



Fermilab/BD/TEV  
Beams-doc-1276  
August 17, 2004  
Version 00

# **Tevatron Beam Position Monitor Front End Software User's Guide**

## **DRAFT DRAFT DRAFT**

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

### ***Abstract***

This document is geared to help past developers remember what they did, future developers to know how to build and distribute the software, commissioners to understand how to configure a system, and maintainers to help diagnose software problems.

## **1 Overview**

This document is intended to be a guide for the front end users of the Tevatron BPM system. It is meant to help

- developers build system and verify the builds
- commissioners now how install and commission boards and crates
- maintainers diagnose problems

## 2 Commissioning

### 2.1 Crate Controller Configuration

The crate controller in a TBPM house is a Motorola 2400 running vxWorks. Node names are selected with the convention of `tbpm<id>` where `<id>` is a two character identifier of location in the ring, e.g., `tbpma3`.

#### 2.1.1 Registration

Generic instructions can be found at:

[http://www-bd.fnal.gov/controls/micro\\_p/rom\\_install.html](http://www-bd.fnal.gov/controls/micro_p/rom_install.html).

IP addresses must be obtained through the Acceleration Division networking group. Before requesting an IP address, you must know the Fermilab ID and Ethernet MAC address of the board. Once that has been identified, register the board at:

<http://www-bdnew.fnal.gov/Netwebrequests/net-connection.asp>

Once the IP address is obtained, each host will also need an ACNET address, which consists of the above node name plus an assigned trunk and node id. Contact Brian Hendricks to procure a trunk and node id.

#### 2.1.2 Boot Parameters

Boards are currently running VxWorks v5.5. Each crate controller should have the following boot parameters:

```
boot device          : dc0
processor number     : 0
host name            : fecode-bd
file name            : vxworks_boot/kernel/mv2400-512MB/vxWorks-a32-512
inet on ethernet (e) : 131.225.<xxx>.<yyy>:ffffff00
inet on backplane (b):
host inet (h)        : 131.225.121.145
gateway inet (g)     : 131.225.<xxx>.200
user (u)             : vxworks_boot
ftp password (pw) (blank = use rsh):
flags (f)            : 0x0
target name (tn)     : <nodename>_0x<trunkid><nodeid>
startup script (s)   : vxworks_boot/fe/tpm/tpmstartup
other (o)            :
```

Where `<xxx>`, `<yyy>`, `<trunkid>`, and `<nodeid>` are obtained in the registration step above.

Note that this is a custom kernel – it specifically adds the gateway node to the route table with the following command:

```
routeAdd "0.0.0.0", "131.225.<xxx>.200"
```

This is done so that all houses can use the same startup script.

