ADVANCED LIGHT SOURCE CONTROL SYSTEM*

S.Magyary, M.Chin, C.Cork, M.Fahmie, H.Lancaster, P.Molinari, A.Ritchie, A.Robb, C.Timossi

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

Abstract

The Advanced Light Source (ALS) [1] is a third generation 1-2 GeV synchrotron radiation source designed to provide ports for 60 beamlines. It uses a 50 MeV electron linac and 1.5 GeV, 1 Hz, booster synchrotron for injection into a 1-2 GeV storage ring. Interesting control problems are created because of the need for dynamic closed beam orbit control to eliminate interaction between the ring tuning requirements and to minimize orbit shifts due to ground vibrations. The extremely signal sensitive nature of the experiments requires special attention to the sources of electrical noise. These requirements have led to a control system design which emphasizes connectivity at the accelerator equipment end and a large I/O bandwidth for closed loop system response. Not overlooked are user friendlyness, operator response time, modeling, and expert system provisions. Portable consoles are used for local operation of machine equipment. Our solution is a massively parallel system with >120 Mbits/sec I/O bandwidth and >1500 Mips computing power. At the equipment level connections are made using over 600 powerful Intelligent Local Controllers(ILC-s) mounted in 3U size Eurocard slots using fiber-optic cables between rack locations. In the control room, personal computers control and display all machine variables at a 10 Hz rate including the scope signals which are collected though the control system. Commercially available software and industry standards are used extensively. Particular attention is paid to reliability, maintainability and upgradeability.

Design Goals

General Requirements

The following design goals were imposed for acceptable machine operation:

1. A 10Hz rate at the operator consoles for acceptable operator response times.
2. Time predictable access to devices and their parameters for smooth operation and guaranteed accelerator hardware protection. This implies a non-collision detect type network for most of the system.
3. Sufficient performance for auto-tuning and machine protection.
4. Remote "real-time" signals. We will install oscilloscopes and other instruments (digitizers, etc.) in the equipment racks, read them via a local controller and then send the data to the control room via a digital network. [2] This provides improved analog signal fidelity by removing ground loops, noise, etc. In addition a considerable amount of expensive analog cabling can be eliminated both around the machine and in the control room. This helps in providing an uncluttered look to the control room and the operator consoles. The ability to archive these digitized analog signals allows comparison and matching past with current machine operation. Furthermore diagnosis from home or office becomes a possibility.
5. Design for peak performance. (This means performance is independent of what each operator display requests and what programs are running).

Desirable Features

After taking into account overall system and long term maintenance requirements we imposed the following desirable features:

1. Minimize cabling between the device to be controlled and the controllers. By keeping most cabling within the equipment racks we can alleviate the rats-nest of cabling found in most accelerators. In a number of cases (i.e. beam position monitors) [3] it is desirable to make the controllers an integral part of the equipment they controll (implies small physical size is desirable). Connection from controller to devices and their interface should be via a single, mechanically and electrically reliable connector at the back of the board. Replacing the controller should not require the removal of cables. This feature will provide for a considerable improvement in reliability by removing one of the main mechanical failure modes associated with control systems.
2. Should be able to work in an electrically noisy, physically demanding environment (implies no external digital buses, completely shielded, no forced cooling).
3. Hardware maintenance should not require a staff with sophisticated computer/digital logic experience. By requiring each controller to be bus independent from its neighbor a replace and swap approach can be used. Live insertion and extraction of controllers might be a desirable feature.
4. Local control and testing via a portable console of modest size and weight.
5. Large overall CPU and I/O bandwidth to minimize concerns over real-time issues. This would allow for programming without time being a consideration.
6. Arbitrary physical placement of controllers (topographic freedom) for optimum performance and functionality tuning. To make this economically viable requires a low cost controller as well as inexpensive support services such as chassis and power supplies.
7. Trade hardware cost for software wherever possible. Since hardware cost per function continues to decline as chip integration progresses but software costs continue to rise with manpower expenses, it is desirable to overkill in hardware functionality if software can be simplified.
8. For minimizing obsolescence, cpu performance should be commensurate with the bandwidth of the controlled device. For example, if a power supply is capable of 10Hz operation and the controller is at least as responsive, then we need not upgrade the controller until the power supply itself is upgraded.
9. User friendly and familiar interface (graphics, languages, spread-sheet, etc.).
10. Use of off the shelf software wherever possible.

