Computer Control

W. Daley & R. Carey
GTRI, Georgia Institute of Technology, Atlanta, GA 30332

U.1. INTRODUCTION

This chapter discusses both the system and subsystem control computers. The purpose of the system control computer is to provide a single machine that will allow a user to control the operation of the CHARA Array. This computer will allow the user to acquire astronomical data, align optics, and perform diagnostics and is also responsible for interfacing with remote users and providing them access to the system. The subsystem control computer, on the other hand, communicates between the system control computer and each of the subsystems, controlling motors, solenoids, *et cetera*, and transmitting status reports on each of the subsystems to the central computer.

U.2. SYSTEM CONTROL COMPUTER

U.2.1. Functionality

The system will provide a graphical user interface with the following functionality:

- Communicate directly with each subsystem
- Control all subsystems together as one unit
- Acquire and store data
- Display the state of the optics graphically
- Display environmental conditions
- Display any subsystem error conditions
- Control individual telescopes and all subsystems

U.2.2. Requirements

The system computer needs to:

- Perform multiple tasks concurrently
- Share file information with other computers
- Communicate simple commands and data to subsystems
- Monitor and record real-time control information of the subsystems
- Provide Internet access for remote users
- Provide reliable and maintainable software

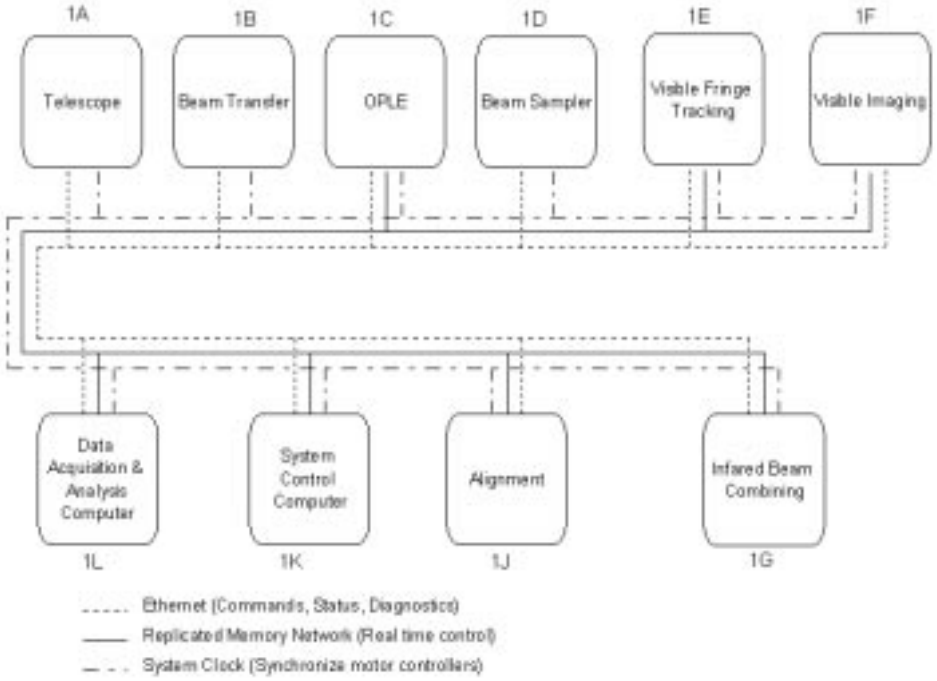

FIGURE U.1. Communication and control signals.

U.2.3. Design

U.2.3.1. Hardware

A SUN Microsystems SPARC 10 was chosen for the system control computer. This machine uses the UNIX based Solaris 1.1 operating system which will allow multiple users to log-on and multiple programs to be executed concurrently. The SPARC 10 supports an ethernet network interface which will be used to send simple commands and data to the individual subsystems. This network will also be used for aligning and trouble-shooting problems with the system. The operating system supports the Network File System (NFS) which will be used to transfer files between the various subsystems.

