Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

An Effective PC/Linux Based Control System Upgrade for the UH 2.2-m Telescope

R. Lovell

Intelligent Stage Research Lab, Arizona State University

E. Sousa, M. Inouye, K. Maesato, A. Pickles, J. Tonry

Institute for Astronomy, University of Hawai'i

A. Mizoshiri, P. Oshiro, L. Amano, H. Motomura

Hawai'i Community College, Electronics Department

Abstract. The University of Hawai'i 2.2-m telescope is a general purpose optical/infrared telescope with excellent wide-field and high resolution (fast tip-tilt) capabilities. An LSI-based TCS with custom-built servo electronics has been successfully used for 20 years, but is now the principle constraint on the telescope's continued efficient operation. We are implementing a modern control system with the following criteria for effective control at modest cost: (1) improve existing performance for slewing, setting, pointing, tracking and offsetting; (2) provide reliable software operation with remote oversight, diagnostic and future remote operation capabilities; (3) utilize existing hardware where appropriate; and (4) provide a user friendly and easily upgradable system to enable efficient interaction between telescope and multi-instrument control.

The system we are implementing derives from the successful TCS upgrade at the Michigan-Dartmouth-MIT (MDM) Hiltner 2.4-m telescope, with significant differences. It uses a PCI based PC running Linux, a 3-axis Galil DC servo motion control card for the 2 telescope and dome axes, modern relay logic control hardware, and existing drive and encoder mechanisms.

1. Differences Between UH and MDM Systems

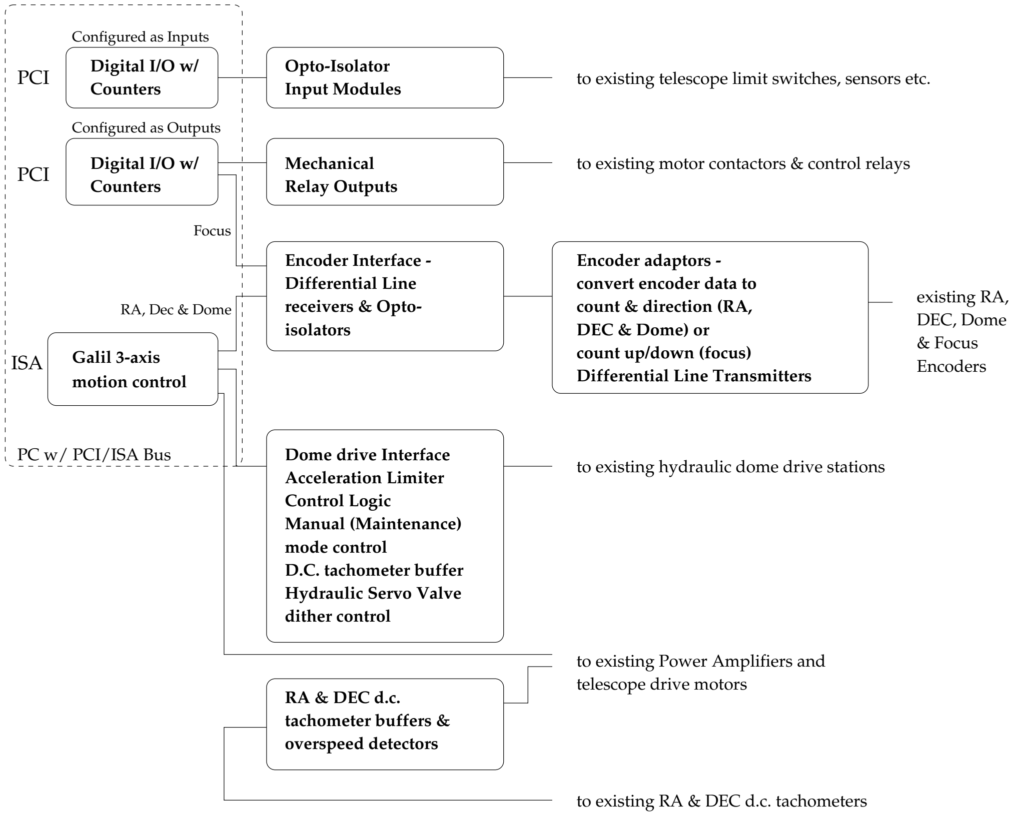
The main differences between the new UH system, and that previously and successfully employed at MDM lie in the choice of operating system and in the specific hardware differences between the two telescopes, thus:

- Linux operating system (UH), vs. OS/2 (MDM),
- DC servo motors (UH) instead of stepper motors (MDM),
- Galil 3-axis controller (at UH),
- Additional hardware features controlled by the UH TCS include: shutter, dome lighting & heaters, windscreen, crane and hydraulic platform,
- Maintenance and observation modes of operation,

- Direct remote network access (password controlled) to the TCS,
- Integral T-Point data collection and pointing model calculation modes.

2. New TCS

The new TCS removes the old electronics from the control room entirely. The PC is a 19-inch industrial rack-mount chassis with ISA/PCI passive back-plane, Pentium-II single-board CPU and Galil 3-axis motion control and interface cards. This is mounted in a new rack, together with opto-isolators and other signal interface circuitry, and located in the lightning protected air-conditioned computer room under the telescope. This rack is interfaced to the existing 24-volt relays and telescope encoder and drive system. The control room is reserved for monitors and keyboards, and a small engineering console for maintenance work.



