Hartebeesthoek 26m

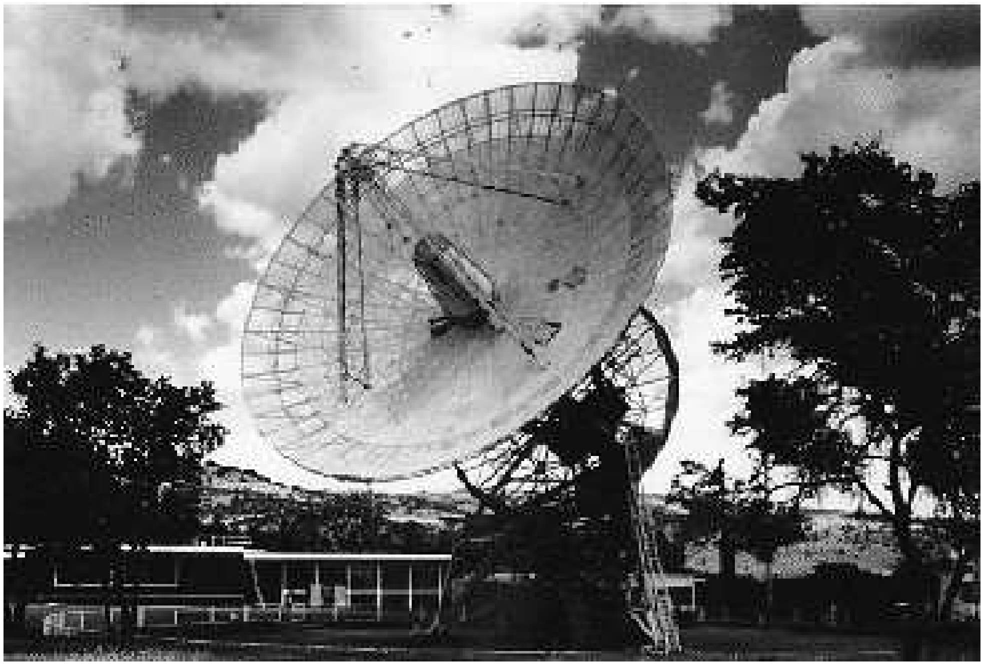
New Computer Control System (NCCS)

DESIGN

version 2.0

Jonathan Quick,
Andy Götz,
Mike Gaylard

July 30, 1997



DESIGN
DESIGN
DESIGN
DESIGN
DESIGN
DESIGN
DESIGN

Contents

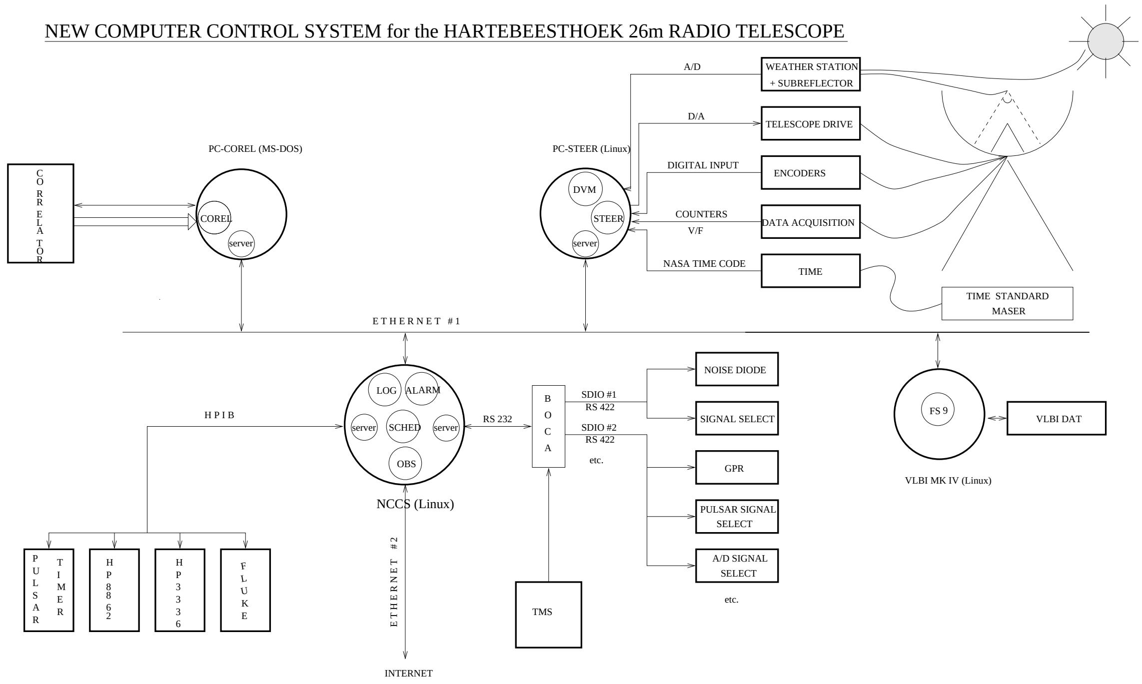


Figure 1: Overview of the New Computer Control System (NCCS)

1 Introduction

The Hartebeesthoek 26m telescope computer control system is currently being redesigned and rebuilt. This document will present the latest design of the New Computer Control System (NCCS).