The workstation will have a replicated memory card installed, which will be connected by fiber-optic cables to replicated memory cards installed in each of the subsystem computers. This real-time replicated shared-memory system will connect computers at high data speeds (150 Mbits/sec) with minimal application-to-application transport delay. This network will allow the system control computer to monitor the status of the control variables located on the individual subsystems, by looking at its local memory located on the replicated memory card. The network will also allow the subsystem computers to do real-time control synchronization by using local variables located on the replicated memory cards.

The system control computer will generate a TTL clock which will be distributed to all the subsystems over a fiber optic network. This clock will be used to synchronize all the motor controllers to insure stable system operation. The connections between the system computer and the other subsystems can be seen in Figure U.1.

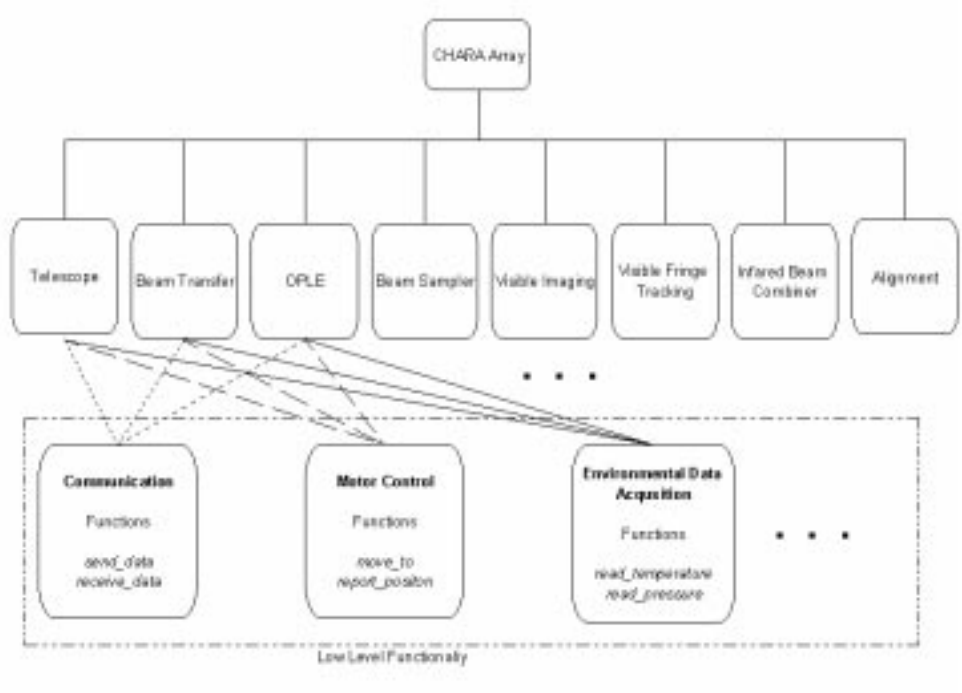

FIGURE U.2. Software object design.

U.2.3.2. Software

The software will be developed in the C++ language, which supports object oriented software design and implementation. The use of object oriented programming (Figure U.2) will ensure that the software is developed so that it is reliable and maintainable. This is done by forcing the consideration of total system functionality and design before any code is written. The program is designed as a number of objects that interact with each other through well-defined interfaces. This means that all objects in the program have to be identified and the data flow between them determined. Once an object has been developed and its associated functions have been debugged, it can be used on any subsystem that has similar requirements without any modification. It also means that an object can be updated without interfacing with the rest of the system.

The Graphical User Interface (GUI) will be developed using the X Windows standard and the Motif widget toolset. This GUI supports widgets that implement pop-up windows, scroll bars, push buttons, list boxes, edit boxes, etc. These widgets will allow the user to interact with the GUI using a mouse and keyboard.

The GUI will allow objects, like mirrors, telescopes and beam paths to be drawn graphically on the screen. The user will manipulate these objects by clicking the mouse on the object and dragging it to its new location. This will be useful in inserting or removing a mirror from the optical path, for example.

U.2.4. Risks

The technical risks associated with the successful completion of this aspect of the project would be minimal.

U.2.5. Options

There are many options to the various parts of the system control computer but only a few possibilities will be discussed here.

