Architecture of the APS Real-Time Orbit Feedback System[*]

J.A. Carwardine and F.R. Lenkszus
Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439, USA

Abstract

The APS Real-Time Orbit Feedback System is designed to stabilize the orbit of the stored positron beam against low-frequency sources such as mechanical vibration and power supply ripple. A distributed array of digital signal processors is used to measure the orbit and compute corrections at a 1kHz rate. The system also provides extensive beam diagnostic tools. This paper describes the architectural aspects of the system and describes how the orbit correction algorithms are implemented.

1 Introduction

The APS is the foremost third-generation synchrotron light source in the United States, delivering intense x-rays to as many as 35 insertion-device and 35 bending-magnet beamlines. As with other light sources, orbit stability is critical in order to achieve the optimum performance for the x-ray users. At APS the rms orbit motion must not exceed 5% of the beam size, translating to limits of 17μm rms horizontally and 4.5μm rms vertically.

The APS real-time orbit feedback system provides real-time (dynamic) control of low-frequency orbit disturbances. The design provides for both global (long spatial wavelength) orbit correction and for local correction at the x-ray source points, although so far, only the global system has been fully implemented.

The global feedback system has been in routine operation with users since June 1997, producing up to 20dB attenuation of orbit motion in a band up to 30Hz. Typical orbit motion with feedback is 8μm rms horizontally and 3.5μm rms vertically, both measured up to 30Hz. Ultimately, a bandwidth of 100Hz is desired.

2 Overview

The concept of global orbit correction has been well established in the literature [1,2]. Once a set of beam position monitors (BPMs) and correctors has been chosen, a linear ′response matrix′ can be determined that describes the change in the beam orbit when corrector magnet currents are changed. Given appropriate system dimensions, the relationship can be inverted mathematically, producing a matrix mapping the measured orbit to a set of corrector currents that will minimize the orbit. This matrix is defined as the ′inverse response matrix.′ Since the mapping is linear and time-invariant, the same matrix can be used for static orbit correction that is applied infrequently and for real-time orbit correction that cancels dynamic orbit errors.

The APS orbit feedback system makes orbit corrections at a 1kHz sample rate, using in each plane 160 BPMs to measure the orbit and 38 dipole magnets to correct the orbit errors. In all, 317 dipole magnets (correctors) are available in each plane, but there are only 38 high-bandwidth correctors suitable for global orbit feedback. The remaining correctors will be used for local correction at the x-ray source points.

3 Hardware layout

The APS system is entirely digital, using digital signal processors (DSPs) to perform the orbit correction calculations and to implement closed-loop feedback on each corrector magnet.

Since the BPM and corrector hardware are distributed around the 1.1km storage ring, the orbit feedback hardware is also distributed, with a total of 20 ′slave′ stations accessing BPMs and correctors and performing orbit feedback calculations. A separate ′master′ station controls the slaves and performs real-time data analysis.

Figure 1 shows an overview of the slave stations. Each station receives digitized data from 16 BPMs and sends digitized setpoints to as many as 32 correctors. There are also x-ray BPMs available on each beamline that measure the trajectory of the x-ray beam as it travels from the storage ring to the users experiment. Each station contains one or more DSP processors for performing the real-time calculations. A dedicated fiber-optic ″reflective memory″ network provides deterministic data transfer between stations. All the stations are connected to the APS controls network.

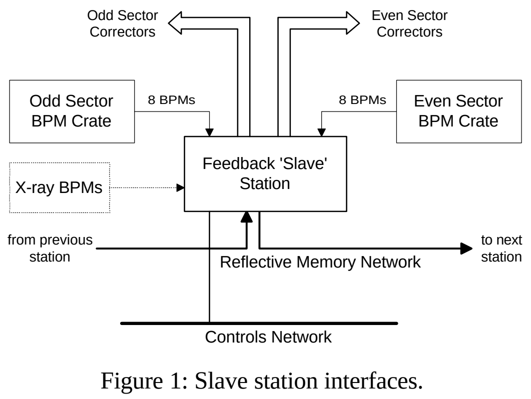

Figure 1: Slave station interfaces.

The global algorithm is divided by output channel (i.e. by corrector), as illustrated in Figure 2. Each ′corrector error′ is produced from the vector dot product of BPM errors and

[*] Work supported by U.S. Department of Energy, Office of Basic Energy Sciences under Contract No.W-31-109-ENG-38

one row of the inverse response matrix. By separating the inverse response matrix into rows, the algorithm is split into 40 identical calculations, one for each corrector. (The orbit feedback system has 40 corrector channels of which two do not actually connect to corrector magnets.) Each calculation requires the same vector of BPM values as every other, but uses a different row of the inverse response matrix. Each slave station handles the calculations for two correctors in each plane (four in total). The resulting ′corrector errors′ are subsequently used to generate the updated corrector setpoints. The channels can be treated independently so long as they remain synchronized in time.