### 2.1.3 Startup Script

The Tevatron BPM software is designed to autodetect hardware and addresses at run time, so all BPMs can share the same startup script. The startup script pointed to in the boot parameters currently contains:

```

#-----
#--- General initialization
#-----
shellPromptSet( "Booting incomplete->" )
taskPrioritySet( taskIdFigure( "tExcTask" ), 1 )
taskPrioritySet( taskIdFigure( "tLogTask" ), 250 )
# get target name in 6 characters
targetName=malloc(20)
gethostname(targetName,20)
targetName[6]=0

#-----
# mount NFS file systems - 1217, 5143 is user vxworks_boot group bdmicrop
#-----
topLevel="vxworks_write/fe/tbpm"
outputDir=malloc (strlen(topLevel) + strlen(targetName))
strcpy(outputDir,topLevel)
strcat(outputDir,targetName)
nfsMount( "fecode-bd", "vxworks_boot/module/PPC604", "/controls" )
nfsMount( "fecode-bd", "vxworks_boot/fe", "/fe" )
nfsMount( "fecode-bd", outputDir, "/write" )
nfsAuthUnixSet( "fecode-bd", 1217, 5143, 0, 0 )

#-----
#--- Load all the code
#-----
# MOOC and ACNET services
ld < /controls/libpctrig-latest.o
ld < /controls/libssm-2.3.o
ld < /controls/acnet-1.021.o
ld < /controls/libmooc-3.6.o
# what is this?
# tbpm stuff
ld < vxworks_boot/fe/rfiinst/devlib/VW_54/MVME2434/libmiscutil.out
#bpm stuff - these locations need to change
ld < vxworks_boot/fe/tbpm/libecdr814multibrd.out
# Warren's ECSG-1R3ADC-PMC I/O drivers
ld < vxworks_boot/fe/rfiinst/lib/VW_54/MVME2434/libecsg1r3adcfermi.out

ld < /fe/gbpm/libgbpm.out
ld < /fe/tbpm/libtbpm.out

#-----
#--- Start things
# THESE LOCATIONS NEEDED TO CHANGE!!!!
#-----
# set the downloadable file
set_rbf_file ("vxworks_boot/fe/rbpm/controls/pmcucd.rbf");
# download the techno box; 0 is the instance or slot
pmcucd_start(0)
# set the addresses for ucd
pmcucd_set_adr (get_alteramem(0))
pmcucd_set_madr (get_alteramem(0)+0x1000)

initssc (0,calloc(0x10000,1))
MocNew()

# get .ini and .ch files for echotek configuration
#
cd( "/fe/tbpm/ini" )
ecdr814gcReadSetup("53MHzNarrowBand.ini", 0);
ecdr814gcReadSetup("53MHzEnsemble.ini", 1);
#ecdr814gcReadSetup("rawDmaD64.ini", 2);
#ecdr814gcReadSetup("rawDmaD64.ini", 3);

```

```
#ecdr814gcReadSetup("rawDmaD64.ini",      4);
#ecdr814gcReadSetup("rawDmaD64.ini",      5);
#ecdr814gcReadSetup("rawDmaD64.ini",      6);
#ecdr814gcReadSetup("count.ini",         6);
ecdr814gcReadSetup("rawDmaD64.ini",      7);
ecdr814gcReadSetup("rawDmaD64Long.ini",   8);
ecdr814gcShowSetupAll();

MoocDeviceDownLoad( 0 )

CreateEventAction("s,146764,1,0,*",printf,"v:cldrst went to %d\n")

#-----
#-- Finish up
#-----
prompt = malloc(30)
strcpy (prompt,targetName)
strcat (prompt,":tbpm> ")
shellPromptSet ( prompt )
free(prompt)
free(targetName)
free(outputDir)

#-----
#-- Start trace buffer
#-----
*0xfef80020=0x63ce0001
traceInit 8000,10,0xfef80100
traceMode 1
traceOn 0,0,20,0

#-----
# Start the readout
#-----

bpmStart
```

### 2.1.4 Changing the vxWorks startup script for all houses

The tevatron BPM houses share a common startup script that resides in `nova.fnal.gov:/fecodde-bd/vxworks_boot/fe/tbpm`; however, it is maintained in the `tbpm` CVS module. If the startup script for all houses need to change, please follow the build instructions in subsequent section.

Note that we are loading fixed versions of the ACNET/MOOC libraries. These versions need to match the versions of ACNET/MOOC header files that are compiled into both `gbpm` and `tbpm`. When reloading a different version of mooc, you will also need to edit the `_acnetheaders.h` file in both packages and recompile (see build instructions below).

### 2.1.5 Changing the vxWorks startup script for a particular house

While debugging problems at a specific house, there will sometimes be a need to have a startup script that is specific to that particular house. Since this is only a temporary state, there is no need to track the changes. Simply:

```
nova> cd /fecode-bd/vxworks_boot/fe/tbpm
nova> cp tbpmstartup temp_startup
```

Edit the temp\_startup script as necessary, being sure to change the vxworks boot parameters to use this script. Please remember to reset the boot parameters to use the canonical script when done.