The system computer could have been another type of workstation such as HP, IBM, Silicon Graphics and DEC but they each have their limitations of networking, processing speeds, hardware and software availability. The SUN workstation provides the most cost effective solution to the problem. An IBM 486 computer could be used but it would have to run a multi-tasking, multi-user operating system to successfully implement the system control computer. The operating systems that currently run on IBM 486s that meet these requirements are UNIX and Windows NT. SUN's UNIX implementation is a lot better than any UNIX that runs on a PC, and Windows NT is an unproven product.

The replicated memory network could be replaced by a deterministic network like a token-ring network. This solution would require that software be developed to handle the message passing between the various subsystems. Although this is not an impossible task, it adds a level of complexity to the system software design, that makes it harder to develop and extremely difficult to maintain.

C could have been chosen as the development language, since it is well known and widely used. Although C is a powerful language, it does not force the developer to write maintainable software. Instead, it leaves it up to the developer to ensure that there are no conflicts between the various modules in the system. In very large software projects like the CHARA Array, a lot of diligence and documentation is required on the part of the developers, to ensure that there are no side effects from one module interacting with another. In C, there is also no well defined mechanism to ensure that code developed for one module will work in another module. C++, on the other-hand, eliminates side effects by hiding the data and function implementations from the rest of the program. This ensures that data is only manipulated by member functions that use the proper procedures, thus eliminating adverse side effects which in most cases reduce the integrity of the software.

U.3. DATA ACQUISITION COMPUTER

While the system control computer is responsible for the real time control aspects of the Array another system is required for data logging and reduction. This system will be required to log any data produced by the fringe tracking and imaging systems, produce real-time estimates of visibilities and allow access to, but not control of, many of the Array subsystems. This computer will allow the operator, or a visiting astronomer, to inspect data 'on the fly' and perform basic data reduction without interfering the the operation of the Array. A Sun Sparc or equivalent Unix machine connected to the local area network should be adequit for this job.

U.4. SUBSYSTEM CONTROL COMPUTER

U.4.1. Functionality

The subsystems will provide the following functionality:

- Communicate with each subsystem and central controller

- Control necessary control devices (motors, solenoids, etc..)
- Transmit current subsystem status to the central controller
- Transmit error conditions to the central controller

U.4.2. Requirements

The subsystem computer needs to:

- Perform multiple tasks concurrently
- Respond to simple commands from the system control computer
- Control hardware devices (motor controllers, digital I/O, etc.)
- Monitor environmental conditions
- Provide reliable and maintainable software

U.4.3. Design

U.4.3.1. Methodology

There were two hardware design methodologies considered:

The first method was to make the initial cost of the system as low as possible while providing the desired functionality. This could be accomplished by determining and defining all the communication protocols between each subsystem. As long as the communication protocols were well defined and adhered to, it would not matter what hardware platform was used for each subsystem. Each subsystem could then be optimized to be the most cost effective for its functionality. For example, a subsystem that monitors environmental variables and controls a set of mirrors could be built on a low-cost STD bus system, whereas a system that needs to acquire real-time data and perform FFT processing would use a VME type system with DSP processor cards installed. Although this design methodology would produce the cheapest initial system, it would be very expensive to maintain in the long run. This is because there would have to be someone who is familiar with each of the different hardware and software architectures used. Such individuals are typically hard to come by. Also, the stocking of spare parts to fix failed boards would be cost prohibitive, since spares would have to be stocked for each different subsystem.

The second design methodology was to make the software development and long-term maintenance as simple and cost effective as possible. This was done by using the same base computer system in all the subsystems. While this might be overkill for some subsystems, it will make for a more reliable and maintainable overall system in the long run. The CHARA solution is to specify a BUS architecture, the main processing board, and a real-time operating system. Various I/O, D/A, A/D, and DSP boards are specified that can be used if needed by the subsystems. This means that there is a base configuration that is the same across all subsystems, with modularity built in with the choice of interface boards. This solution allows software developed for one subsystem to be used on another subsystem, and the same spare parts can be stocked for all subsystems.

Choosing the correct BUS was very important and the following Buses were considered because of the availability of real-time hardware and software:

- PC
- Multibus

- STD
- STD-32
- VME
- VXI

Table U.1 shows a comparison of these options; the different category ratings are discussed below.