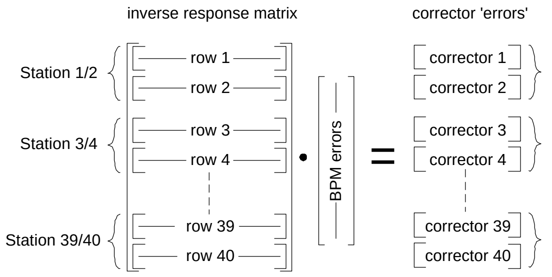

Figure 2: Separating the global algorithm.

4 Slave stations

A more detailed view of the slave stations is shown in Figure 3. Each station is implemented in a VME crate containing a Motorola MVME167 processor that runs EPICS core routines under VxWorks [3]. Real-time processing is performed with Texas Instruments TMS320 C30 and C40 floating-point DSP processors that reside on Pentek VME boards. Interfaces to BPMs and correctors are provided by custom electronics (FSIC, CMPSI, MSI, XRI). Synchronized 1kHz clock ticks are delivered to all the stations from the APS timing system.

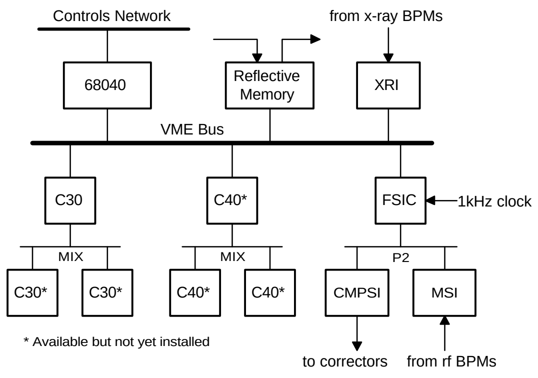

Figure 3: Slave station architecture.

The reason for the mix of C30 and C40 processors is mainly historical. The C30 processors were purchased some time ago, before the C40 versions were available. The more recently acquired C40s were chosen because of their improved performance over the original C30s.

The figure shows that six DSP processors will be installed in each slave; these will be used to implement both global and local orbit correction at a 2kHz sampling rate. The two main processors will have direct access to the VME bus, with each hosting two auxiliary processors over a MIX bus. Currently only the main C30 DSP is installed, and this is just sufficient to implement global orbit correction in both horizontal and vertical planes at a 1kHz rate. It takes the processor about 900μs to complete each cycle, of which 200μs is consumed in computing the two vector dot products.

The reflective memory network (supplied by VMIC) provides an essential link between the slave stations since all the BPM values must be transferred to all stations in order to implement the global orbit correction algorithm. The network is implemented with a ring topology over fiber optic and is accessed like any other block of memory in VME-space. It provides deterministic data transfer rates of 29.6Mbytes/second. Despite some reliability problems with earlier versions of the product, this system has proven to be a very effective method of transferring data at high speeds in a deterministic way.

In addition to transferring BPM errors from slave to slave, the reflective memory is used to transfer real-time quantities from each slave to the master station and to provide an interchange of control and status information between the master and slave stations. A flowchart of the slave software is shown in Figure 4.

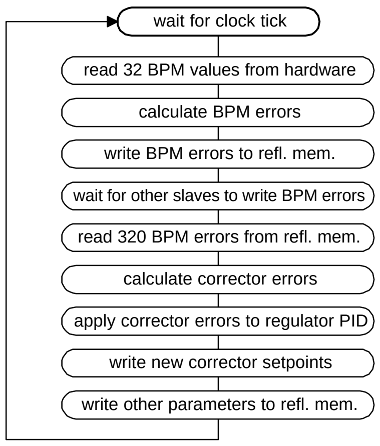
wait for clock tick
read 32 BPM values from hardware
calculate BPM errors
write BPM errors to refl. mem.
wait for other slaves to write BPM errors
read 320 BPM errors from refl. mem.
calculate corrector errors
apply corrector errors to regulator PID
write new corrector setpoints
write other parameters to refl. mem.

Figure 4: Slave station flowchart.