The design of the NCCS is intended to fulfill the list of requirements specified in the original *Requirements & Specifications* document. In order to present a coherent view, this version of the *Design* document has been extensively rewritten to reflect the various implementation phases subsequently defined and/or realised by the *NCCS Working Group*. An overview of the new NCCS is presented at the beginning of the *Design* document.

The design presented in this document aims to satisfy all of the **mandatory** requirements of the *Requirements & Specifications* document i.e. all requirements marked with $\surd$. Most if not all of the **desirable** requirements i.e. those marked with a $\heartsuit$, will only be satisfied in further extension phase(s) of the NCCS project, beyond the scope of this document.

2 Overview

An overview of the proposed design of the NCCS is presented in figure 1.

The **NCCS** will be a *distributed control system*. Each hardware and software module in the NCCS will be modelled as an *object*. Each *object* will belong to a *class* and have its own *data*. The *class* implements and determines the *behaviour* of the *object*.

Objects will be distributed over a number of *computers* running the *Linux* operating system. As many computers will be used in the NCCS as performance and design requires. At present dedicated computers are required for PC-STEER, PC-COREL and the main computer, PC-NCCS.

The **computers** will be linked via a dedicated *local area network* (Ethernet). Communication on the network will be based on *Sun remote procedure calls* (NFS/RPC). The ESRF *device server* technology will be used to handle *network address resolution*, *device resource database* and the *root device class.*

Instrumentation control will be done via the locally developed and maintained *SDIO* (Serial Digital Input and Output) bus together with suitable remote interface units.

Fast data acquisition will be done using *Voltage to Frequency Converters* (V/Fs).

The **user interface** will be ASCII and graphics capable based upon *TK/TCL.* TCL will be extended to support highlevel routines required for observing and setting up hardware configurations.

In what follows each development phase of the NCCS document will be treated individually. Each phase is broken down into a set of required components in rough order of completion, identifying the tasks which need to be done (marked with a $\Longrightarrow$) and designating the person responsible.

3 Phase 1 - Limited VLBI Support

Although the original definition of Phase 1 was the support of all facilities currently available on the HP1000 system, it became evident that this task needed to be broken down into smaller units. Several precursors of the NCCS currently form a completely independent VLBI observing system, which forms in essence the core of the new system. The first phase of the project was thus envisaged to be the reworking (in an integrated fashion) of the hardware and software components of the embryo VLBI system into their final NCCS form.

3.1 Development Environment *(J. Quick)*

This has been defined in the *Code Development Environment Document.* The development area is currently set up on `apus` in the `/home2/local/src/nccs` directory. Note that this area is for development only, independent operating copies of all necessary software will be kept on each NCCS component computer to eliminate any dependence on external fileservers.

Tasks completed :

$\Longrightarrow$ **SETUP CODE DEVELOPMENT ENVIRONMENT ON APUS**

The development language and tools chosen are as defined in the *Requirements & Specifications.* They are :

1. GNU `ANSI C` as the main programming language,
2. IEEE Posix standard for operating calls,
3. `make` for code management,
4. `RCS` for revision control.

Other programming languages e.g. Fortran and/or Pascal, will only be permitted where code already exists in one of these languages and it is considered uneconomical to port the code in question to ANSI C. The following compilers will be used for the "foreign" languages :

- GNU's `g77` or `f2c` for Fortran,
- GNU's `p2c` for converting Pascal to C.

3.2 PC-STEER *(J. Quick)*

PC-STEER is one of the fundamental components of the NCCS. It refers to the hardware and software used to move and track the telescope. In its original form PC-STEER consisted of a digital and analog interface to the telescope encoders and drive, an HPIB interface to the observer's computer and a control program in Turbo-Pascal running under DOS.

