United Kingdom Infra Red Telescope

Joint Astronomy Centre

Royal Observatory Edinburgh

Observing Sequencer: Design Document

Alan Bridger

orac013-seq2 Version: 01 **Original: 18 May 1998 Modified: 17 June 1998**

This document sets out a design for the Observing Sequencer part of the ORAC system. This document is a significantly modified replacement for orac007-seqd and supersedes that document.

1.0 Introduction

1.1 Purpose

This document presents the overall design of the ORAC Observing Sequencer. It is intended to present a design of the system that will, as far as is possible, meet the requirements set out in [O-5].

1.2 Scope

This document presents a design of the Observing Sequencer that does not specify the likely language or environment to be used, but does attempt to specify the functionality and interfaces that are required in some detail. It should be noted that a prototype sequencer written in the Drama environment by Alan Pickup already meets most of the requirements and also that the Todd system, written in Java by Dennis Kelly for the JCMT also looks as though it would be close to meeting the requirements.

1.3 Definitions, acronyms, abbreviations and references

These may be found in "*ORAC: Glossary and Reference List*" (orac000-gar). In addition the document uses the term "Sequencer" commonly in place of the full "Observing Sequencer" and there are occasional contractions (e.g. "OD") which I hope are obvious in context. These saved my typing wrist a little.

1.4 Overview

Section 2.0 of this document provides an overview of the position of the Observing Sequencer with respect to the rest of ORAC and other UKIRT systems and gives a short background to the project.

Section 3.0 then outlines the context of the system.

Section 4.0 outlines the design of the system, highlighting the interfaces.

Section 5.0 addresses the issue of backwards compatibility, i.e. the fact that the sequencer must be capable of controlling existing instruments.

Section 6.0 describes the engineering functionality that the sequencer must provide.

Section 7.0 describes how the sequencer should cope with errors or failures in the systems it controls, or within itself.

Issues that are still to be resolved are listed in Section 9.0.

Appendix I presents an example of an Observation Definition.

2.0 System Overview

2.1 Context

The *Observing Sequencer* is the central component in the UKIRT Observatory Control System outlined in [O-12]. It is responsible for reading an *Observation Definition Sequence* and then executing this as a Sequence of commands, either built-in or by commanding other observatory control systems - the instrument control system, the telescope control system and the data handling system and also any facility systems like the FP and IRPOL systems. It is also responsible for sequencing configuration of these systems with the data taking according to the *Observing Sequence* part of the *Observation Definition*.

The *Observing Sequencer* relates to the *UKIRT Preparation System* and the O*bserving Database* only through the *Scheduler* or *Chooser.*

Despite this central and important role, the overall ORAC OCS design presented in [O-12] indicates that the *Observing Sequencer* is, however, a relatively simple task that does not know very much about the systems it is controlling. It should thus be possible to make it very robust.

2.2 Background

The *Observing Sequencer* is the clear descendent of the EXEC handling part of the AIM task in the current CGS4 and IRCAM3 systems, performing essentially the same role but more clearly devolved from the instrument control and sequencing which in the older systems was also partly executed by AIM.

3.0 System Context

Figure 1 shows the context diagram for the sequencer along with the descriptions of the items and data flows.

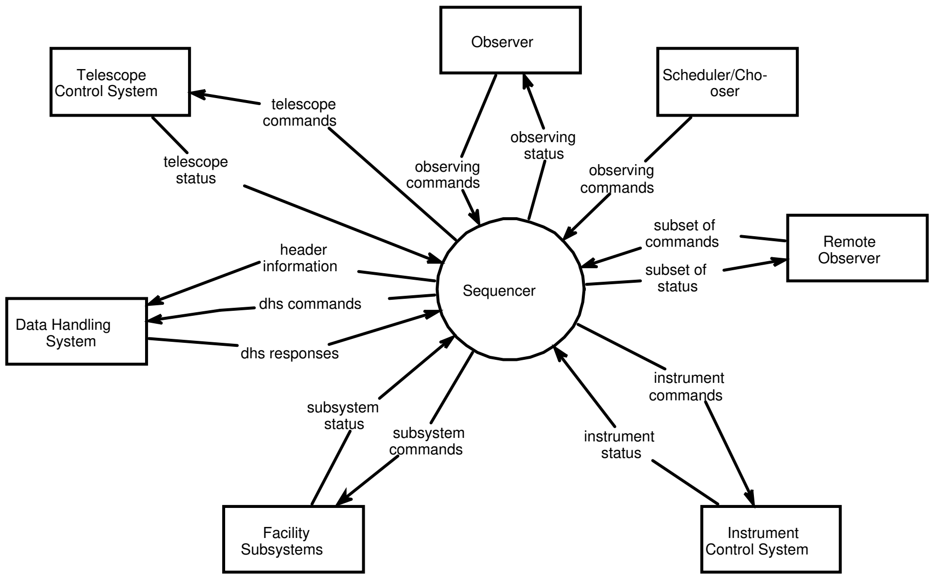

FIGURE 1. The Observing Sequencer Context Diagram

In this diagram the external entities are:

Observer. This represents the Observer, or Engineer, issuing commands to the sequencer. (Are engineering commands sufficiently different to warrant a separation?)