TABLE U.1. Comparison of available BUS architectures.

BUS	Ethernet Support	DSP Support	A/D, D/A, Digital I/O Support	Motor Controller Support	Real-Time O/S Support	Cost	Reliability
PC	Very Good	Very Good	Excellent	Excellent	Good	Excellent	Fair
Multi	Good	Good	Good	Good	Good	Good	Good
STD	Good	Fair	Excellent	Good	Good	Very Good	Excellent
STD-32	Good	Good	Excellent	Good	Good	Good	Very Good
VME	Excellent	Excellent	Excellent	Very Good	Excellent	Good	Very Good
VXI	Good	Fair	Very Good	Good	Very Good	Good	Good

U.4.3.2. Category Rating Description:

- Ethernet Support

Excellent	CPU cards have ethernet ports built in, and software drivers supplied
Very Good	Ethernet cards and drivers are available from many vendors
Good	Ethernet cards and drivers are available from a few vendors
Fair	Ethernet cards and drivers are available from one vendor
Poor	No ethernet cards or drivers available

- DSP Support

Excellent	Many different vendors with very high speed computational boards
Very Good	Many different vendors
Good	A few different vendors
Fair	One vendor
Poor	No vendors

- A/D, D/A, & Digital I/O Support

Excellent	Many different vendors with many different boards
Very Good	Many different vendors
Good	A few different vendors
Fair	One vendor
Poor	No vendors

- Motor Controller Support

Excellent	Many different vendors with many different boards
Very Good	Many different vendors
Good	A few different vendors
Fair	One vendor
Poor	No vendors

- Real-Time O/S Support

Excellent	BUS supports many different processors and O/Ss
Very Good	BUS supports many different O/Ss
Good	BUS supports a few different O/Ss
Fair	BUS supports one O/S
Poor	None

- Cost

Excellent	Many low-cost boards and software (<\$500)
Very Good	Few low-cost boards and software
Good	Many medium-cost boards and software (\$500 – \$1500)
Fair	Few medium-cost boards and software
Poor	Only high-cost boards and software (>\$1500)

- Reliability

Excellent	Very high MTBF, very good mechanical support for cards
Very Good	High MTBF, good mechanical support for cards
Good	Average MTBF, average mechanical support for cards
Fair	Low MTBF, poor mechanical support for cards
Poor	Very low MTBF, very poor mechanical support for cards

Assigning a scale of 1 – 5, with 5 being excellent and 1 being poor, gives the following overall rating shown in Table 0.2.

All categories were determined to be of equal importance and no weighting was used. Based on this selection criteria the VME BUS scored the highest and was chosen for the subsystem controllers. The available software tools and development environment were not taken into consideration but will be discussed in a later section.

U.4.3.3. Hardware

A Motorola–based CPU board was chosen as the central processor for the VME BUS because it is widely supported by the real-time operating systems. The CPU board will have the following specification:

- 25 MHz MC68040 microprocessor with 8KB of Cache

TABLE U.2. Rating of available BUS architectures.

BUS	Ethernet Support	DSP Support	A/D, D/A, Digital I/O Support	Motor Controller Support	Real-Time O/S Support	Cost	Reliability	Total
PC	4	4	5	5	3	5	2	28
Multi	3	3	3	3	3	3	3	21
STD	3	2	5	3	3	4	5	25
STD-32	3	3	5	3	3	3	4	24
VME	5	5	5	4	5	3	4	31
VXI	3	2	4	3	4	3	3	22

- 8 MBytes of DRAM
- Ethernet interface
- Sockets for on-board ROM/EPROM

The base system will also contain a replicated shared-memory fiber-optic network card, which will allow each subsystem to have an exact copy of a block of memory. This real-time replicated shared-memory system will connect computers at high data speeds (150 Mbits/sec) with minimal application-to-application transport delay. Variables stored in this block of memory will be available on all subsystems as local variables. This will make developing the system software much easier, since time will be spent developing software to control the subsystem as opposed to implementing complex communication schemes between the individual subsystems. For example, the OPLE subsystem needs error information from the fringe tracking subsystem, which can be implemented by having the fringe tracker write the error information to a specific memory location and then set a flag in another memory location. The OPLE subsystem would read the error information when the flag is set, and then clear the flag. In this approach, no software development need be expended on the communication between the OPLE and fringe tracker, since it is all handled in hardware. The replicated memory system will also be invaluable for trouble-shooting and system debugging, since the control variables of each subsystem can be seen and recorded at the central computer.