Common Control System Characteristics

We also wanted to recognize some common characteristics of accelerator control systems:

1. They are well suited to parallel processing.
2. They are mostly asymmetrical in their data flow. The vast majority of data flows from devices to the operator rather than in the other direction.
3. They are highly distributed in functionality, but centrally connected. (i.e. like a human nervous system).

System Architecture

The system architecture (Fig. 1) is a refinement of a previous, all microprocessor based, system used on the SuperHilac Third Injector control system. [4],[5],[6] The primary change is the introduction, at the lowest level of the system, of the ILC, a custom designed controller. (Fig. 2) The ILC is the functional

*This work was supported by the Director, Office of Energy Research,Office of Basic Energy Sciences, Material Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

equivalent of a Multibus I chassis in the old system, except it controls fewer devices, has more processing power, and has a serial connection that runs at a bit rate 50 times faster. In addition, due to the ILC's considerably smaller size it is possible to place it in modest size equipment, and far closer to the devices it controls than would be reasonable for a bus based systems such as Multibus^{TM} or VME^{TM}. The next level in the architecture is represented by the Collector Micro Module (CMM), which resides in the control room. The CMM collects data from all the ILC-s and so contains the total accelerator database. This data is accessed by the third level consisting of the Display Micro Module (DMM) and the operator stations. This level is the operator and beam development groups interface to the control system. The DMM can also act as an intelligent database server for the accelerator since it has fast parallel access into the CMM. The highest level is the general network where file servers and development workstations reside. This level is also the gateway for experimenter access to the database.

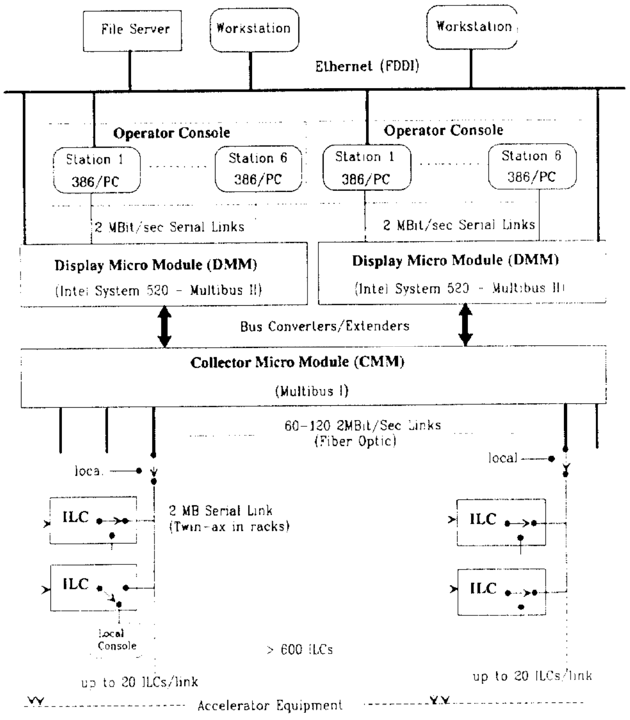

Figure 1. Control System Architecture

The Intelligent Local Controller (ILC)