For the NCCS the PC-STEER system underwent a number of major changes :

1. total redesign of hardware interface; the new interface addressed the problems of unstable powersupply, and noise on link between computer and hardware,
2. the PC interface card was replaced with a single Eagle PC-14B digital input/output interface card,
3. fast (100 Hz) synchronous data acquisition via V/F counters was added,
4. time code acquisition from the time standard using the NASA time code format was added. The time thus acquired is used to synchronise the time of PC-STEER and can also (via ntp) synchronise the rest of the control computers,
5. acquisition of certain weather parameters (air temperature, relative humidity and barometric pressure) was added for future atmospheric refractive correction modelling purposes.
6. acquisition of hyperbola tilt and focus position was added to support future computerised control of the hyperbola mechanism.
7. the control program (`control`) was ported from `Turbo-Pascal` running under `DOS` to `C` running under `Linux`,
8. the HPIB interface to the observing computer was replaced by an Ethernet connection and a device server talking TCP/IP protocol,
9. a time-of-day dependant hour angle and declination error correction fudge was implemented.

Tasks completed :

$\Longrightarrow$ **DESIGN NEW HARDWARE INTERFACE FOR PC-STEER**,

$\Longrightarrow$ **BUILD TWO COMPLETE HARDWARE INTERFACES**,

$\Longrightarrow$ **PORT `steer` TO LINUX AND C**,

$\Longrightarrow$ **REPLACE HPIB WITH TCP/IP AND DEVICE SERVER**,

$\Longrightarrow$ **LOCK PC-STEER CLOCK TO ACQUIRED NASA TIMECODE**.

Tasks still to be done :

⟹ **INTEGRATE PC-STEER WITH PROPOSED ALARM/LOG SERVERS**.

Outstanding issues :

- Although the new interface has been designed to handle most of the real-time requirements via hardware synchronous data-acquisition, it may be necessary to add real-time scheduling capabilities into the Linux kernel to ensure that critical software components e.g. the main control process, keep pace.
- The telescope error correction model needs a complete overhaul:
  1. The error correction needs to be split into static physical & dynamic atmospheric refraction components.
  2. The second declination encoder should be integrated to improve accuracy.
  3. Support should be added for handling continuous UT1 time instead of UTC with its associated leap second events.

3.3 Instrumentation Control Bus *(J.Quick)*

The main instrumentation bus which will be used to control digital input/output is the so-called **Serial Digital Input Output** bus (SDIO). This bus which relies on a serial link and a simple protocol was developed by J.Jonas of Rhodes University and is now being further developed and maintained by J.Quick at HartRAO. SDIO will be used on the NCCS through the BOCA 16 port serial interface. For details of the actual SDIO based devices see the relevant sections below.

The original SDIO concept has been adapted for use on the NCCS as follows :

1. The RS232 protocol was converted to RS422 in order to transmit over long distances noise free.
2. The topology of the SDIO links was converted from a daisy-chain loop to a tree so that failure of any one SDIO unit has the minimum effect on other equipment.
3. Each tree of SDIO units is serviced by a single monolithic device server capable of handling all possible SDIO based devices.

Tasks completed:

⟹ **RE-DESIGN THE SDIO TO USE A RS422 TREE TOPOLOGY**,

⟹ **BUILD SIX SDIO REMOTE UNITS AND THREE COMPUTER INTERFACES**,

⟹ **WRITE A GENERIC DEVICE SERVER FOR SDIO**.

Tasks still to be done :

⟹ **INTEGRATE SDIO WITH PROPOSED ALARM/LOG SERVERS**.

3.4 Noise Diodes *(J.Quick)*

The HP1000 controls the calibration Noise Diodes through a 3:8 multiplexed relay based driver circuit. In the NCCS independent control of up to 16 noise diodes is required. Furthermore all noise diodes must be capable of being continuously toggled by some external reference signal.

The following solution was proposed :

- Each noise diode should be driven by a circuit capable of being switched into one of three states: ON, OFF or MODULATING, producing a TTL level command output.
- Each noise diode circuit when MODULATING can be switched to track any one of up to eight external reference inputs under computer control.
- Each noise diode circuit should include a state detector driven by a feedback signal originating as close to the actual noise diode as possible.
- Up to eight of these circuits should be consolidated into one board with a common multiplexed computer interface capable of being driven by a SDIO unit, ie. supporting full remote configuration and control.
- Each board should interface through a noise diode specific adapter that produces the necessary drive voltages/currents and returns a direct TTL state indicator.

Tasks completed:

$\Longrightarrow$ **DESIGN NEW NOISE DIODE CONTROL CIRCUIT**,

$\Longrightarrow$ **ADD SUPPORT FOR NOISE DIODES INTO SDIO DEVICE SERVER**,

Tasks still to be done :

$\Longrightarrow$ **BUILD THREE INTEGRATED EIGHT-WAY NOISE DIODE CONTROLLERS**.

$\Longrightarrow$ **DESIGN AND BUILD CALIBRATION NOISE DIODE DRIVER INTERFACE**.