Telescope Control System. This is the existing UKIRT telescope control system.

Remote Observer. This represents a potential remote observer allowed to send commands to the sequencer. The commands sent and the status received may be a subset of the full range.

Instrument Control System. This is the system that has overall control of the instrument - it can configure it and instruct it to acquire data. The instrument includes any array controller and data acquisition system.

Facility Subsystems. This represents other facility subsystems such as the FP or IRPOL systems.

Scheduler/Chooser. This is the task (which will be the Chooser in the first instance, the Scheduler later) that will instruct the Sequencer as to which Observation Definition is to be performed next.

Data Handling System. This represents the system responsible for storing data and also for performing other services for the instrument control system (quick-look, provision of bad pixel masks etc.). The latter services probably have no relevance to the sequencer.

The major data flows on are:

observing commands. This represents the commands to observe that come from the Observer or Engineer. Commands to interact with the observation may also be allowed. There may be more complex engineering commands.

observing status. This is a flow of status information - responses to **observing commands** or displays of the observing status.

subset of commands. This is a subset of **observing commands** suitable for a remote observer.

subset of status. This is a subset of **observing status** suitable for a remote observer.

instrument configuration. This is how the component of an Observation Definition that represents an instrument configuration gets to the instrument system. This flow may be one of three things: A name of such a configuration; a data structure ("blob" of information) containing the configuration; or a set of atomic parameters designed to achieve the configuration. The initial implementation will use a name of a configuration. Further implementations will depend on demand.

instrument commands. This represents the commands sent to the instrument control system instructing it to configure itself or to acquire data or to perform some other function.

instrument status. This is a flow of status information - usually responses to **instrument commands**.

subsystem commands. These are the commands sent to facility subsystems.

subsystem status. This is status information sent back by facility subsystems.

observation definitions. This is the flow of **Observation Definitions** from the Scheduler/Chooser, probably via a file.

dhs commands. This is the flow of commands sent to the data handling system by the sequencer informing it of changes to the data storage situation, possibly including telling it to expect certain data frames.

dhs responses. A flow of responses to the **dhs commands**.

header information. This is header information known or acquired by the Sequencer being made available to the Data Handling System for storage along with the associated data set.

telescope commands. This represents the commands sent to the telescope control system instructing it to configure itself or to acquire a target or to perform some other function.

telescope status. This is a flow of status information - usually responses to **telescope commands**.

target description. This represents the *target description* component of an *Observation Definition*. As with an instrument configuration it may be one of three things: A name of such a description; a data structure ("blob" of information) containing the description; or a set of atomic parameters designed to achieve the description.

4.0 System Design

One starting point for the system is to use EXECs, similar to their use in the AIM task, to perform the sequencing of the operations required to take data. In order to fully meet the requirements the following enhancements would need to be made:

1. Extend the EXEC dictionary to contain actions not currently included (e.g. commands to the TCS regarding guide star to be used, focus position, etc.)
2. Extend the EXEC parser to allow token substitution in EXECs. This will allow some standard sequences that change occasionally (e.g. how far along a slit to offset) to be more fully standardised.
3. Extend the EXEC interpreter to allow some sequence actions to be executed in parallel - implies either splitting some into initiation and wait for completion or providing knowledge (in the dictionary) of which commands may be parallalised and which may not.
4. Allow for concurrently executing EXEC threads. This is different from 3. in that this is intended to allow data to be taken with two instruments simultaneously. One intent is to allow calibration frames to be taken with one instrument whilst target frames are taken with another, though other applications can be envisaged. Clearly one thread will have restrictions placed upon it as to what it is allowed to do.
5. One implication of the above enhancement is that it will be necessary for the Sequencer to know when the telescope is “in position” and ready to observe following a telescope movement.
6. Extend the EXEC dictionary to include a standardised description of how to perform the required action, e.g. technology to use (ADAM message, DRAMA message, EPICS channel access), target (task name, record name), and command, etc. This might require sections in the dictionary for each system controlled. The idea behind this extension is that it will allow the system to be general and will go some way towards meeting the goal that existing instruments can be put into the new system.
7. New commands to “post” header information for the DHS to collect.
8. The ability to be able to ignore certain “commands” in the EXEC language as they will in fact be flags intended for the user-interface.

Despite these additions, and thus the apparent increase in complexity, in fact this Sequencer will be a relatively simple task. The reason, as outlined in [O-12], is that most of the intelligence about observing is removed and placed into standard sequences. Thus this sequencer becomes a relatively simple task, able to issue messages to other systems, receive simple responses and perform some simple internal tasks.