When a clock tick is received, each slave reads the latest values from its local BPMs and writes the computed errors to reflective memory. Once all slaves have completed their writes, they all read back the entire vector of 160 BPM errors (320 for the two planes). Corrector errors are then computed from the dot product of the BPM error vector and the appropriate row of the inverse response matrix. The corrector errors are then applied to a digital PID regulator before being sent to the corrector power supplies.

The process of reading 320 BPM values from reflective memory has proven to be a significant bottleneck, taking almost 200μs of the available 1ms. The problem is worsened by the fact that VME block transfers are not supported on the DSP boards used in the slave stations.

Adding a second main DSP processor will allow us to spread the burden of reading the BPM values from reflective memory. The plan is to separate the task such that one main DSP will handle only horizontal orbit correction and the other will handle only vertical orbit correction. The two DSPs will be staggered in time, with the first DSP reading its BPM values while the other waits. Once the first DSP has finished accessing the reflective memory, the second will start reading its BPM values. Thereafter the two DSPs will proceed independently. This arrangement does mean that there will be a delay between writing new horizontal and vertical corrector setpoints, but since the two correction planes are essentially decoupled, there will be minimal impact from doing this. By splitting the two correction planes, it will be possible to double the sampling rate to 2kHz despite the bottleneck of reading the BPM values from reflective memory.

The four auxiliary DSPs will be used to implement local correction at the x-ray source points. At this stage it is unclear whether the local feedback algorithm will be implemented using ′local bumps′ or using a second inverse response matrix [4]. Depending on the chosen algorithm, each slave may have to compute as many as eight local bumps or 28 additional vector dot products.

5 Master station

The original system design did not include provision for a ′master′ station. This was added in order to simplify the task of synchronizing the operation of the 20 slave stations. We have subsequently extended the functionality of the master in order to include real-time data analysis.

The master station, shown in Figure 5, differs from the slaves in that it has no access to BPMs or correctors and has a different arrangement of DSP processors.

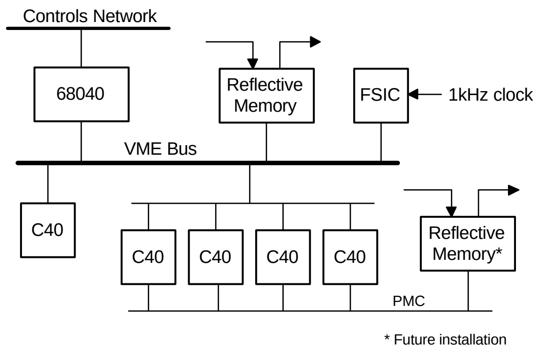

Figure 5: Master station architecture

The single C40 DSP implements supervisory control and some real-time analysis. The array of four C40s was added to increase real-time processing capabilities. The station relies entirely on the reflective memory network for data transfer between stations.

The figure shows a second reflective memory board connected via a PMC bus to the array of four DSPs. This will be added in order to speed up read access to the reflective memory and will considerably improve the effective throughput of the four C40 processors.

Supervisory tasks include delivering system parameters such as regulator PID settings and detecting problems with the slaves via various status flags. For example, if any control parameters exceed operating limits in one of the slaves, the master turns off all the other slaves in order to prevent dumping stored beam. Each slave also increments a ′heartbeat′ register on each clock tick. Missed heartbeats are detected by the master station and reported to the APS control system.

The system can operate in several modes including the normal ′closed-loop′ mode. For example, we can drive any of the correctors with a synthesized function (e.g., a sinusoid) in order to carry out beam-related measurements. We can also run the orbit correction in ′single-step′ mode for algorithm checkout.

6 Real-Time beam diagnostics

Significant effort has gone into providing real-time beam diagnostics through the orbit feedback system, taking advantage of the fact that 320 BPM values are collected and processed at a 1kHz rate. Initially the diagnostics were aimed at quantifying orbit motion power spectra and orbit feedback system performance. More recently we have found the diagnostics equally useful for analyzing transient orbit motion that dumps stored beam, as a tool for locating the sources of rms orbit motion.

Dspscope

This is the most elementary diagnostic, providing 40 channels of user-selectable data collection. The system can collect 4000 data points from a variety of sources, including any of 640 BPM values and 76 corrector errors. There are also extensive triggering capabilities. Data is accessed as EPICS waveform records.

AC Voltmeter

This diagnostic follows the time evolution of a user-selectable frequency in various signals. There are actually two *AC Voltmeters*, the first operating on the 40 selectable channels of *dspscope* and the second operating simultaneously on all 320 BPM values in either plane. The 320-channel *AC Voltmeter* is implemented using four C40 DSPs. It takes about 800μs to read the 320 BPM values and compute one Fourier component in each.