The ILC controls the accelerator hardware via an Intel 80C186 CPU (running at 12.5 MHZ, for approx. 2 Mip performance) with an Intel 80C187 high performance math co-processor (approx. 0.2 Mflop). These two processors perform all of the algorithms required for closed loops, interlock checking, data averaging and sorting, etc. In addition they offload much of the work from the higher levels of the system (i.e. real to ASCII conversion, DAC/ADC units to engineering values conversion, etc.). The processors interface to a 64 Kbyte dual ported, battery backed CMOS static memory; 8 Kbytes of which contains a hardware protected monitor for loading of code, debugging etc., 8 Kbytes for the database, and the rest for user code, data, stack etc. The 64 Kbyte memory is also shared with an Intel 80C152 intelligent serial controller chip that services the 2 Mbit/sec serial link. This serial link connects the ILC-s and the rest of the control system. Because the 80C152 has a built in CPU (an Intel 8051 core), it need not disturb the main processor for the vast majority of communications requirements, but can independently access the database in memory, correct for errors, allow for remote reset, provide watchdog protection, etc. Up to six ILC-s can share a 19 inch rack mount chassis, but the backplane connecting them carries only a few lines such as power, the serial link, etc. (Fig.3). This means the ILC-s are mostly independent from each other, allowing easy removal and replacement upon failure. Live insertion is a possibility, and since no digital bus connection exists between ILC-s bus latchup cannot occur. Therefore if an ILC (connected to a non-critical accelerator device) fails it can be replaced without shutting down the rest of the ILC-s in the same chassis and their associated devices (which may be connected to more critical parts of the machine). Fault isolation is also straight forward since an ILC generally only controls a single device.

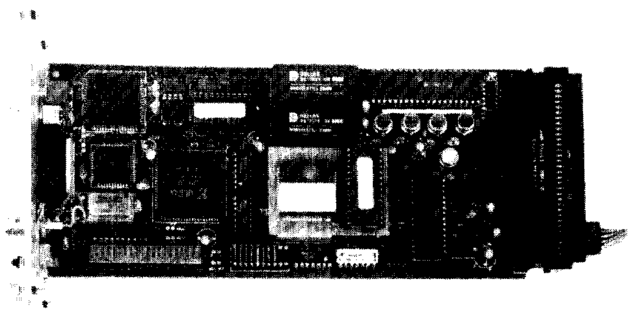
Figure 2. The Intelligent Local Controller

Programming for the ILC can be done in C or PLM on an IBM PC. This code and the database (which has all the necessary information for the ILC to run in a stand alone fashion) can be remotely down-loaded into the ILC-s. The database is generated via a DBASE^{TM} program running on an IBM PC. Programs in the ILC can be remotely debugged over the serial link via a source code driven debugger, or locally via an In Circuit Emulator (ICE) hardware/software debugger. A unique feature of the 80C186 allows ICE to function without having to remove the processor, thereby providing a very powerful tool for real time debugging. (Fig. 4) ICE provides for not only a 4000 instruction deep history of previously executed cycles (so we can tell what real time event led up to the error), but also has a very general set of hardware/software oriented breakpoint criteria.

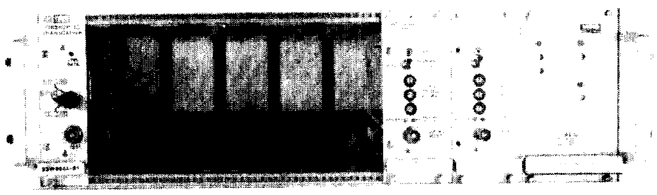
Figure 3. Chassis with ILC-s, power supply and optical interface.

Each ILC has four 16 bit DAC channels, four 13 bit ADC channels, and 24 bit lines for connection to the accelerator hardware. An SBX^{TM} connector is provided so we can take advantage of the more than 250 commercially available modules based on this interface. These modules can be used for connection to accelerator devices that require additional I/O (i.e. IEEE-488 connection, additional DAC, ADC channels etc.). All of the I/O interfacing is done via a single 96 pin Eurocard connection at the back of the board. The common lines, including two interrupt input lines (for computer timing synchronization), power, and the serial interface take up the bottom 18 pins of the 96 available. They are connected to the backplane shared by the ILC-s within a rackmount chassis (Fig. 3). The remaining pins are for the process I/O interfacing and have no connection to the neighboring ILC-s. Twenty-four pins are for DAC and ADC connections, 27 are for parallel or bit I/O, the rest are left open for I/O needed by the occasionally required SBX module. The serial link is also brought out to the front panel for local testing of individual ILC-s.

