the control system upon request. The **FlashData** structure has the following definition:

```
      struct Flash {    // Flash data structure
        // flags type of data
        EOperatingMode  dataType;
        // VxWorks POSIX style time stamp with resolution to uSeconds
        struct timespec timestamp;
        // zero = OK, negative = error code
        long int        status;
        // zero = antiproton, one = proton
        long int        beamMode;
        // MDAT type code term received in azimuthalDelay parameter
        long int        mdatTypeCode;
        // trigger delay term derived from Mdat
        long int        mdatDelay;
        // trigger delay term received in azimuthalDelay parameter
        long int        globalDelay;
        // startEvent parameter - upper word non-zero = external trigger
        long int        startEvent;
        // Flash mode only - turnNumber parameter
        long int        turnNumber;
        // ClosedOrbit mode only - numSamples parameter
        long int        numSamples;
        // horizontal positions - mm
        float           horizData[ 40 ];
        // vertical positions - mm
        float           vertData[ 40 ];
};
typedef struct Flash FlashData, *FlashDataPtr;
```

Length and offset may be used to access specific fields or ranges of fields if required.

5.2.1.2 Background Flash Position Data

ACNET device R:HPxxxx and R:VPxxxx - where xxxx indicates the associated quadrupole location in the Recycler tunnel.

The background flash position data can be requested on a channel by channel basis. This is the only BPM data type which supports fast time plotting, at rates of up to 720 Hz.

5.2.2 Flash Data

As mentioned above, Flash mode data are stored in a circular buffer in the form of **RawFlash** structures. This data may be requested from the circular buffer or with a special request that accesses the most recent structure.

5.2.2.1 Most Recent Flash Structure

ACNET device R:BPxFLV.

The **RawFlash** flash data are transformed into a **FlashData** structure and made available to the control system upon request. The **FlashData** structure has the form described in 5.2.1.1 above. Length and offset may be used to access specific fields or ranges of fields if required.

5.2.2.2 Flash Circular Buffer

ACNET device R:BPxFLB.

The **RawFlash** flash data stored in the circular buffer are transformed into **FlashData** structures and made available to the control system upon request. The **FlashData** structures have the form described in 5.2.1.1 above. Length is fixed at the byte count of the data structure and offset is used to address the individual buffer entries with offset 0 indicating the most recent data in the buffer.

5.2.3 Closed Orbit Data

As mentioned above, Closed Orbit mode position data are stored in a circular buffer in the form of **RawFlash** structures. This data may be requested from the circular buffer or with a special request that accesses the most recent structure. Each Closed Orbit measurement includes evaluation of the RMS position values. The RMS data are available for the most recent measurement only. See section 3.3 above for a description of the processing involved in evaluating closed orbit and closed orbit RMS values.

5.2.3.1 Most Recent Closed Orbit Structure

ACNET device R:BPxCOV.

The **RawFlash** flash data are transformed into a **FlashData** structure and made available to the control system upon request. The **FlashData** structure has the form described in 5.2.1.1 above. Length and offset may be used to access specific fields or ranges of fields if required.

5.2.3.2 Most Recent Closed Orbit RMS Structure

ACNET device R:BPxCOR.

Each Closed Orbit measurement also evaluates the RMS value of the orbit for each channel. The RMS data are placed into a **FlashData** structure with the form described in 5.2.1.1 above and made available to the control system upon

request. Length and offset may be used to access specific fields or ranges of fields if required.

5.2.3.3 Closed Orbit Circular Buffer

ACNET device R:BPxCOB.

The **RawFlash** flash data stored in the circular buffer are transformed into **FlashData** structures and made available to the control system upon request. The **FlashData** structures have the form described in 5.2.1.1 above. Length is fixed at the byte count of the data structure and offset is used to address the individual buffer entries with offset 0 indicating the most recent data in the buffer.

5.2.4 Turn-by-turn Data

As mentioned above, Turn-by-turn mode data are stored in a circular buffer in the form of **RawTbT** structures. This data may be requested from the circular buffer or with a special request that accesses the most recent structure.

5.2.4.1 Most Recent Turn-by-turn Structure

ACNET device R:BPxTBV.

