FNPL BPM System RAM Module Readout Software

Jason Wennerberg
Fermi National Accelerator Laboratory

April 6, 2004

Abstract

This note describes the software created to facilitate the readout of the FNPL beam position monitor system RAM module on a Linux platform.

1 Introduction

The beamline at the FNPL at Fermilab is equipped with button beam position monitor (BPM) electromagnetic *pickup stations*. BPM *read-out electronics*, developed and manufactured at DESY, are used to acquire the beam position and orbit data. Most of the BPM electronics hardware is concentrated in two 19″ SEDAC (*SE*rial *D*ata *AC*quision) crates. They keep all analogue and digital electronics, as well as a serial link to the server PC.

The RAM module can determine the position of individual bunches in a bunch train.

2 New software Operation

The new software is called ramtest and was written in March 2004 in the C language. It is run via a control panel with the following switches available:

- -d: run and write debugging messages to screen
- -f: file name for output
- -q: quiet mode: runs with no output to screen
- -i: number of reads

When the program is run it will read the number of bunches and record their x and y position into the file designated by -f. Because the data file can contain more than one read, the end of the bunch is marked with a line containing the string "endofbunch". This should be taken into consideration when the data file is parsed.

3 New Software Overview

The procedure for reading the RAM module is listed below. BSA is the base address of the RAM module. At the time of this writing the RAM module is connected to BPM2 and its base address is 160. The RAM module is most useful if connected to the last BPM after the spectrometer.

1. The program is run via the command line with options

2. The RAM module initialized by reading the information about their addresses, constants, etc. from a file
3. The Sed crate is initialized using the function 'SedIni' from SedPc
4. Test for communication with the crate
5. Set mode: write 0x440 to BSA+7
6. Set gap detector: write 2 to BSA+10
7. Set RAM address: write 0 to BSA+9
8. Read from BSA+6: the purpose of this is unknown
9. Start data storage: write 0 to BSA+8
10. Wait for data to finish. This is when the value read from BSA+7 operated on by & 0x4000 = 0
11. Read number of bunches: read from BSA+6, divide the value by 8
12. Set address to 0: write 0 to BSA+6
13. Read the data: from BSA+8, read 2*number of bunches.
14. Each read contains data for 2 pickups so 2 reads are needed for each bunch. The first reads contains pickup 1 in the low byte and pickup 2 in the high byte. The second read contains pickup 3 in the low byte and pickup 4 in the high byte.
15. Check to see if sum of E values (see reference [2]) is between min and max values (400 - 800)
16. If sum of E is not acceptable or saturated the user is notified and should correct the error by running the BPM monitoring program)
17. If sum of E is acceptable keep data and stop or read again

4 Graphical display of data in real time

A program was also created that will read and display the data from the RAM module in real time. This program is called burst_display.py and is a python2.3 script. It can be run from the command line by simply typing /path/burst_display.py. The current path is /home/jasonw/BPM/RAM.

5 Remote Operation

Remote monitoring of the BPM system is possible by conecting to a0pi-spitefire6.fnal.gov via telnet with a CryptoCard. If you don't have an account on a0pi-spitefire6.fnal.gov that matches a kerberos principal, something else will have to be worked out.

5.1 Data Post-processing

Data post-processing is required to compute the normalized horizontal and vertical beam displacement Δ_h and Δ_v from the four acquired signal values of the electrodes E_i:

$$\begin{aligned} \Delta_h &= k_h \frac{\tilde{E}_4 - \tilde{E}_3}{\tilde{E}_4 + \tilde{E}_3} \\ \Delta_v &= k_v \frac{\tilde{E}_1 - \tilde{E}_2}{\tilde{E}_1 + \tilde{E}_2} \end{aligned} \tag{1}$$

The calibration factor (so-called *monitor constant*) (k_h, k_v) varies for different geometries of the BPM pickup. $\tilde{E}_i$ in eqn. (1) are the modified mean-values of the acquired electrode signals E_k: The mean-value is computed according to the ADD-values n, set at the BSA+9 registers of the ADD-modules.

$$\bar{E}_i = \frac{E_i}{n}$$

This mean-value $\bar{E}_i$ has to be modified for two aspects, the difference of the cable attenuation in the delay-line and the nonlinearities of the electronics at lower levels:

$$\tilde{E}_i = k_{\mathrm{att}_i} \bar{E}_i + 10$$

First order compensation of the nonlinearities is done by simply adding a "10" to each of the four digital values.

All the constants $k_h, k_v, k_{\mathrm{att}_i}, \ldots$ are kept in a file that is read into the program during the initialization process.

References

[1] K. Desler, J. Neugebauer, R. Neumann, F. Wedtstein, and M. Wendt, "Programming the A0 BPM System," Not published.

[2] K. Desler, R Neumann, and M. Wendt, "Introduction to the A0 BPM System," Not published.