The ILC is a six layer, 3U high, 220mm deep Eurocard format board with the major components surface mounted for manufacturability. The current power draw is approximately five watts, due in large part to the extensive use of CMOS devices including a large XILINX Programmable Logic Array that provides almost all of the "glue" logic required. Due to the low power requirements of the ILC, it can be placed in a shielded, metal can, (for EMI/RFI protection), and needs no forced air cooling (Fig. 4). These features make it possible for the ILC to operate in electrically noisy and physically dirty environments. In addition the low wattage requirement of the ILC means smaller power supplies can be used, and the low heat dissipation implies increased reliability of the entire system. There will be in excess of 600 ILC-s, (approximate cost $500-$600 each) for a global CPU bandwidth exceeding 1000 Mip-s and 100 MFLOP. Future improvements to the ILC will increase CPU clock frequency to 16 Mhz, memory to 256 kbytes, ADC resolution to > 14 bits, decrease power consumption to three watts, and use more surface mount components. In addition for large systems, such as the SSC, the possibility exists to make hybrid equivalents of the board for increased reliability and smaller size. While the ILC meets more than 95% of our device control requirements, we still allow connection to PC-s, Multibus I (or Multibus II), or VME systems for special requirements. However we feel that a serially connected, low power controller, such as the ILC, is superior both in long term system cost, ease of maintenance, flexibility and reliability than currently available bus based systems. Furthermore development costs can be easily amortized (less than 15% of the cost of an ILC) for medium to large systems where volume production (in excess of a few hundred) allows the economies of scale to come into effect.

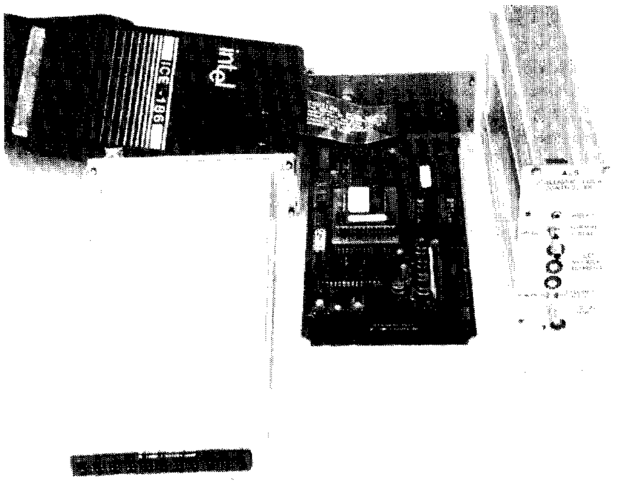

Figure 4. ILC with ICE, front panel and shielded cover.

The Collector Micro Module (CMM)

The ILC-s are connected via serial multi-drop links (each link will have no more than 20 ILC-s), using an SDLC/HDLC master-slave protocol. Within the racks (i.e. for short distances) twisted pair wiring (for rugged handling) is used via an RS-485 electrical standard; but for the long haul to the control room and connection to the CMM fiber optics is used. A dual wavelength fiber-optic module (Fig. 5) allows transmission and reception over a single cable, thereby requiring only a single fiber per link. The CMM can handle up to about 120 links, using commercial Multibus I CPU-s and a custom built SBX interface module (Fig. 6). The SBX interface module will contain four of the same 80C152 chips as in the ILC; two modules are required per Multibus board. (Incidentally since the same SBX module can be used on an ILC, it could act as a mini CMM to other ILC-s, and they in turn could be mini CMM-s etc.) Each Multibus board can handle eight links, and since the 80C152 has a built in CPU, it can do a great deal of the work dealing with the serial channel, unloading the processor (Intel 80386) of the Multibus board. This will allow each CMM board to run the eight links at the full bit rate of 2 Mbit/sec continuously. The packets are kept short (maximum 100 bytes), for low latency, with a small overhead (< 20 % on the average), so the "net useful" data rate will be about 1.6 Mbits/sec. These two features and a short link start-up time will provide a fast and responsive network for the kind of messages a digital feedback system requires.