The **RawTbT** turn-by-turn data are transformed into a **TbTData** structure and made available to the control system upon request. The **TbTData** structure has the following definition:

```
struct TbT {      // Flash data structure
  // flags type of data
  EOperatingMode  dataType;
  // VxWorks POSIX style time stamp with resolution to uSeconds
  struct timespec timestamp;
  // zero = OK, negative = error code
  long int        status;
  // zero = antiproton, one = proton
  long int        beamMode;
  // MDAT type code term received in azimuthalDelay parameter
  long int        mdatTypeCode;
  // trigger delay term derived from Mdat
  long int        mdatDelay;
  // trigger delay term received in azimuthalDelay parameter
  long int        globalDelay;
  // startEvent parameter - upper word non-zero = external trigger
  long int        startEvent;
  // beginTurn parameter
  long int        beginTurn;
  // numTurns parameter
  long int        numTurns;
  // horizChannel parameter
  long int        horizChannel ;
```

```
        // vertChannel parameter
        long int        vertChannel ;
        // horizontal positions - mm
        float           horizData[ 1024 ];
        // vertical positions - mm
        float           vertData[ 1024 ];
};
typedef struct TbT TbTData, *TbTDataPtr;
```

Length and offset may be used to access specific fields or ranges of fields if required.

5.2.4.2 Turn-by-turn Circular Buffer

ACNET device R:BPxTBB.

The **RawTbT** turn-by-turn data stored in the circular buffer are transformed into **TbTData** structures and made available to the control system upon request. The **TbTData** structures have the form described in 5.2.4.1 above. For control system accesses length and offset are used to describe the desired portion of the **TbTData** structure. Length indicates the requested number of bytes as expected. The offset value contains the circular buffer entry number combined with a 256 byte block offset as follows:

offset = block << 8 + entry.

With this scheme all entries in the circular buffer can be addressed directly with entry zero indicating the most recent data in the buffer.

5.2.5 Turn-by-turn Scan Data

As mentioned above, Turn-by-turn Scan mode data are stored in a buffer in the form of **RawTbT** structures. This data may be requested from the buffer or with a special request that accesses the structure from the most recent scan.

5.2.5.1 Most Recent Turn-by-turn Scan Structure

ACNET device R:BPxTSV.

The **RawTbT** turn-by-turn data are transformed into a **TbTData** structure and made available to the control system upon request. The **TbTData** structures have the form described in 5.2.4.1 above. Length and offset may be used to access specific fields or ranges of fields if required.

5.2.5.2 Turn-by-turn Scan Buffer

ACNET device R:BPxTSB.

The **RawTbT** turn-by-turn data stored in the scan buffer are transformed into **TbTData** structures and made available to the control system upon request. The **TbTData** structures have the form described in 5.2.4.1 above.
For control system accesses length and offset are used to describe the desired portion of the **TbTData** structure. Length indicates the requested number of

bytes as expected. The offset value contains the scan buffer entry number combined with a 256 byte block offset as follows:

offset = block << 8 + entry.

With this scheme all entries in the scan buffer can be addressed directly with entry zero indicating data from the first channel pair scan.

5.2.6 BPM Delay Sweep Data

ACNET device R:BPxDSD.

The BPM Delay Sweep data are stored in memory within a **RawTbT** structure which is transformed into a **TbTData** structure of the form described in 5.2.4.1 above when requested. Since there are only 588 points per delay sweep, the structure's horizData and vertData arrays will be only partially populated.

5.2.7 Main Injector - Beam Line Repetitive Flash Data

As mentioned above, the Beam Line Repetitive Flash mode data are stored in the form of a **RawFlash** structure. This data may be requested in whole or as individual channel position data. The **BlFlashData** structure has the following definition:

```
struct BlFlash {  // Flash data structure
  // flags type of data
  EOperatingMode  dataType;
  // VxWorks POSIX style time stamp with resolution to uSeconds
  struct timespec timestamp;
  // zero = OK, negative = error code
  long int        status;
  // zero = antiproton, one = proton
  long int        beamMode;
  // MDAT type code term received in azimuthalDelay parameter
  long int        mdatTypeCode;
  // trigger delay term derived from Mdat
  long int        mdatDelay;
  // trigger delay term received in azimuthalDelay parameter
  long int        globalDelay;
  // startEvent parameter - upper word non-zero = external trigger
  long int        startEvent;
  // Flash mode only - turnNumber parameter
  long int        turnNumber;
  // ClosedOrbit mode only - numSamples parameter
  long int        numSamples;
  // horizontal positions - mm
  float           horizPosData[ 20 ];
  // vertical positions - mm
  float           vertPosData[ 20 ];
  // horizontal intensity - ppb
  float           horizIntData[ 20 ];
  // vertical intensity - ppb
  float           vertIntData[ 20 ];
```

```
        // horizontal intensity status
        char            horizBpmStatus[ 20 ];
        // vertical intensity status
        char            vertBpmStatus[ 20 ];
};
typedef struct BlFlash BlFlashData, *BlFlashDataPtr;
```