Note: The same circuitry can be used to control Dicke Switches as well merely by adapting the final driver interface suitably.

3.5 Time *(J.Quick)*

The NCCS will be synchronised to the time standard originating from a master clock driven by the 5MHz output of the station Hydrogen Maser Frequency standard. Synchronisation is achieved through a Network Time Server module (connected to the master clock IRIG-B timecode output) via the Network Time Protocol (NTP). PC-STEER receives the same UT time from the NASA BCD timecode output of the same master clock via a coaxial cable direct from the timing room. The main NCCS control computer and its backup are configured to synchronise primarily to the NTS with the PC-STEER system clock as backup. The two NCCS machines then function as secondary time servers to all other machines on both the control and main HartRAO networks.
Tasks completed :

$\Longrightarrow$ **PURCHASE, CONFIGURE AND INSTALL MASTER STATION CLOCK**,

$\Longrightarrow$ **PURCHASE, CONFIGURE AND INSTALL NETWORK TIME SERVER**,

$\Longrightarrow$ **EXPORT NASA TIMECODE LOCKED TIME FROM PC-STEER VIA NTP**,

$\Longrightarrow$ **INSTALL NTP SOFTWARE ON ALL CONTROL COMPUTERS**.

3.6 Alarms *(J. Quick)*

Alarms will be centrally managed and logged in the NCCS by a dedicated alarm device server. Any client can trigger an alarm by sending a command to the alarm server. On receiving the command alarm the alarm server will trigger the relevant hardware alarm (beeper), bring up an alarm message on an appropriate console and log the alarm via the system logging facility. The alarm server will also be able to poll a predefined list of device servers to check their state. If an equipment goes into error state or goes offline the alarm server will generate a suitable alarm. Clients will also be able register unique software watchdogs timers within the server which result in automatic alarms if they are not reset timeously. Implementing this alarm service as a server means it is easy to extend e.g. adding voice synthesis.

Tasks still to be done :

$\Longrightarrow$ **WRITE AN ALARM DEVICE SERVER**.

3.7 Logging *(J. Quick)*

Logging will be done in a similar unified manner in the NCCS via a **log server**. The log server will add a standard prefix to all log messages containing date, time, and originating program. All system commands will written to automatically log all parameters to the system logfile (via the log server). The system logfile will thereby keep a complete record of all observations and system parameters in a fixed format. The format should facilitate the easy production of time usage statistics to replace the current log sheets. Observing programs can also send additional informative log messages to the server to report on observing status e.g. minor problems etc.

Tasks still to be done :

$\Longrightarrow$ **WRITE A LOGGING DEVICE SERVER**.

3.8 Mark IV VLBI System Interface *(J. Quick)*

The Mark IV VLBI System has already been ported to Linux and is currently running version 9.13 of the field system (FS). Unfortunately the supported Linux kernel is rather old and is not compatible with the latest ESRF ELF-based Linux binary distribution. At present when the FS runs it has full control of the telescope, calibration noise diodes and weather data acquisition via direct HPIB and SDIO links. Under NCCS the direct link via HPIB going to the DOS PC-STEER will be replaced with a device server which will accept commands from Ethernet (via TCP/IP) and then send them to the Linux PC-STEER to control the telescope. Access to the weather parameters acquired by PC-STEER will also be made through the same device server. Control of the noise diodes will be done via a SDIO device server as described above. In addition the local software in the field system will have to be modified so that it communicates with the NCCS and requests and informs it of what it is doing (for alarms, logging and security purposes).

Tasks still to be done :

$\Longrightarrow$ **UPGRADE VLBI FIELD SYSTEM TO ELF-BASED LINUX VERSION 2.0+**,

$\Longrightarrow$ **PORT LOCAL FIELD SYSTEM SOFTWARE TO INTERFACE TO PC-STEER, NOISE DIODE, ALARM AND LOG DEVICE SERVERS**.

4 Phase 2 - General Observing Support

Although Phase 1 will provide a good test of the NCCS conceptual design, all other observing modes require further hardware support. With the failing HP1000s in mind clearly phase 2 should comprise the integration of all existing hardware into the NCCS system, together with the development of a minimal prototype of the user interface. Hopefully this would provide sufficient observing capacity to continue operations in the unlikely event of a complete failure of the present HP1000 based system.