When used with the ′drive corrector′ mode, the *AC Voltmeter* provides a fast method for measuring response matrices, allowing measurement of a 320-BPM by 38-corrector response matrix in about ten minutes.

Corrector Error Statistics

At each time step, a running *mean* and *variance* of each corrector error is computed. These statistics are useful for detecting malfunctioning of the orbit feedback system itself and for detecting problems with BPMs.

Corrector Error History Buffers

In addition to computing corrector error statistics, we also store the last 128mS of each corrector error in a circular buffer that is frozen when stored beam is dumped. Since the corrector errors provide a measure of the location of any source of motion, we can often use these history buffers to localize any source of unwanted beam motion that results in an unintentional beam dump.

7 Real-Time algorithms

A major consideration in selecting algorithms for the real-time analysis is that the system is configured to run the same identical code on each clock tick. Consequently, algorithms that operate on blocks of time-sequence data are more difficult to implement. Our algorithms for computing error statistics and frequency components are all ′sliding′ algorithms, updating previous results given the latest data sample. The algorithms are straightforward to derive and are described here for reference.

The mean value is simply the output of a lowpass IIR filter. The variance is computed from the square of the difference between the instantaneous value and the mean, with a lowpass filter generating the expected value.

The sliding Fourier transform takes advantage of the fact that the previous value of the Fourier component can be updated given only the latest data sample and the sample from *N* time steps earlier (for an *N*-point discrete Fourier transform). The equation is as follows:

$$Y_k(n) = \left[Y_k(n-1) + x(n) - x(n-N) \right] . e^{-j2\pi \frac{k}{N}}$$

where $Y_k(n)$ is the k^{th} (complex) Fourier component at time step *n*, $Y_k(n-1)$ is the previous Fourier component, *x(n)* is the latest data input, *x(n-N)* is the data input at time step *(n-N)*, and *N* is the number of points in the discrete Fourier transform. It takes *N* time steps for the calculation to be completed, but thereafter, the Fourier component is updated at every time step. The algorithm is very efficient on a sample-by-sample basis, even though it requires considerably more computations in total than the corresponding FFT algorithm.

8 Code development

The DSP software is developed on a Unix workstation using the Texas Instruments TMS3203x/C4x Code Generation Tools. While most of the code is written in C , procedures performing operations on arrays are written in C-callable assembly language routines in order to improve performance. The C code is compiled with optimization. This can cause unexpected behavior, particularly when the compiler rearranges code to minimize DSP pipeline conflicts. When this occurs, inspection of the resulting compiler assembler output is necessary to diagnose and correct the problem.

Code is downloaded to the DSP over the controls LAN from the control system file server using tools in SwiftNet, a product available from the DSP vendor [5]. These tools run on the VME controls processor under the VxWorks [3] real-time kernel. Our system is configured to download the DSP software each time a feedback VME crate is booted.

We have not made extensive use of debugging tools. Rather, we have used reserved ′test′ locations in dual-access RAM on the DSP to pass debugging information to the VME controls processor. This method has little impact on the DSP algorithm and allows us to debug at the normal feedback system operating speed.

To date we have not used a DSP real-time kernel, but have chosen to run a single task which linearly executes the feedback code. Thus we don′t incur any overhead due to context switching. We will revisit this issue as we add additional DSP processing power to the system.

9 Interface to the control system

The DSP operation is controlled and monitored through data structures residing in dual-access RAM on the DSP board. Elements of these data structures are interfaced to EPICS process variables through EPICS sequence programs. Values such as BPM readings and computed corrector values are deposited in a data structure by the DSP. A sequence program on the controls processor periodically scans the data structure and deposits values in corresponding process variables. The process variables are control system entities and therefore are accessible by all the standard control system tools.

A separate EPICS sequence program transfers local control information such as inverse response matrix rows, BPM selection, etc., to the DSP resident data structure and sets a flag commanding the DSP to load the new values into fast local static RAM. Global controls such as feedback loop open/close, filter cutoff frequencies, etc., are passed to the slave stations via the reflective memory. The master station writes global control values to reflective memory and sets a flag that commands slave stations to read in the new control information.

References

[1] Synchrotron Radiation Sources - A Primer, H. Winick, pp 344-364, World Scientific, 1994.

[2] Krinsky et. al., AIP Conf. Proc. 249 (1992).

[3] Wind River Systems, Alameda, CA, 94501.

[4] Carwardine et. al., Proc. IEEE PAC, (1997).

[5] SwiftNet Node Software for VxWorks 5.x, Pentek Upper Saddle River, N.J. 07458.