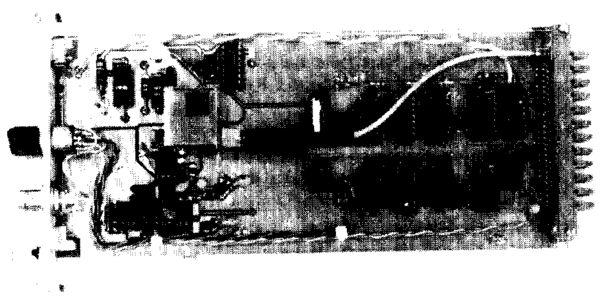

Figure 5. Dual wavelength fiber-optic serial interface.

To handle the 120 links we may have up to 15 Multibus CPU-s operating in parallel, but since the data received from the serial link is placed directly into the on-board memory, the Multibus is only minimally impacted. The CMM collects the data from the ILC-s and puts it into sections of dual ported memory reserved for each ILC's database. The software in the CMM is quite simple (less than ten pages of PLM code) since its function is primarily to do I/O to database memory copying. However, we do allow special messages to be used for ILC to ILC communication or to signal interrupts to the CMM (or the DMM).

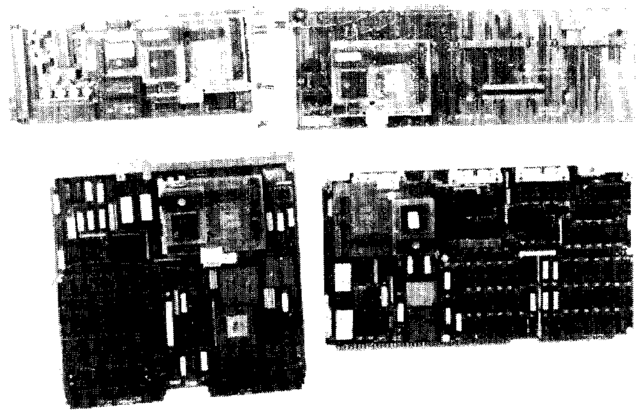

Figure 6. Multibus I, II, PC-AT bus, and ILC with SBX module.

The total database (i.e. the sum of all the ILC databases) is expected to be about 5 Mbytes. However since the part of the database that is fixed (names, conversion values, etc.) takes up the majority of memory, the total dynamically changing part is expected to be about 0.5 Mbytes. Since we will have at least 60 links, each with a net useful throughput of approx. 1.6 Mbits/sec (or 0.2 Mbytes/sec.), the total I/O bandwidth is on the order of 10-12 Mbytes/sec. Therefore the dynamic part of the database can be refreshed on the average about 15-20 per second. Furthermore by restricting the number of ILC-s on a given link even faster refresh rates can be supported for a few special devices. This would allow for digital feedback networks for some special beam position/orbit correction requirements. When a Multibus II (MB II) to MB II bus extender becomes commercially available (or Futurebus to MB II) we would replace the CMM with a MB II based system; this would allow for even bigger databases and easier connection to other types of links.

The Display Micro Module (DMM)

The DMM is a commercial (Intel System 520) MB II based system with multiple CPU-s (eventually 6 or more), each sharing a single disk, and using RMX-286™, a fast real-time operating system. Intel Corp. provides a MB II to Multibus I bus converter so the DMM can treat the CMM essentially as just a large block of memory where the database resides. We picked MB II for its numerous features that provide critical support for the kind of multiprocessing the DMM requires. MB II is a synchronous bus (thereby providing better noise immunity), requires built in self test for each CPU, and has hardware supported message processing. In addition, parity checking, geographic addressing (this allows self configuration), and virtual interrupts are provided. When combined with RMX-286, the system 520 greatly facilitates the kind of multiprocessing we wish to do, by allowing all CPU-s to share a single disk and network interface, booting individual CPU-s and restarting them separately, and eventually allowing dynamic distribution of tasks and jobs among the CPU-s. A multi-windowed human interface is provided by Intel for managing the multiple CPU-s. One can think of the system as a "network in a box" with an extremely high bandwidth (up to 40 MBytes/sec.).