4.1 Hardware Interfaces *(J. Quick)*

Many of the various hardware control interfaces are based on archaic mechanical components with rudimentary computer control and require extensive upgrades before they will be able to to function correctly under SDIO control. The primary functions currently envisaged as being under computer control in Phase 2 are :

- Full control of all Noise Diodes ie. including current Noise Adding Diodes,
- Full control of all Dicke Switches (utilising Noise Diode Controllers).
- IF input signal select to Mark I Pulsar Timer System,
- Front-end RF (1612/1665/VLBI/1720) filter selection at L-band,
- Left and Right IF input signal selects to Spectroscopy Receiver Systems,
- Back-end internal IF filter selects within General Purpose Radiometers,

Tasks still to be done :

$\Longrightarrow$ **DESIGN AND BUILD NOISE ADDING DIODE DRIVER INTERFACE**,

$\Longrightarrow$ **DESIGN AND BUILD DICKE SWITCH DRIVER INTERFACE**,

$\Longrightarrow$ **DESIGN AND BUILD SWITCHING RELAY DRIVER INTERFACE**,

$\Longrightarrow$ **WRITE DEVICE CLASSES FOR SELECTS TO BE BUILT INTO SDIO SERVER**.

4.2 Radiometer Data Acquisition *(J. Quick)*

Our simplest data acquisition task takes the form of monitoring the radiometric output of various receivers as a function of telescope orientation and of time. Such fast data aquisition is currently done using the *Fluke* 8840 digital voltmeters. Position and time stamping are handled in software, relying on the real time nature of the HP1000 operating system for accuracy. The NCCS plans to replace the Flukes which are on their last legs with high resolution V/F converters connected to custom counter cards built into the new PC-STEER hardware interface. Synchronous acquisition of counter readings, NASA timecode and telescope position encoder readouts is handled by means of a common hardware strobe. All radiometers will be equipped with dedicated V/Fs with each counter card being switched between up to eight different V/F outputs under computer control. Several binary divider outputs based on the hardware strobe are available for the switching of Dicke Switches and Noise Adding Diodes to facilitate future synchronous digital radiometer modes.

Tasks completed :

- ⟹ **SELECT A V/F CHIP AND DESIGN AND BUILD SUITABLE CONVERTERS**,
- ⟹ **DESIGN AND BUILD COUNTER CARDS FOR PC-STEER HARDWARE INTERFACE**,
- ⟹ **ADD SYNCHRONOUS ACQUISTION OF COUNTER READINGS INTO PC-STEER**.

Tasks still to be done :

- ⟹ **QUALIFY V/F CONVERTERS FOR USE ON EXISTING RADIOMETERS**,
- ⟹ **INTEGRATE A SUITABLE RADIOMETER DEVICE SERVER INTO PC-STEER**.

4.3 Auto-Correlation Spectrometer *(M.Gaylard & J.Quick)*

Another data acquisition task involves using an Auto-Correlation Spectrometer to determine the output of a celestial source as a function of frequency. At present it is unclear just what form the NCCS correlator will take. The current correlator is very limited in capabilities, and there are no spares for certain critical components. The original NCCS Design envisaged the building of a second correlator with an upgraded computer interface from the present pool of spares along with reworked versions of boards as and where necessary. The working correlator was to be left untouched to avoid reliability issues. The NCCS proposed to interface this new Correlation Spectrometer to a dedicated PC running *Linux* to be called PC-COREL. PC-COREL would operate in an analogous manner to PC-STEER, using the same Eagle PC-14B interface card to communicate with the correlator. The NCCS would interact with PC-COREL through the standard ESRF device mechanism using NFS RPC's over Ethernet.
Tasks involved would be :

- ⟹ **COMPLETE THE CONSTRUCTION OF THE NEW CORRELATOR**,
- ⟹ **SETUP SUITABLE PC-COREL LINUX SYSTEM**,
- ⟹ **FINALISING THE INTERFACE CABLE BETWEEN PC-COREL AND CORRELATOR**,
- ⟹ **WRITE SIMPLE CORRELATOR DEVICE DRIVER BASED ON PC-STEER INTERFACE DRIVER**.
- ⟹ **WRITE CORRELATOR DEVICE SERVER TO INTERFACE TO THE DRIVER**.

Recently HartRAO acquired an Australian Long Baseline Array (LBA) Data Acquistion System (DAS). Although this was acquired to support the VLBI Space Observatory Program initiative by producing 2-bit sampled data for S2 recording, it also produces direct correlator outputs suitable for the Australian LBA Multibeam Correlator. An alternative NCCS correlator solution would be to purchase a Multibeam Correlator for HartRAO. Clearly this option needs to be urgently investigated as it directly impacts on Phase 2 manpower requirements vis-a-vis the tasks outlined above.

4.4 Pulsar Timer *(C.Flanagan??)*

Control of the Mark I Pulsar Timer is to be done directly from the NCCS via HPIB using a device server custom written for the purpose. The device server will accept commands from the network and convert them to HPIB, implementing all HPIB commands of the Mark I Pulsar Timer.

