Software Environment and Configuration for the DSP Controlled NSLS Booster Power Supplies*

R. Olsen, J. Dabrowski
National Synchrotron Light Source, Brookhaven National Laboratory
Upton, New York 11973 USA

J. Murray
SUNY Stony Brook
Stony Brook, New York 11794 USA

Summary

The booster at the NSLS is being upgraded from 0.75 to 2 pulses per second by means of the installation of new dipole, quadrupole, and sextupole power supplies. The control system of these power supplies employs general purpose digital signal processing modules, and therefore, software support is required. This paper outlines the development system configuration, and the software environment.

INTRODUCTION

At the beginning of this project, it was realized that the single largest task would be the writing of the system software. We therefore wished to make the maximum possible use of commercially available software packages. The design process was driven, to a considerable degree, by what software could be purchased. The hardware modules, for example, were designed to have a multiple I/O capability, being equipped with a VME interface in addition to the internal bus(1). In this way, early operation of various sub-systems could be achieved by using equipment on hand, without the necessity of waiting for other system components. A VME format DSP module was purchased to allow for software development, and initially, real time data transfer was done over the VMEbus rather than the VSBbus. Although it did not have enough power to execute more than one task at a time, this processor was, non-the-less an invaluable development tool.

SYSTEM HARDWARE CONFIGURATION

The control system components are mounted in a 21 slot 6U VME crate. The slot functions for the dipole system are allocated as follows(2).

1. interface to NSLS control
2. spare
3. spare
4. ramp generator
5. |
6. | host processor (PC-386)
7. DVM interface (current feed-back)
8. signal processor 1
9. A/D (60 Hz phase reference)
10. phase locked loop
11. signal processor 2
12. A/D's (supply voltage, filter current) (100 volt supply)
13. signal processor 3
14. A/D's (supply voltage, filter current) (1000 volt supply)
15. spare
16. spare
17. signal processor 4
18. trigger generator
19. trigger generator
20. gate driver controller
21. gate driver controller

The partitioning of system functions is shown in Fig. 1. The system host is a VME format PC-386 equipped with VGA and an 80MB SCSI hard drive. A PC was chosen as the system host, since all the development tools are available in a PC version, and they are relatively inexpensive. A 386 processor is more than adequate, since its function is only to download, initialize, and monitor for error interrupts. During normal operation, neither a display nor a keyboard are equipped. On power-up, a .BAT file initializes the downloading sequence of the signal processors and starts program execution.

SYSTEM SOFTWARE CONFIGURATION

The signal processing functions which are performed are, for the most part, filters. Although some of these filters may have many poles, and have very stringent performance requirements, they are all constructed by cascading simple biquads. An example of a single biquad is shown in (9).

For system functions, such as the phase locked loop, which operate at low sampling rates (2-60Hz interrups), the bulk of the code is written in C, and SPOX(7)is used as the

*Work performed under the auspices of the U.S. Department of Energy, under contract DE-AC02-76CH00016.

0-7803-1203-1/93$03.00 © 1993 IEEE

operating system. This allows us to implement filters simply using system level functions to create arrays for coefficients, data and delay nodes, without having to consider low level detail. Functions which require high sampling rates are generally less complex, and are written in assembler and serviced by a simple interrupt handler. On the DSAP module (3), SPOX is run in node 0 to perform data streaming for module to module data transfers in conjunction with the 68030 I/O controller. Processes which run at high sample rates would generally be relegated to the mezzanine board. In the case of the phase locked loop which has only two discrete functions, the loop program runs in node 0, while node 1 runs a ten pole constant phase filter with a sampling rate of 10 KHz which filters the output of the phase reference A/D converter and conditions the signal to the format required by the phase comparator. In this case, the mezzanine in not equipped.

PROTOTYPING OF FEED FORWARD SYSTEM

The booster dipole power supply employs a feed forward system to improve tracking accuracy(4). This system employs digital data storage and analog signal processing. In order to gain an immediate improvement in booster performance, it was decided that a prototype of the digital feed forward block which had been developed(5) for this project would be constructed and installed on the old power supply. To accomplish this, we made use of the fact that the SDIC (servo data I/O channel) had a VME interface in addition to the internal bus, and also that a D/A converter had been provided for test purposes. A transition module was constructed to take the place of the old feed forward module in the regulator NIM bin. This module supplies the start of cycle strobe, clock, and error signal, and receives the feed forward correction voltage which is summed into the analog voltage feed back loop.