## 2.2 Echotek Board Configuration

The current echotek driver supports up to 20 different setup files that are specified in the startup script. Different configurations are used based on the mode of operation of the software. See document #860 for a description of the operational modes. In a particular mode, all echotech modules in a house will use the same setup file, ie there is no provision for allowing the boards to be configured differently. By convention, these setup files have a extension of .ini.

Additionally, each board setup file contains another pointer to a file that contains information regarding a channel setup. Unless you really know what you're doing, all channels should be configured with the same channel. By convention, the channel setup files have an extension of .ch. By convention, the <abc>.ini file will read in a channel file named <abc>.ch.

The driver reads the associated files from the current working directory, so it's important to cd to this directory in the vxWorks startup script.

Operational Mode	Setup file index
Closed orbit	
Turn X Turn	
First injection	

## 2.3 Timing Module Configuration

### 2.3.1 Recycler Timing Board

Additional software that needs to be added to the startup script to run the recycler timing board are:

```
# class library to initialize the techno box
# only needed for
ld (1, 1, "vxworks_boot/fe/rbpm/controls/pmcclassLib_mv2400-lastest.o");
# void ecsg( int fref, int n, int p, int r, int b, int a, int c, int d );
# request 73.5 MHz and get 73.444444 MHz
ecsg( 53104696, 504, 8, 40, 63, 0, 3, 3)
```

## 3 Developing

There are 3 packages that constitute the tbpm software, they are:

- gbpm – generic bpm classes
- tbpm – bpm implementation specific to the tevatron
- echotek\_tbpm – underlying echotek driver

### 3.1 Build Environment

The Tevatron BPM software uses the RFI build methodology. Please refer to [Beams Document #1271](#) for a description of the tools and, for first time users, how to configure your account on nova.fnal.gov. Following these instructions, please set your work group to rfiinst. All code is compiled on nova.fnal.gov.

### 3.2 Build Instructions

All packages are built in a similar fashion

```
nova-> cd ~/esd/src           # only do this once
nova-> cvs checkout gbpm      # only do this once
nova-> setup gbpm
nova-> cvs update -d          # cvs checkout gbpm if first time
nova-> make clean
nova-> make
nova-> make development       # put in development location
nova-> make test              # put in test location
nova-> make production        # put in production location (will
                             # be installed at next reboot)
```

Library locations are:

```
nova.fnal.gov:/fecode-bd/vxworks_boot/fe/gpm/devgpm.out # development
nova.fnal.gov:/fecode-bd/vxworks_boot/fe/gpm/testgpm.out # test
nova.fnal.gov:/fecode-bd/vxworks_boot/fe/gpm/libgpm.out # production
```

There is a similar set of commands to build tbpm and echotek\_tbpm.

The Targets

### 3.3 Unit Tests

The three tbpm packages have a unit or module test associated with them. The unit tests are designed to verify the core functionality of the module and are designed to be as standalone as possible. They use an underlying software package called culite, which is a Computing Division maintained product that is built remotely and then distributed to nova. It is installed in

```
nova.fnal.gov:/fecode-bd/vxworks_boot/fe/culite
```

Because the unit tests in and of themselves are large, they should not be loaded when running in a production mode and are meant primarily for release verification. Running the unit tests involves the following steps:

1. build with unit tests turned on. Put in the “test” library:
2. Load the unit test software:
  - a. Create a startup script for unit testing, which includes loading unit test helper library along with the test libraries just built

```
ld < /fe/tbpm/libculite.out
```

3. Run the Unit tests. Each package has a self-contained unit test which calls all individual module tests. To run all the unit tests, from the vxworks shell:

```
vxworks-> echotek_tbpmUnitTest
vxworks-> gbpmUnitTest
vxworks-> tbpmUnitTest
```

