

Computer Controls Programming for Decelerating Beam in the Fermilab Antiproton Accumulator

V. Bharadwaj, D. Bogert, M. Church, M. Glass, D. McConnell, W. Marsh, D. Peterson
*Fermi National Accelerator Laboratory**

Abstract

This paper describes the applications programming written to facilitate deceleration of beam and associated studies in the Accumulator. The paper will include a description of the front-end programming, console programming for beam and tune control, console programs for calculating, manipulating and loading deceleration ramps, and an automatic tune measuring program. In addition there is a program in a VME processor to control the RF oscillator. Operational experience with these programs will be described.

I Introduction

The availability of intense antiproton beams in the Fermilab Pbar Accumulator opens up the possibility for doing low energy antiproton physics. The first experiment approved for the Accumulator Ring at Fermilab is E760^[1], a charmonium production experiment which requires antiproton beam with momenta in the range of 6.3 GeV/c to 3.9 GeV/c. The Fermilab Accumulator was designed and built as a fixed energy ring of momentum 8.9 GeV/c.^[2] This paper describes the technical solutions to some control problems arising from the new energy requirement and the additional software written to control the deceleration process and facilitate deceleration test studies using protons. The test studies are described in a companion paper.^[3]

A schematic overview of the Fermilab Accelerator Control System^[4] is shown in Figure 1. The top row shows 20 PDP-11/34 computers which support accelerator operator consoles on a one to one basis. Most human intervention is accomplished through application programs running on these computers. The next level down represents the central VAX's. These computers serve as a central database repository for routine and scaling information, as a central message switching facility between the network busses, as an additional computation facility for larger calculations and simulations, and as a development environment for ongoing work. In the next level are the front end PDP-11's which serve as an interface between the operator consoles and the CAMAC modules and microprocessor based systems which control the accelerator.

II Hardware Choices

Normally in the Fermilab Control System, magnet ramps are generated by dedicated microprocessors.^[5] Ramp tables are downloaded by application programs running on the console computers. Then by using timing and scaling signals put into these microprocessors, ramps are generated at a 1 kHz update rate.

However, since the Pbar source was designed as a fixed energy machine, it was instrumented with D/A's and digital outputs directly on the front end's serial CAMAC link rather than with the more expensive microprocessor based ramp modules. To accomplish deceleration for E760 one could have reinstrumented the Accumulator with ramp modules. This would have been rather expensive and because of the reconfiguration involved would have been disruptive to the ongoing Collider program. Another method was needed.

Fortunately, several factors conspired to make a more elegant solution possible. Because of the large inductance of the Accumulator power supplies and magnets, deceleration could only proceed slowly over a period of several minutes. Most devices would need to be updated only at 1 Hz while a few would need a 60 Hz rate. Also, the "home grown" serial CAMAC crate controller^[6] used in the Fermilab control system has a possibility of a second link being attached with the appropriate hardware arbitration. Two computers can therefore hook up to the same CAMAC crates. Thus, the solution adopted was to create a second auxiliary front end PDP-11 (PAUX) whose sole purpose is to ramp all the devices of the Accumulator during deceleration. The Antiproton front end is left unchanged while the new auxiliary front end is connected to the second port of the relevant CAMAC crates.

PAUX is seen by the consoles as just another front end. An application program running on one of the consoles downloads ramp tables to PAUX. On a signal from this application program PAUX generates the ramps necessary for deceleration by setting the D/A's and digital outputs.

The piecewise linear ramp tables are functions of a pseudo-variable P (ie. $f(P)$ where P normally is the momentum). There is an additional table which maps changes of P onto time in 60 Hz ticks. At each 60 Hz tick PAUX generates a new P value, updates all the 60 Hz devices, and then finally updates the slower 1 Hz devices if necessary. Beginning and ending P values are set by the console application program before the ramp. In addition, the ramp can be stopped and continued by the console program at any time.

III Front End Software

In developing the software for PAUX, a stripped down version of the normal PBAR front end software was used as a starting point. Included in this stripped down version were tasks which support communication to the consoles, tasks which interpret read and set requests from the consoles, tasks which support debugging of software and the serial CAMAC link, a periodic scheduling task, the shell of a task which does individual device reads and sets, and the RSX-11M operating system. The periodic scheduling task was changed to support 60 Hz scheduling for ramp device updates. The shell of a task which does individual device reads and sets was expanded to manage the ramp tables so that they could be read and set from a console. In addition, two tasks were written which actually do the ramps—one task for the 60 Hz devices and one task for the 1 Hz devices. The project took about 2 to 3 man-months to complete.