What is not yet clear is how the Pulsar Timer internal clock time will be setup. Time under Linux does not work in the same way that time worked on the HP1000s and it may be necessary to modify the Pulsar Timer to cope with this.

Tasks still to be done :

$\Longrightarrow$ **INVESTIGATE CLOCK SETTING DETAILS OF MARK I PULSAR TIMER**,

$\Longrightarrow$ **WRITE A DEVICE SERVER FOR THE MARK I PULSAR TIMER**,

4.5 Frequency Synthesisers *(M.Gaylard)*

The two frequency synthesisers (HP8862 and HP3336) will be controlled directly from the NCCS via HPIB. A generic frequency synthesiser device server will be written to control each of the synthesisers. The device server(s) will accept a standard set of commands for setting up and reading a frequency synthesiser from the network and convert them to the necessary HPIB equivalents.

Tasks still to be done :

$\Longrightarrow$ **WRITE DEVICE SERVERS FOR HP8862 AND HP3336**

4.6 Observing - System Libraries *(J.Quick)*

To the astronomers the observing is one of the most visible and important parts of the system. The NCCS proposes to change fundamentally the way observing is done at HartRAO. On the old HP1000 control system observing programs were written in Fortran. Setting up the system was done directly by the observing program at a very low level. Although some common subroutines and/or examples for certain procedures existed observing programs often needed to know about the system at a very low-level and much code duplication exists. Modification of multiple programs to reflect any new system change takes a long time and is highly inefficient.

The NCCS proposes to implement all system aspects up to a very high level as atomic "system commands". These actions will be directly callable either from an interactive shell, or directly from observing programs written in an interpreted high level observing language, or in C. Accessing of system parameters will only be possible via these system commands. This will ensure there is only one implementation of each system command functionality. Observing programs should no longer need to worry about low-level system aspects anymore (e.g. device details like addresses, data formats etc.). Writing programs in an interpreted language also ensures that modifications can be done rapidly *on-the-fly*. All these system commands will be implemented only once in C as an ELF dynamically loadable library (DLL). This will ensure only that the most current version of each command is in use.

Tasks still to be done :

$\Longrightarrow$ **DEFINE A SUITABLE SET OF BASIC SYSTEM COMMANDS TO BE IMPLEMENTED**,

⟹ **CONSTRUCT A DYNAMICALLY CONFIGURABLE SYSTEM SETUP RESOURCE DATABASE**,

⟹ **IMPLEMENT ALL COMMANDS IN C AND GENERATE OBSERVING SHARED LIBRARY**.

4.7 User Interface *(J.Quick)*

The user interface will be based on a **library of C routines** as defined above, together with a **high-level observing command language** (Tcl is the most likely candidate), and a **graphical user interface** (Tk is the most likely candidate here). Although Tk/Tcl is well-suited to this task there are some reservations about it as a language[1]. For the present time however we assume we will use Tk/Tcl. The C interface will allow programmers to use a typed language and go down to a low-level when and if they need to. The system commands will also be encapsulated as simple commands available in Tcl for writing observing shell scripts or doing interactive observing and/or programming. Tcl will be extended by these commands to implement an "observer's shell" (`obssh`) which will be the main command line interface for simple observing. Astronomers will use `obssh` and `Tk` to create graphical user interface for observing programs. In the same manner system programmers will use `obssh` and `Tk` to create graphical user interfaces for test programs and system utilities e.g. a generic program for setting up radiometers, a manual drive program etc.

Tasks still to be done :

⟹ **IMPLEMENT THE MANDATORY SYSTEM COMMANDS IN C AND AS EXTENSIONS TO TCL**,

⟹ **BUILD GRAPHICAL USER INTERFACES FOR BASIC SYSTEM CONFIGURATION TASKS**.

5 Phase 3 - System Integration

5.1 Scheduler & Security *(J.Quick)*

The scheduler will play a central role in the NCCS. Its main role is to manage the observing system's resources in an optimum manner. Three different classes of observing programs have been defined in the *Requirements & Specifications* document. The so-called Master, Routine and Background Observing Programs (MOP, ROP and BOP) classes. The priority of each of these classes is well-defined in the *Requirements & Specifications* document. The scheduler has to make sure that only one observing program has control of critical resources (e.g. PC-STEER and the telescope) at a time and according to the priorities of each class.

The scheduler proposes to implement the priority in a device server. The fundamental unit of the scheduler is a **scan**. Different types of observation can have different definitions for a scan. Mapping could define a scan to be a real scan of the sky, spectral line observers would define a scan to be a single integration for example. All clients (observing programs) who want to do a scan or reserve a resource for their exclusive use e.g. a radiometer, must request authorisation from the scheduler. If the scheduler grants authorisation it will return a unique ID number to the client and inform the device server managing the resource which ID is current.