The remaining cards that will go in the VME chassis will be optional and depend on the functionality of the subsystem. A diagram of a typical subsystem can be seen in Figure U.3.

U.4.3.4. Software

The software will be developed in the C++ language, which supports object–oriented software design and implementation. The use of object–oriented programming will ensure that the software is developed for reliability and maintainability, since it forces the software to be completely designed before any code is written. It is anticipated that the CHARA Array will use an open system that will adjust to the needs of its users. The software should thus be designed to accomodate these changes and modifications as painlessly as possible. The program is designed as a number of objects that interact with each other through well-defined interfaces. This means that all the objects in the program have to be identified

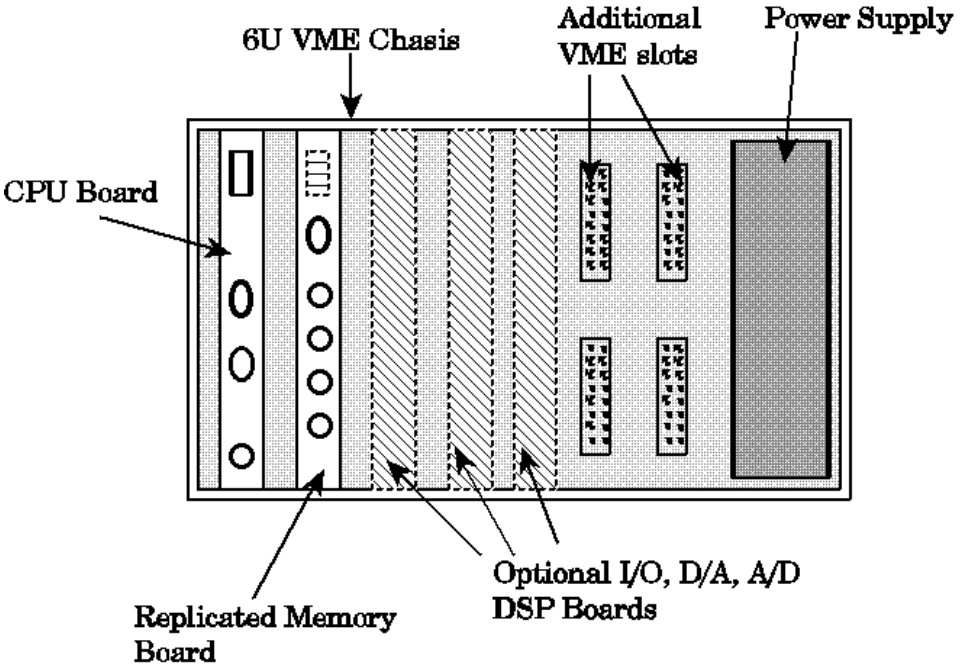

FIGURE U.3. Subsystem controller.

and the data flow between them determined. Once an object has been developed and its associated functions have been debugged, it can be used on any subsystem that has similar requirements without any modification.

VxWorks by Wind River Systems is a real-time operating system that will be used on the subsystem controllers. This UNIX-like operating system will allow multiple tasks to run concurrently while maintaining their priority schemes. This means that the task with the highest priority will execute whenever it needs to, while other lower priority tasks will be put on hold. VxWorks also allows the user to log in from remote locations and interact with tasks running on the system.

The software development environment will consist of a product called ObjectCenter from CenterLine Software, Inc. This product is hosted on a Sun Sparc Station and provides an intuitive user-friendly environment to develop, debug, and document C++ code.

U.4.4. Risks

The technical risks associated with the successful completion of the subsystem design and integration are not significant. The areas that would be of most concern would be the development of low-level device drivers to control various pieces of hardware. In most cases this should not be necessary, as it is anticipated that we will be able to purchase most of these drivers with the associated equipment.