The ramp tables appear to the console application program as a standard ramp device. An entry for each such device is made in the system database residing on one of the VAX's. Information about the device (ie. CAMAC crate, slot, and subaddress, type of CAMAC module, and update frequency) is recorded in this entry. When the application program reads or writes a ramp device, this information along with the request and data is sent to PAUX for appropriate action.

The ramp tables in PAUX are maintained in a dynamic shared memory in the form of three linked lists—one list for the pseudo device, one list for the 60 Hz devices, and one list for the 1 Hz devices. At boot time these lists are empty. When the console application program reads or writes a ramp table device, a search is made of the relevant list for the appropriate table using the information from the system

*Operated by the Universities Research Association under contract with the U. S. Department of Energy

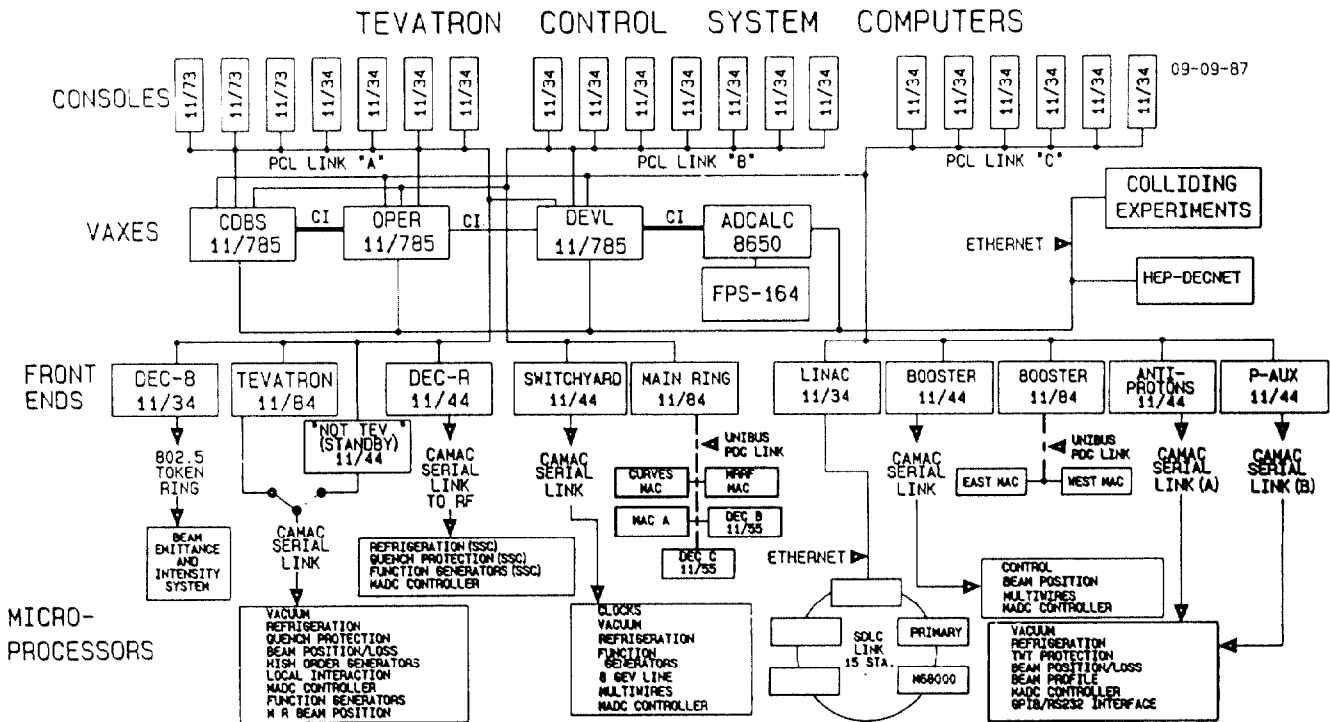


Figure 1: The Fermilab control system computer network

database. If a device is not found an entry is created at the end of the list. The appropriate action is then taken. Each device entry contains not only the ramp table, but also the database information telling the type of CAMAC module and its CAMAC crate, slot, and subaddress and a status word which can be used to enable or disable the device from ramping. When the application program tells PAUX to start the deceleration, the two ramp tasks use these lists to generate the ramps.