[1](cf. *http://icemcfd.com/tcl/comparison.html*)

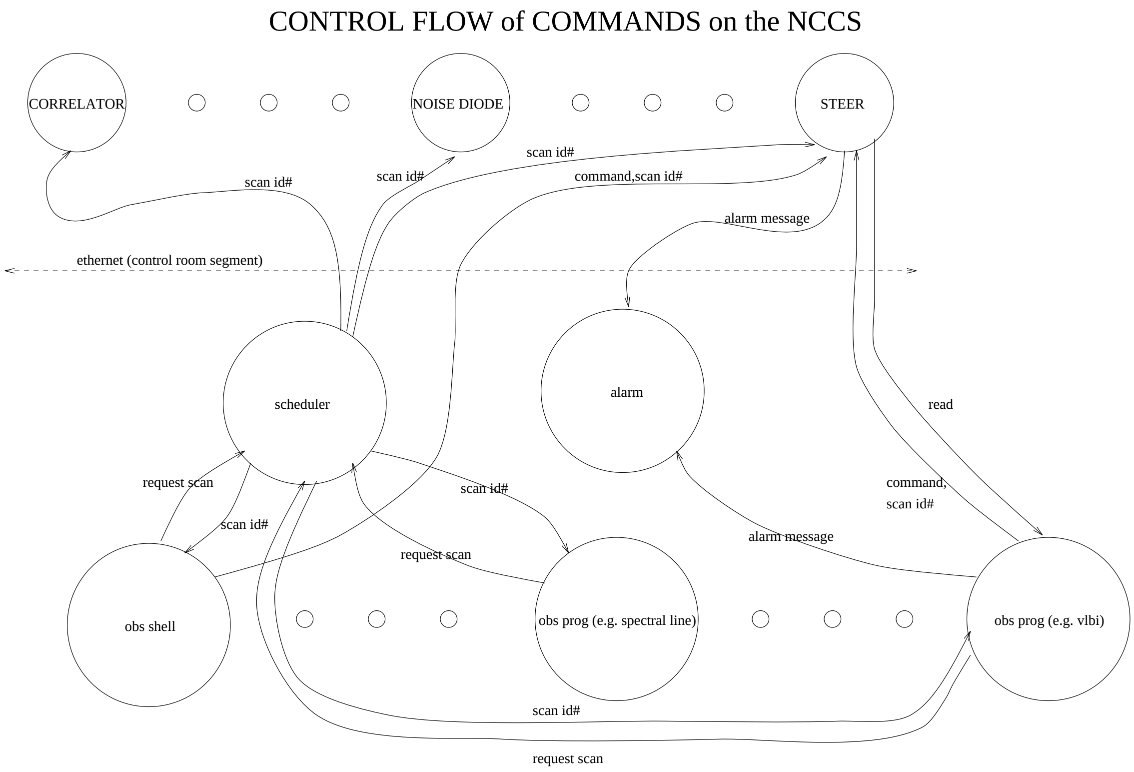


Figure 2: Control Flow of Commands via Scheduler on the NCCS

The client can then use the resource until it decides to free it or the scheduler gives priority to another client. Clients can ask to be queued and be woken up at a later time. Queued clients are kept in batch file and are woken up using the Unix `cron` mechanism. This way in the event of a system crash or restart queued clients are automatically rescheduled. The data flow for the scheduler is presented in *figure 2*.

Tasks still to be done :

⟹ **DEVELOP CUSTOM HARTRAO SCHEDULER AND RELATED SYSTEM COMMANDS**,

⟹ **INTEGRATE SCHEDULER BASED SECURITY INTO ALL DEVICE SERVERS**.

5.2 Common Data Format

All data should be stored in a common locally-developed FITS compatible format still to be defined.

Tasks still to be done :

⟹ **DEFINE AND IMPLEMENT A FITS COMPATIBLE DATA FORMAT**

5.3 Observing Programs *(Astronomers)*

The NCCS has to support a large number of different kinds of observing programs. The programs to be supported can be found in the *Requirements & Specifications* document. They will be implemented by the respective astronomers once the NCCS is designed and the observing language is defined. Most of the programs should be fairly easy to port once the observing algorithms have been squeezed out of the astronomers and written done in a pseudo-code. Continuum observing will however require considerable restructuring and rewriting in order to unify all the different programs which exist.

Tasks still to be done :

$\Longrightarrow$ **GENERATE ALL POSSIBLE OBSERVING ALGORITHMS AS PSEUDO-CODE**,

$\Longrightarrow$ **IMPLEMENT SUITABLE OBSERVING PROGRAMS IN HIGH LEVEL LANGUAGE AND/OR C**.