Length and offset may be used to access specific fields or ranges of fields if required.

5.2.7.1 Main Injector - Most Recent Beam Line Repetitive Flash Data

ACNET device I:BPxFLV.

The **RawFlash** beam line flash data are transformed into a **BlFlashData** structure and made available to the control system upon request. The **FlashData** structure has the form described in 5.2.7 above. Length and offset may be used to access specific fields or ranges of fields if required.

5.2.7.2 Main Injector - Beam Line Repetitive Flash Circular Buffer

ACNET device I:BPxFLB.

The **RawFlash** beam line flash data stored in the circular buffer are transformed into **BlFlashData** structures and made available to the control system upon request. The **BlFlashData** structures have the form described in 5.2.7 above. Length is fixed at the byte count of the data structure and offset is used to address the individual buffer entries with offset 0 indicating the most recent data in the buffer.

6.0 Software Processing

The **Bpm** software package is distributed across three primary source modules:

Bpm - Initialization and real-time data acquisition,
MoocAcnet - Data processing and MOOC interface,
Commands - Interactive data display & control commands.

The **Bpm** module handles all beam position monitor specific activities and is the heart of the system. The **MoocAcnet** module acts as an interface between the **Bpm** real-time processes and the non real-time activities of MOOC and control system communications. The **Commands** module provides interactive commands to display parameter values, measurement data and system status.

6.1 Initialization and Real-time Data Acquisition Module - **Bpm**

The **Bpm** module contains all code to initialize and process the various operating modes of the BPM system. Various entry points are provided for each BPM target.

6.1.1 Recycler Ring BPMs

RBpm() - General Recycler ring initialization,
BackgroundFlash() - Operating mode initialization and processing,
Flash() - Operating mode initialization and processing,
ClosedOrbit() - Operating mode initialization and processing,
TurnByTurn() - Operating mode initialization and processing,
TurnByTurnScan() - Operating mode initialization and processing.
BpmDelaySweep() - Operating mode initialization and processing.

RBpm() is called once from the VxWorks startup script and is responsible for one-time initialization of the Recycler Ring BPM software system.

RBpm(printInformationals, autoBkgFlash)

printInformationals - specifies whether informational messages are printed on the VxWorks console. It is a 32 bit unitless integer with zero meaning informationals disabled and non-zero meaning informationals enabled.

autoBkgFlash - specifies whether the background flash activity should be started at initialization and automatically restarted as various operational parameter values change (see description of **BkgFlashControl()** immediately below). It is a 32 bit unitless integer with zero meaning auto background flash disabled and non-zero meaning auto background flash enabled.

BackgroundFlash(), Flash(), ClosedOrbit(), TurnByTurn(), TurnByTurnScan() and **BpmDelaySweep()** are called by the **MoocAcnet** module as operating mode commands come in from the various control system application programs. These routines configure the hardware subsystems and read raw beam position data into shared memory structures within real-time constraints. The shared memory data are then available for processing by the **MoocAcnet** module. The call parameters for these five functions are described in sections 3.1 through 3.6 above.

A low priority task, **BkgFlashControl()** , executes once every second to restart the Background Flash activity when the beam mode or background flash azimuthal delay parameters change in value.

6.1.2 Main Injector beam line BPMs

IBpm() - General Main Injector beam line initialization,
BeamLineFlash() - Operating mode initialization and processing.

IBpm() is called once from the VxWorks startup script and is responsible for one-time initialization of the Main Injector beam line BPM software system.