The scheduling task activates the fast ramping task at 60 Hz. The fast ramp task first computes a new P value, and then goes through the list of 60 Hz devices computing and then setting the new values. The scheduling task also activates the slow ramping task at 15 Hz. When it is time to update the 1 Hz devices, the slow ramp task goes through the list of 1 Hz devices computing and then setting new values. The updating of the 60 Hz devices is done at AST level. Thus, because the priority of the fast ramping task is set higher than the slow ramping task, several updates of the fast devices may and usually do take place during one 1 Hz update list. However, experience has shown that there is plenty of computer time to update the ten 60 Hz devices and the eighty 1 Hz devices in the given time constraints necessary for deceleration.

IV Console Software

Two primary application programs give the user access to the deceleration ramps. One of these programs (P76) is used only for generating, fitting, smoothing, and plotting the ramps. The other program (P78), in addition to manipulating the ramp tables, will load the ramp tables in PAUX, disable or enable specific devices, start and stop the deceleration process, and keep track of the current location (momentum) in the deceleration. The ramps are displayed as tables of device settings (current, voltage, time, frequency) vs. momentum.

P76 is an "offline" program in the sense that it does not communicate with PAUX or any device in the Accumulator. It reads a selected

set of ramp tables from a disk file on the Central Database VAX and displays them at the console graphically and in table form. The primary purpose of this program is to smoothly extrapolate the ramps to a new momentum range, thus generating new ramp tables. This can be done with a polynomial (linear, quadratic, or cubic) or simply by hand. Similarly, different table points (device setting vs. momentum) can be generated by smoothly interpolating between existing table points with a linear, quadratic, or cubic polynomial, or by hand.

P78 is an "online" program in the sense that it gives the user direct control of the deceleration process. Ramp tables are read from the Central Database VAX and loaded into PAUX by this program. The deceleration may be started or stopped by an interrupt from the operator, or the deceleration can be stopped at a preselected momentum. All communications with PAUX appear to the user as standard reads or sets to devices. Devices may be enabled or disabled; ramp table points may be changed by hand and the ramps reloaded into PAUX; ramps may be saved in the Central Database VAX; and two device specific calculations may be performed. One calculation generates an RF voltage ramp from input of initial and final RF bucket areas. A second calculation generates an RF frequency ramp from input of RF harmonic number (84, 85, and 86 were used) and Accumulator circumference.

P78 also provides the mechanism by which the ramps are "trained". As an example of this procedure, consider the dipole bus. The dipole magnets are heavily saturated at 8.9 GeV/c and there exist no suitably accurate measurements of B vs. I for these magnets. Therefore the following empirical method was used to generate the correct ramp. Starting with a dipole ramp linear in momentum we decelerate 40 MeV/c. The beam position is measured and then the dipole bus current is corrected to bring the beam back to the center of the aperture. A "retraining" procedure in P78 then generates a corrected dipole ramp, based on the new dipole bus setting, and this new ramp is loaded into PAUX. This procedure is repeated until the desired momentum is

reached. Although time consuming, this technique was found to be effective. This general method was also applied to the quadrupole busses (keeping tunes constant), the trim magnets and dipole shunts (keeping the orbit constant), and the skew quads (minimizing the horizontal and vertical coupling).

In order to easily determine the quadrupole ramps during deceleration, a third program was written to automatically measure the tune of the machine. It interfaces to a HP8568B spectrum analyzer through a CAMAC to GPIB interface card that was built at Fermilab.^[7] Longitudinal and transverse Schottky signals are put into the spectrum analyzer.

The most basic feature of the program is to accept an expected tune range from the user, and read the RF hardware to determine the revolution frequency and then calculate the frequency range to set into the spectrum analyzer. Then a peak search is done to determine the Schottky sideband peak frequency, and the sideband power is measured. The tune is calculated and displayed along with the sideband power.

In addition, the program can setup, read, and plot the Schottky longitudinal, and upper and lower sidebands in both horizontal, and vertical planes. From this data momentum spread, tunes, emittances, and chromaticities are calculated and displayed. The full measurement process takes about 30 seconds.

A fourth Console program was written to facilitate the use of the Pbar Source Complex by non-experts. The program has two parts. The first part recycles the power on the appropriate Accumulator magnet busses, stores beam at the correct orbit and turns on the stochastic cooling. The second part can initiate tune and position measurements and correct the bend and quadrupole busses using the measured information. This program, although primitive, proved to be very useful in the deceleration studies. For the next set of running a complete deceleration sequencer will be written which will remember to do all the required accelerator operations automatically. This automatic approach will be essential when using the much more valuable antiprotons for real physics running.