6 Phase 4 and beyond

6.1 Antenna Pointing System Upgrade *(J.Quick)*

The current optical encoder couplings are known to have serious performance deficiencies. A suitable replacment coupling has been identified and a full new set of couplings are currently on order. As part of the integration of these couplings, it was felt that the current antenna pointing system was sorely in need of the following upgrades

- the use of a parameterised error map model, including explicit weather dependent refraction corrections,
- the use of two independent Declination optical encoders for redundancy and improved accuracy,
- the parametrisation of ambient temperature dependant errors.

6.2 Technical Monitoring System *(M.Ainley)*

The Technical Monitoring System is not part of the NCCS *per se*. It will be bought and managed independantly in a first stage. As soon as the TMS is available the NCCS must be able to interface to it to read alarms and equipment status. Interfacing will be done via NFS files or via a specially developed driver. Both of these solutions are viable and the exact solution will be decided once the TMS has been bought and installed.

7 Other Issues

7.1 Testing

Testing of the NCCS must be done in parallel with normal operations in order not to interfere with observing. The best way of achieving this is to simulate all hardware components currently in use by the HP1000 control system and test the new software in simulation mode. The PC-STEER system includes all the code necessary for simulating the telescope and will be used to test the PC-STEER device server, new observing programs, the scheduler, etc. New software which is being developed for the NCCS e.g. PC-COREL, pulsar timer and other hardware device servers, must have a software simulation mode built in where possible so that it is possible to test some of their functionality without necessarily having access to the actual hardware. All such simulation modes must only be operable where the appropriate software has been manually started from an interactive console.

7.1.1 Disks and Backups

The NCCS has a 1 GByte SCSI hard disk. This is sufficient for doing data collection and observing over short periods of approximately weeks. Over longer periods data has to be moved off the `nccs` to a backup medium and/or to user's local machines. System software and log files will be backed up on a regular basis over the network onto DAT.

8 NCCS Physcial Configuration *(J. Quick)*

The NCCS has its own independant Ethernet segment dedicated for doing control. This segment is isolated from the Internet via the main control computer `nccs` & its backup `nccs2` which both have two Ethernet cards and act as IP bridge routers between the control network and the local HartRAO network (and thereby to the Internet). Both computers also provide secondary nameserver service so that the control network can run completely disconnected from the main HartRAO network if necessary. Note that the IP routing only works locally, the control network is not directly reachable from any external site. This section deals with the individual computers that will make up the control network, detailing their hardware and the hardware and software components that they service.

8.1 Main (`nccs`) & Backup (`nccs2`) Control Computers

These two computer systems have been bought and setup to be as similar as possible with regard to hardware and software. They are hardware compatible with the FS standard configuration defined by the NASA Field System Group.

Hardware :

Motherboard: Pentium 100 MHz, 16MB RAM,

Disks: twin removeable 1 GByte SCSI disks,

Interfaces: AT-GPIB interface, 16 port BOCA serial interface.

Attached Hardware :

- Mark I Pulsar Timer (HPIB),
- Frequency Synthesisers (HPIB),
- SDIO instrumentation bus (Serial).

Software Services :

- Secondary HartRAO Domain Nameserver (DNS),
- Secondary Time Standard (NTP).

8.2 VLBI Field System Computer (`vlbi`)

This is the original FS standard configuration.

Hardware :

Motherboard: 80486 50 MHz, 16MB RAM,

Disks: twin removeable 200 MByte SCSI disks,

Interfaces: AT-GPIB interface, 8 port DIGI serial interface.

Attached Hardware :

- Mark III Data Acquisition Terminal (Serial),
- LBA (S2) Data Acquisition System (Serial).

Software Services : (none).

8.3 PC-STEER Computer (`steer`)

Hardware :

Motherboard: 80486 33 MHz, 16MB RAM,

Disks: internal 810 MByte IDE disk,

Interfaces: Eagle PC-14B interface.

Attached Hardware :

- PC-STEER Servo Interface Cage (PC-14B).

Software Services :

- Backup Primary Time Standard (NTP).

8.4 PC-COREL Computer (`correl`)

Hardware :

Motherboard: 80386 33 MHz, 8MB RAM,

Disks: internal 810 MByte IDE disk,

Interfaces: Eagle PC-14B interface.

Attached Hardware :

- HartRAO Auto-Correlation Spectrometer (PC-14B).

Software Services : (none).

8.5 Network Time Server (`tick`)

Hardware :

Motherboard: Unknown,

Disks: (none),

Interfaces: IRIG-B Timecode Receiver.

Attached Hardware :

- HartRAO Main Station Clock (IRIG-B).

Software Services :

- Primary Time Standard (NTP).