Programming for the DMM is done on target; C, PLM, Pascal, Fortran are available as languages. A debugger as well as binders and builders are provided along with an interactive system configuration utility. Because RMX-286 is a full featured real time operating system, the DMM will perform all time critical functions. This includes timer driven database service to the operator stations and beamline experimenters, and general functionality of a fast database server. While the Intel system 520 is new, our previous experience with an earlier Multibus II based system as well as experience from the Superhilac Third Injector control system (including familiarity with languages and parallel processing architectures) [4],[5],[6],[7],[8],[9] will allow us to use it for running a test stand (for the Linac part of the accelerator) as early as April of 1989 (only 1.5 years into the project).

Operation Station

From each Multibus II CPU there is a connection, via the same type of serial link connecting ILC-s to the CMM, (we use the same SBX module as in the CMM (Fig. 6)) to the operator station which is a high performance Personal Computer (PC). The PC-s offload the DMM from having to carry out the human interface aspects of the control system, by doing the color graphics, handling mouse input, knob input, etc. A custom knob panel has three knobs with 16 characters of LED display for identifying the device controlled. It is self contained, since it includes an Intel 80C152 chip; all that is needed to run the panel is power and a twisted pair for the serial I/O (Fig. 7).

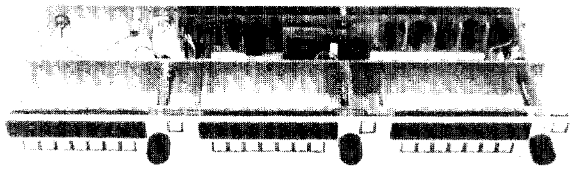
Figure 7. Intelligent knob panel with LED-s.

The DMM and the PC-s perform concurrent processing (i.e. while the DMM is gathering data, the PC is displaying it); this allows the DMM to do what it is best suited for, i.e. dealing with rapid access to the database and handling time critical functions. This concurrency also allows us to do remote instrumentation (oscilloscopes, etc.) [2] with fast operator response. Six operator stations make up a console; the plan is to have at least 2 consoles. (i.e. two DMM-s, and 12 PC-s). The DMM-s and PC-s also have connections to an Ethernet (eventually FDDI) network to which file servers and program development PC-s and Workstations are attached. This network will be the gateway for experimenter access into the database. All requests to change data will go through the DMM and will be password validated.

The choice of a PC for the operator station has proved fortuitous in a number of areas. First, for our portable console we can use a standard laptop/portable computer, and its human interface can look very much like an actual operator station (Fig. 8). This portable console can be used for local testing, debugging, and running an ILC or several ILC-s residing on the same link. Second, this interface should be familiar to the operator and even the casual user since a large number of people have used PC-s in other parts of their jobs (word processing, spread sheets, etc.). Third, we can take advantage of the wealth of inexpensive commercial packages, such as EXCEL^{TM}, DBASE IV^{TM}, Micrografx Designer^{TM}, MS Windows^{TM}, etc. that are available now, and will become available for OS/2^{TM} shortly. We intend to use OS/2 with Presentation and LAN Manager as our final PC operating system and environment.

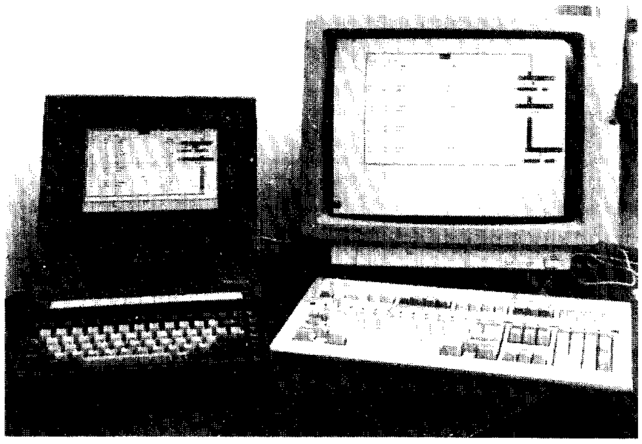
Figure 8. Portable console and PC operator station.