3. Software

Designers: Robb Lovell (ASU), John Thorstenson (Dartmouth College).

The 88 inch TCS is written in C++ and is designed to be relatively easily modified to control telescopes that use distinct control hardware. The software is written in six functionally independent, but interrelated layers:

1. Hardware manipulation,
2. Device control,
3. Telescope interface,
4. Time keeping and Data Acquisition,
5. Astronomical calculations,
6. User interface.

The operational part of the code consists of series of 22 independently looping processes that handle different functions. Each process sleeps and wakes up ten to two hundred times a second depending upon the criticality of its function.

These procedures communicate information through a *"share everything"* memory scheme that uses semaphores to control access to writing of data. The system is currently controlling the Hiltner 2.4 Meter telescope, and McGraw-Hill 1.3 Meter telescope, both on Kitt Peak in Arizona. Both of these systems are implemented on top of OS/2 Warp. The UH88 TCS has an identical design philosophy, but is implemented on top of Linux.

At the lowest nitty-gritty level of the program are the devices that interface to the telescope. These are the digital I/O cards, motor control cards, counter timer boards, and A/D and D/A cards. Control of each card is device specific and must be re-written for any new hardware implemented. These **hardware interfaces** are implemented as C++ classes that encapsulate each board's functions.

Above the **Hardware layer** is an abstracted layer called **Device Control**. This layer implements control of DC motors, stepper motors, switches, encoders, switches, signals, push buttons, toggle buttons, encoders etc. These objects use the hardware manipulation layer to implement standard interfaces to these devices. For instance, the DC motor is implemented as a C++ object that has (as the data of a C++ class) the location in the hardware of the signals that control the motor. There are many different ways to control a motor: some use a direction signal and an on/off switch, others might use an enable and an analog voltage. These signals are located in the hardware layer. But these signals are not of concern to the higher levels of the program. The motor is abstracted as a set of procedures that specify speed and direction the motor is to move.

These procedures can be called regardless of how the motor is implemented in the hardware. For each category of device, several different hardware implementations are written. Each hardware layer class contains the intelligence necessary to convert speed and direction into manipulations of the hardware interface, to control the device.

The **Telescope interface** layer consists of looping processes that use several of the device control layer objects to manipulate the telescope. This layer controls tracking, positioning, front panel switches, guiding, paddle control, focus control, dome control and other things like platforms, cranes, and windscreens. Each telescope element is controlled through the coordinated manipulation of several devices: DC motors, switches, etc. and feedback from others: encoders, signals, etc.

The **Time keeping and Data acquisition** layer is responsible for reading and updating the state of the telescope. This process is the fastest looping process in the system so that no other procedure will miss necessary data in coordination with other processes. This procedure locks out any reading of shared variables while their values are being updated so that no corruption of data will occur.

The **Astronomical calculation** layer performs all the calculations related to transforming mean location into telescope position. Essentially a telescope is a robot and, because of this, it is useful to separate astronomical control from robotic control so that changes in the astronomy do not require changes in the code that moves the telescope. All telescope manipulations at the hardware level are performed as if the telescope were a robot with no relationship to astronomical position. Both RA and Declination are manipulated in encoder increments within the processes that actually turn on and off the motors and read encoders. In this way, expensive calculations are not required at this level of the code, and an efficient feedback loop is easily implemented.

Contained in the astronomical calculation layer is the pointing model, transformations from mean location to encoder location and back, transformations in time between Julian, Universal, and Sideral etc. The program makes use of Skycalc written by John Thorstenson of Dartmouth College to implement these calculations. The pointing model used is based upon T-Point code (Pat Wallace, `http://www.tpsoft.demon.co.uk`).

The **User interface** layer consists of the display of relevant data to a terminal window, and communication of data to and acceptance of commands from the outside world. The display is implemented in the `ncurses` environment. This has the advantage of being implemented with the minimum throughput required to the window from the program. For remote control of the telescope this is a great advantage. Any X-window or simple terminal window can accept the data to be displayed. The display layout is modifiable, through a configuration file of about 180 possible pieces of information that records the (X,Y) position of each displayed item. Displayed items include relevant program variables, displayed text, messages, and error conditions.

Communication to the outside world is currently through the serial port on the machine using a protocol invented at the Hiltner 2.4 Meter telescope. At the time of this writing, a new design for this protocol is being explored so that communication can occur through UDP network connections.

Inter-communication between instruments, guider and telescope control computer is via TCP/IP protocol (vs. serial port previously). A suite of telescope commands will be available for incorporation into observer scripts so that optimized integration of instrument and telescope operation can be achieved. This library is expected to evolve with new instruments and telescope configurations. The TCS includes an assist mode for the collection of T-Point data for modeling the telescope flexure and pointing. The new TCS includes a substantially modified display monitor for status, target position, offset, focus, guide status and next target, together with a reduced set of information appropriate to a remote control/display.