### 3.3.1 Echotek Driver Unit Tests

### 3.3.2 GBPM Unit Tests

No test requires any more hardware than the crate controller except where otherwise noted.

```
void DataEntryUnitTest (int);
void TestDataEntryUnitTest ();
void DataBufferUnitTest (int);
void PackStrategyUnitTest ();
void TestPackerUnitTest ();
void EventUnitTest ();
void EventListenerUnitTest ();
void EventGeneratorUnitTest ();
void InterruptEventGeneratorUnitTest ();
void HardwareInterruptEventGeneratorUnitTest ();
void TimeEventGeneratorUnitTest (int);
void AuxTimeEventGeneratorUnitTest ();
void TCLKEventGeneratorUnitTest ();
void AlarmGeneratorUnitTest ();
void DataAcquisitionTaskUnitTest ();
void DataBufferUnitTest (int);
void CircularDataBufferUnitTest (int);
void TestControlUnitTest (int);
void TestBufferReadoutUnitTest (int);
void TestControlTaskUnitTest (int);
void AlarmTaskUnitTest (int);
void ConfigManagerUnitTest (int);
void PackerUnitTest ();
```

### 3.3.3 TbpM Unit Tests

No test requires any more hardware than the crate controller except where otherwise noted.

```
void TBPMClosedOrbitPackerUnitTest (int);
void TBPMDiagnosticsPackerUnitTest (int);
void TBPMControlUnitTest (int);
void TBPMDiagnosticSystemUnitTest (int);
```

## 4 Maintaining

How/when to change .ini and .ch files. Tune timing delays.

## 5 Diagnostics

Once the system is up and running, how do you start analyzing problems? There are many diagnostic tools available. Choosing the right one to most expediently analyze the problem is a combination of experience and instinct. The following sections describe various diagnostic information that is available in the tbpm software.

### 5.1 Software Statistics

The bpm software internally keeps structures that one can peek at through the vxworks shell. The bpmHelp command will list the most current set of commands that are available:

```
fccts1:tbpm-dev> bpmHelp
bpmStart (source, rate) where source=0: EchoTek, source=1: Test
bpmPeek (bpm, buffer)
  bpm - 0 through 11
  buffer - fast abort = 11
          - slow abort = 12
          - profile    = 13
          - display    = 14
          - snapshot   = 15
          - tbt        = 16
          - inj. tbt   = 17
          - inj. c.o.  = 18
bpmShow - Displays system information
bpmShowQueues - Display status of event queues
bpmShowBuffers - Display status/info of system buffers
bpmTCLK - Generate TCLK (0x47, 0x4D, 0x71, 0x75, 0x77, 0x78)value = 0 = 0x0
```

#### 5.1.1 BpmStart (source,rate)

#### 5.1.2 bpmPeek

#### 5.1.3 bpmShow

#### 5.1.4 bpmShowQueues

#### 5.1.5 bpmShowBuffers

#### 5.1.6 bpmTCLK

### 5.2 Trace Diagnostics

The Tevatron BPM software packages are using a trace facility to monitor software behavior. This document is not meant to be a tutorial for that facility - Refer to asdof for

detailed documentation. As a quick overview, we typically use the tracing facility in a circular buffer mode where all traced processes write into that buffer.

The software is exploiting many levels of trace that can be turned on/off to fine tune which information

### **5.3 VME Backplane Diagnostics**

Bus analyzers are a powerful tool in understanding what is actually happening on the backplane. Using a VMETRO `xdkjldsf`, the following traces were obtained to show vme timing of typical transfers:

Figure x. Closed Orbit Measurement

Figure x. TurnByTurn Measurement

Figure x. First Turn Measurement

### **5.4 Looking at ACNET variables**

## **6 Test Stand**

### **6.1 Test Crate Modules**

## **7 Appendix: Node name/ACNET addresses**

<b>node name</b>	<b>ip address</b>	<b>ACNET address</b>	<b>Location</b>
Fccts0			FCC3
Fccts1	131.225.126.234	0x0bd4	FCC3
TBPMA3	131.225.127.233	0x0c2b	A3

configuration files

acnet devices

labview board testing code