The user has a number of ways to interface to the control system. He can write programs in C, Pascal, Fortran, and Basic (interpreter and compiler) that can access the database via a set of standard procedure calls. This approach is well suited for algorithm development by both an experienced or inexperienced programmer. Codeview^{TM}, Quick C^{TM}, and Quick Basic^{TM} are available as debuggers or interactive development tools.

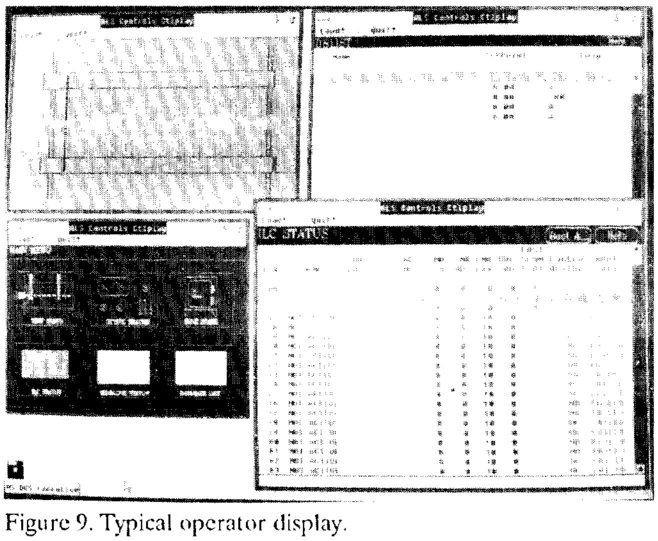

Figure 9. Typical operator display.

However, we also wanted to allow display generation via an object oriented approach either via languages (C++ or Objective C) or graphically. For the latter we currently use Micrografx Designer, as a graphics editor (Fig. 9), to generate pictures and symbols to which an 80 character descriptor can be attached. This descriptor is used to make connections to database values, list drivers, and other pre-canned routines (Fig. 10). This makes it possible to interactively develop list driven interfaces (we wrote a "virtual terminal server" in the DMM). Using the same mechanism we are also able to draw pictures to which we can connect spread-sheets (EXCEL) via the Dynamic Data Exchange protocol available to MS Windows and OS/2 applications (Fig. 11). This means that one can attach numerical functions, conditions, etc., just by setting values or equations in a spread-sheet rather than having to write code, compile, link, and test it. This allows quick exploration of what-if scenarios. Designer can also import CAD (such as Autocad) (Fig. 10) generated pictures, making possible the use by the control system of layouts generated by mechanical/electrical engineers. As OS/2 takes off we expect additional tools to aid in object oriented, graphic based program development. Using all of the above mentioned tools operators can interactively create individualized displays and interfaces for running the accelerator.

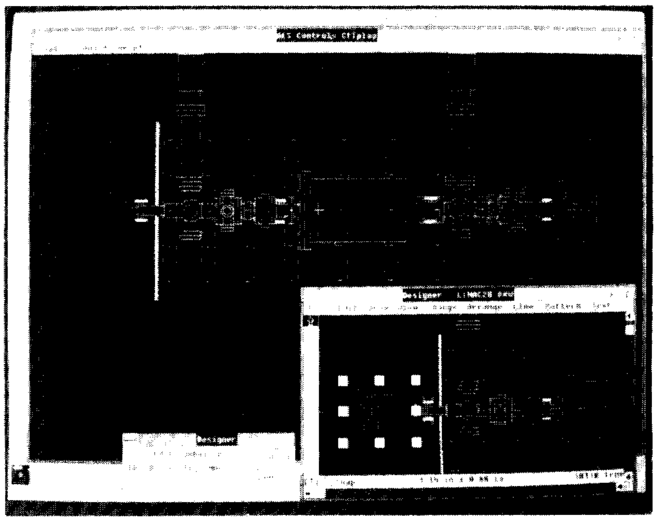
Figure 10. Micrographix Designer and CAD drawing.