V Frequency Control

Smooth deceleration of the Accumulator beam requires a high frequency source with low phase noise, wide tuning range and small tuning steps. A Sciteq direct digital synthesizer (DDS) with 29 bits of resolution between 32 and 64 MHz was selected.^{[8],[9]} The DDS is controlled by a Motorola 68000 microprocessor in a VME crate. The VME processor provides frequency ramp control and DDS test modes.

The front end PDP-11 (PAUX) sends 16 bit current control words to the bend bus supply and 32 bit frequency control words to the DDS. The bend supply has a large inductive load and responds with a time constant of approximately 200 milliseconds. The DDS can settle to a new frequency in a few hundred nanoseconds. The VME processor uses a filtering algorithm to allow the DDS to slew to the new frequency with a time constant nearly the same as that of the bend supply.

The frequency control software for the VME processor is written in Pascal, compiled and programmed into read-only-memory. The filter algorithm is in the form

$$F(n) = F(n-1) + [F(input) - F(n-1)]/K \quad (1)$$

where $F(n)$ is the frequency in the n th iteration; $F(input)$ is the requested final frequency; K is the filter time constant factor.

The filter time constant factor K is dependent upon the cycle time of the processor and the desired real time constant for the filter algorithm. The advantages of this algorithm are that only three 32 bit words needed to be stored to provide for smooth filtering and that the execution is very fast which allows for small frequency steps.

The variables are stored and manipulated as 32 bit integers and an addition to the algorithm is needed to handle the case where the calculated step is smaller than one least significant bit (LSB). If $F(input) - F(n-1)$ is not exactly zero but is smaller than K the integer division from eq. 1 will return zero and the frequency will settle to

a value different from that requested. The filter will continue smoothly to the proper value if the following addition is made

$$W = K/[F(input) - F(n-1)] \quad (2)$$

where W is the number of wait cycles before stepping one LSB in the proper direction.

The filter algorithm in its final form consists of three sections; the first as shown in eq. 1 for large steps, the second as shown in eq. 2 for delayed single LSB steps and the third is a simple loop when the input and output frequencies are exactly equal.

Though the original concept for computer control of the DDS was merely for the filter implementation it soon became apparent that the flexibility of computer control would allow other features to be easily added. One such mode was designed for phase displacement deceleration of the beam^[10] due to limitations on the maximum RF cavity voltage. The sweep width and sweep time can be set prior to the actual execution of the phase displacement mode. Once in process the frequency is swept each time the input frequency value changes.

Computer control of the DDS allows for many different modes of operation. The control program written in a high level language allows for easy review and quick modification to test new ideas. Having the program resident in PROM is a bit of an inconvenience for modifications but insures the integrity of the software during power failures and system resets. The computer and synthesizer have been operating continuously for over a year without failure.

References

- [1] Fermilab Experiment 760, Fermilab, U. Ferrara, U. Genova, U. California(Irvine), Northwestern U., Pennsylvania State U., U. Torino.
- [2] *Design Report Tevatron I Project*, Fermi National Accelerator Laboratory (October 1984).
- [3] V. Bharadwaj, M. Church, E. Harms, S. Y. Hseuh, W. Kells, J. Maclachlan, W. Marsh, J. McCarthy, R. Pasquinelli, N. Pastrone, J. Peoples, D. Peterson, X. Wang, "The Use of the Fermilab Antiproton Accumulator in Medium Energy Physics Experiments", this conference.
- [4] D. Bogert, Nucl. Instr. and Meth. A247(1986)8.
- [5] D. Bogert, Nucl. Instr. and Meth. A247(1986)8.
- [6] R. Ducar, IEEE Trans. Nucl. Sci. NS-28,3 (June 1981)2303.
- [7] K. Seino, L. Chapman, W. Marsh, Nucl. Instr. Meth. A247(1986)215.
- [8] Sciteq Electronics Inc., San Diego, California. Data sheets for VDS NMR-3000 synthesizer.
- [9] David A. Sunderland, et al., "CMOS/SOS Frequency Synthesizer LSI Circuit for Spread Spectrum Communications", IEEE Journal of Solid State Circuits SC-19,4 (August 1984)497.
- [10] Edmond Ciapala, et al., "The CERN ISR Control Scheme for Acceleration by Phase Displacement", IEEE Trans. Nucl. Sci. NS-26,3 (June 1979)3395.