IBpm(printInformationals, intensityDiscFlag, startEvent)

printInformationals - specifies whether informational messages are printed on the VxWorks console. It is a 32 bit unitless integer with zero meaning informationals disabled and non-zero meaning informationals enabled.

intensityDiscFlag - specifies whether intensity descrimination should be performed for this BPM. It is a 32 bit unitless integer with zero meaning intensity descrimination disabled and non-zero meaning intensity descrimination enabled.

startEvent - specifies the BSYNC event # to be used as a start trigger for data acquisition. It is a 32 bit integer with units of event# and a valid range of 0 to 255. Note: If the triggerSource engineering parameter is set to external mode by the initialization code the external trigger is enabled thereby overriding the **startEvent** parameter.

BeamLineFlash() is called by **IBpm()** at startup and is provided with parameter values specified in the VxWorks startup script. Note: Since this mode is only called at startup the only method of restarting the mode is to reboot the BPM.

6.2 Data Processing and MOOC Interface Module - **MoocAcnet**

The **MoocAcnet** module contains all code to make beam position parameters and data available to the control system. This module contains the classes that implement the MOOC objects which are available to the control system. The single entry point, **BpmDevNew()**, creates instances of all MOOC objects in the system.

The **MoocAcnet** classes assure that control system settings and readings are synchronized with **Bpm** module real-time activity to guarantee data coherency. To minimize the real-time computational load BPM data are calibrated and scaled to engineering units when it is requested rather than as it is collected. The **MoocAcnet** module is also responsible for dispatching operating mode commands and assuring that only one operating mode is activated at a time.

6.3 Interactive Data Display & Control - **Commands**

The **Commands** module contains functions that can be called interactively from the VxWorks shell command line interface. The module includes various functions for displaying data and for manipulating parameters including:

dchan() - display most recent measurement data,
ddata() - display most recent measurement data,
dflash() - display flash data,
dparams() - display parameter values,
dstat() - display measurement status,
dtbt() - display turn-by-turn data,
gbpm() - go do BPM initialization task,
gbf | gflash | gco | gtbt | gtbts(p1..pn) - go do measurement task.
helpeng() - print BPM engineering command help,
helpme() - print BPM command help (this text),
pchan() - print background flash channel value,
pts() - print current time stamp,
sbeam() - set beam mode,
sbfad() - set background flash azimuthal delay,

sformat() - set data format for display commands.

The following commands address the special needs of the beam line BPMs:

dbpmstat() - display beam line intensity status,
dline() - display most recent beam line data,
sthreshold() – set beam line intensity threshold.

Several additional commands are provided for manipulating engineering parameters and running test procedures including:

dtiming() – display timing system activity,
helpiptsg() – print IPTSG test command help,
login() - login to host as vxworks_boot,
pcoeff() - print scaling coefficients,
sbdelay() - set individual BPM delay in TSG,
sbpmdelays() - set BPM delays,
scoeff() - set scaling coefficient term,
scoefficients() - set scaling coefficient x^1 term, others to 0.0,
shdelays() - set house delays,
shousedelay() - set house delay in TSG,
spdelay() - set pre trigger delay,
strigger() - set start trigger source,
sturn() - set BSYNC turn event number.

7.0 Software Development Process

The BPM resident software is developed and maintained in a cross development environment residing on the Beams Division development computer called nova.fnal.gov. The code is designed to run under the VxWorks operating system on the MVME162 CPU card. Since the code is maintained in a project called **rbpm** in nova's CVS code repository any front-end developer may make modifications to the project's code base. To make changes to the software the developer must first initialize the UN*X environment for VxWorks development and then get a copy of the rbpm project out of the CVS repository. An outline of this procedure follows below.

To do embedded system development for the MC68040 chip and VxWorks version 5.3.1 as used in the rbpm project you must type: **env_68k** at the shell prompt. This sources a script provided by the UN*X sysadmin folks to set up the gnu tools for VxWorks. Don't forget this step each time you log in to do **rbpm** development.

To get started with the **rbpm** project you should do the following:
- move to your your 'sandbox' area and do a CVS checkout of rbpm.
- cd into the newly created **rbpm** directory.
- Type: make. This will make the project in your sandbox. The product is called librbpm.out.

To install librbpm.out from your sandbox onto fecode-bd simply type one of:
make development,
make test or

```
make production.
```

There are several install rules in the Makefile that you can use for installing various files such as the startup script. Read the file Makefile in your rbpm sandbox to learn more.

END