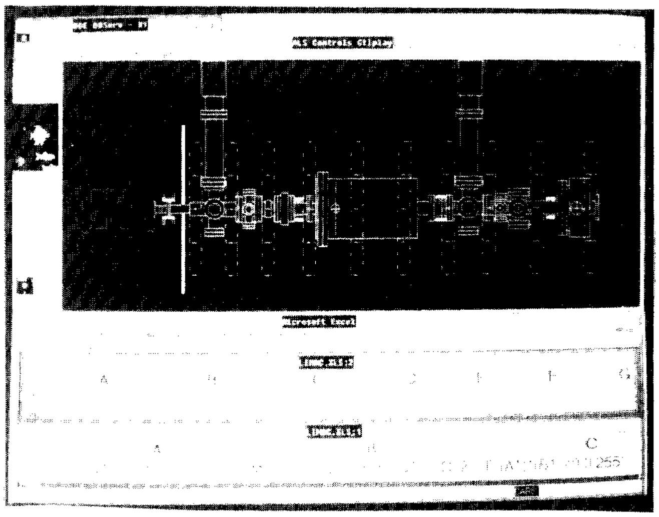
Figure 11. EXCEL spread-sheet and CAD drawing.

Manpower, Modeling, Cost and Schedule

Staffing for the project consists of five programmers, one digital designer and a fabrication coordinator. We receive additional support from accelerator physicists who are helping us with auto-tuning, orbit correction algorithms, and operations scenarios. In particular a simplified version of the code [10] developed by H. Nishimura to validate the lattice design is being adapted by him for actual operation of the accelerator. By using such a specialized code we hope to be able to run orbit correction at about a 10Hz rate. The total construction budget (over the 4.5 year schedule) is $4.4 million. Approximately $1.4 million is for hardware (including the ILC-s) the remainder is for software (including physics support) and installation. The schedule calls for running of a Linac test stand by mid 1989, booster operation by 1991 and full operation in 1992.

Conclusion

The ALS control system is a very high performance (both in CPU and I/O bandwidth), technology driven, control system with an innovative architecture. The ILC, and its serial interconnection scheme, could become the prototype for an SSC type controller (a hybrid version of the ILC is a possibility for further reduction in size and increase in reliability). Furthermore, the serial connection and the modular nature of the ILC makes the system easily maintainable by allowing simple diagnosis and replacement since live insertion is a possiblility. The use of parallel processing (at all levels) of the architecture will guarantee a very responsive system. Since we depend on commercial buses in only a small fraction (CMM, DMM, PC-s) of the system, we can easily adapt to newer standards such as Futurebus, EISA , etc. The choice of using commercial software and PC-s for front ends insures a friendly and flexible human interface, and is one of the factors that makes possible the low staffing for this control system.

References

[1] Marx J., "The Advanced Light Source at Lawrence Berkeley Laboratory", This conference.

[2] Chin M., et al., "Networking Remote Instrumentation for the Advanced Light Source (ALS)", This conference.

[3] Hinkson J., et al., "Advanced Light Source (ALS) Beam Position Monitor", This Conference.

[4] Lancaster H.D., et al., "A Microcomputer Control System for the SuperHILAC Third Injector", Proc. of the 1979 Linear Accelerator Conf., Brookhaven National Laboratory BNL-51134.

[5] Magyary S., et al., "A High Performance/Low Cost Accelerator Control System", IEEE Trans. Nucl. Sci. NS-28, (1981)1461.

[6] Magyary S., et al., "Operating Experience With a New Accelerator Control System Based Upon Microprocessors", IEEE Particle Accelerator Conference, March 1981.

[7] Pines H., "Automatic Tuning of the LBL SuperHILAC Third Injector Transport Line", IEEE Trans. Nucl. Sci. NS-30, (1983)2314.

[8] Lancaster H.D., et al., "A High Performance Control System for a Heavy Ion Medical Accelerator", IEEE Trans. Nucl. Sci. NS-30, (1983)2311.

[9] Timossi C., "Error Monitoring for the SuperHILAC Third Injector Control System", Lawrence Berkeley Laboratory report LBL-20331.

[10] Nishimura H., "TRACY, a Tool for Simulations", European Particle Accelerator Conference, Rome 1988, Lawrence Berkeley Laboratory report LBL-25236.