An adjacent VME crate holds a VME AT, a purchased DSP module with a VME interface, and an SDIC. The control program was written in C, and runs at an interrupt rate of 720 Hz. Initialization is as described above. This system resulted in a factor of 20 decrease in acquisition time, and a factor of 5 increase in tracking accuracy.

Measurements(6) indicate a improvement in booster tune, and we now observer a considerable improvement in pulse to pulse stability as well as increased booster current.

DEBUG ENVIRONMENT

The system host PC is provided with compilers for the 80386, 68030, and TMS320C3X processors. In this way, an application program may be written in C and compiled to run on any of the three processors. Programs are written to be bus independent, and a bus specific driver is written for each processor type as required.

The first level of debugging was related to hardware in application modules such as the trigger generator and servo data I/O. This was accomplished by writing test programs in C and running them on the on the PC. Liberal use of color in the displays greatly facilitated the development effort. These programs read and write registers, and in addition to providing application specific functions, could be made to loop in order to facilitate trouble shooting with test instruments.

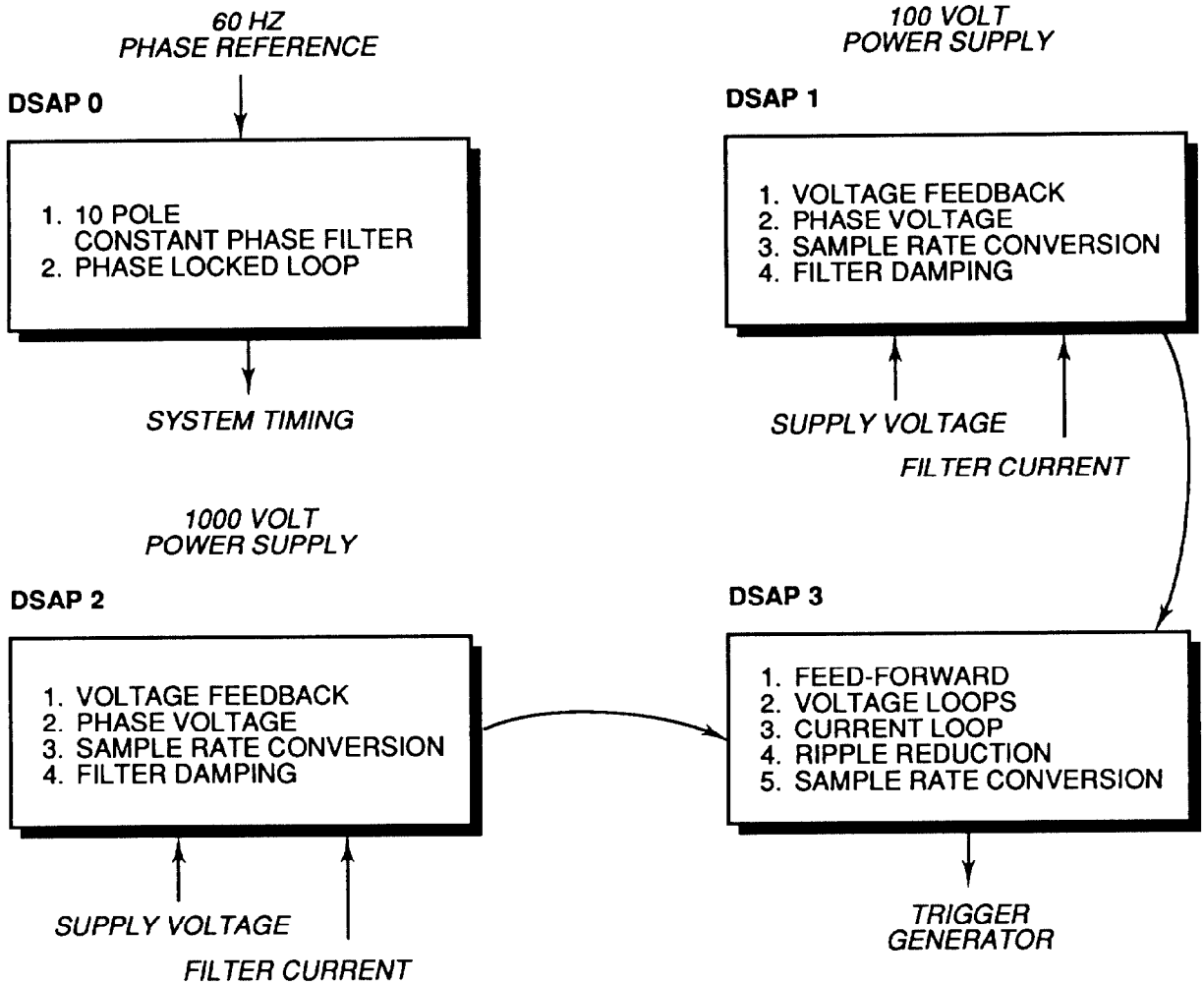
At the second level, an application was written and run on the purchased DSP module. This communicated with the application directly over the VMEbus, or through an adapter module which mapped VME A16 D16 to the A32 D32 of the P3 internal bus. In this way, a program could be run on a TMS320C30 with a VME interface, then by changing the driver, transferred to a DSAP module where communication takes place over an internal module to module bus or the VSBbus.

The third level of debugging is that of the DSAP module. This debug facility was designed around two commercially available products. The TMS320C3X system is designed around the Emulator Porting Kit(7) and is described in (3). The 68030 system consists of a monitor which is resident in SRAM, and a debug controller which runs on a PC-386. The two systems communicate over a serial port. The monitor is down loaded to SRAM by means of the TMS320C31 loader/DMA on the 68030 bus(3). In the case of the DSAP, the EPK runs on the system host, while the 68030 debugger runs on a stand alone PC. In this way, the 68030 I/O processor and a DSP node may, for instance, be single stepped simultaneously, to check message passing between the two processors. Both systems provide extensive window based facilities which allow manipulation of memory and processor registers.

OPERATING ENVIRONMENT

During normal operation, all process initialization is done via the test bus controller/emulator bus. In this instance, we invoke not the EPK, but a program whose only function is to load executable files to a given starting address. The system load resides on a hard disk, but we envisage converting to a WORM drive before the equipment is placed in permanent service. The hard drive will be retained for further development work and to trap errors as required, but would be powered down when not in use.

Since each DSAP module is equipped with a test bus controller IC, system problems may be resolved by using the on-board emulation rather than resorting to external test equipment.



SYSTEM SOFTWARE CONFIGURATION

Fig. 1

REFERENCES

[1] R. Olsen, J. Dabrowski, J. Murray "Control System for NSLS Booster Power Supply Upgrade" these proceedings.

[2] For an outline of each module function see R. Olsen, J. Dabrowski, J. Murray, "Dipole Power Supply for National Synchrotron Light Source Booster Upgrade" 1992 IEEE Nuclear Science Symposium Conference Record, p. 572.

[3] R. Olsen, J. Dabrowski, J. Murray, "Digital Signal Array Processor for NSLS Booster Power Supply Upgrade" these proceedings.

[4] B. B. Culwick, R. E. Olsen, "Application of a Digital Trigger Generator to NSLS Booster and Storage Ring Power Supplies; Proceedings, 1983 Particle Accelerator Conference, IEEE Transactions on Nuclear Science.

[5] J. Murray, R. Olsen, "A Differential - Decay Control for Ramped Magnet Current" Proc. 1992 IEEE Regional Controls Conference, Polytechnic University of Brooklyn, NY, pp. 85-88.

[6] E. Blum, R. Nawrocky, "Tune Measurement in the NSLS Booster Synchrotron" these proceedings.

[7] Spectron Microsystems.

[8] Texas Instruments

[9] TMS320C3X User Guide pp. 11-56