A prototype Sequencer using EXECs and many of the enhancements outlined above is under development by Alan Pickup for Michelle testing.

Another approach that could be taken here is to use some of the components that form the Todd system ([X-13],[X-14]). The Todd is a JCMT OCS system under development (in Java) by Dennis Kelly. It is required to drive JCMT observing at a slightly different level, but many of the concepts are similar, and the Todd has Java Classes that handle Drama and ADAM communications and command sequencing. Obviously work would be required to use the Todd as a Sequencer for UKIRT.

From its position in a central role in the observing process and the short discussions above it is clear that interfaces are an important part of this task. Later sections of this document summarise the interfaces to each system from the Sequencer. However, each of these interfaces should be implemented using the same technology in the Sequencer itself, making its design simple.

4.1 Sequencer Functionality

The main functions of the Sequencer are as follows:

1. It must be able to read an ASCII file which defines its internal dictionary.
2. It must be able to read an ASCII file which defines the technology to be used to issue commands to each system it may need to control.
3. It must be able to read an ASCII file for each system it is to control listing and defining the commands available for that system.
4. It must be able to read an ASCII file containing Sequence commands to issue.
5. It must be able to execute these commands in sequence, and if allowed by the command definition, in parallel.
6. Each sequence command shall result in only one action being performed by the sequencer, either internally or in the guise of a command sent to another task. ("One action" in the latter case includes the receipt of one or more messages up to the completion of the action in the remote task.)
7. Sequences may "call" other sequences. The Sequencer must be able to handle this transparently.
8. Must respond to the directives STOP, ABORT, PAUSE and CONTINUE in the manner defined in Table 1.
9. Must interface to the Sequencer User Interface in such a way as to allow that system to satisfy its requirements. (This is deliberately fuzzy - the two systems must be developed to work together.) The Sequencer User-interface will be part of the Chooser [O-14].

In addition there are a number of functions which are to be performed by the Sequencer itself, rather than by other tasks. These are specified in section Section 4.10.

4.2 Sequencer Command Interface

This is the command interface that the sequencer presents to the outside world, and in particular to its user-interface and to the Chooser or Scheduler. The functions provided are outlined in Table 1. Some of the names are probably available for debate.

4.3 Interface to the Observer

This is the user-interface of the Observing Sequencer. It will allow access to the functions outlined in Table 1 and in addition will provide a feedback to the user of the status of the sequencer. This will include:

- Programme information: Observers, TAC code, etc.
- Observation Definition Information: Sequence name, targets, etc.
- Observation Details: Observation number/data label, time to completion, etc.
- Observation Sequence feedback: View of the observing sequence in progress
- An optional verbose mode of the commands sent to subsystems.

This user interface will be part of the Choosers' user-interface and more detailed requirements may be found in [O-14].

Name	**Parameters**	**Action**	**Notes**
Startup	Observer names/ identification, TAC Number	Initialises the system, opens observing database with default access to the relevant ODs.	Must be done before any other actions. Leaves the system in “Stopped”.
Load Sequence	Sequence name	Loads a sequence into the system, does not initiate it.	System remains in the state it was previously in. Only one sequence at a time may be loaded. In the case where a sequence is already in progress this will be rejected with an error.
Run		Starts the sequencer running, executing the observation.	State changes to “Running”. If there are no queue items the state remains at “Stopped”. Not available if “Paused”.
Pause		Pauses the sequencer as soon as possible, alerts other systems of the pause?? (their behaviour is up to them.) Highlights where the pause will occur in the feedback.	Because of the unknown behaviour of other systems the exact effect of this cannot be stated, but it will stop the sequence running after the current item is complete. State changes to “Paused”.
Continue		Restarts the sequencer running, alerts other systems of the continue	The effect on the sequencer is almost identical to “Run”, but this is subtly different from “Run” as other systems may be in a “Paused” state. Only available when sequencer is “Paused”. State changes to “Running”.
Stop_ASAP	GotoEnd	Causes the sequencer to halt after the current item is complete. Alerts other systems of the stop. Highlights where the stop will occur in the feedback. Following the Stop will position the sequencer at the next “End Point” if GotoEnd is TRUE.	Only available when sequencer is “Running” or “Paused”. Overrides the effect of a “Pause”. If “GotoEnd was FALSE, state changes to “Stopped”. If “GotoEnd was TRUE, state changes to “Paused”.

TABLE 1. The Sequencer Command Interface

Stop_At_Break	GotoEnd	Causes the sequencer to halt at the next specified “break point”. Alerts other systems of the stop. Highlights where the stop will occur in the feedback. Following the Stop will position the sequencer at the next “End Point” if GotoEnd is TRUE.	Only available when sequencer is “Running” or “Paused”. Overrides the effect of a “Pause”. If “GotoEnd” was FALSE, state changes to “Stopped”. If “GotoEnd” was TRUE, state changes to “Paused”.
Abort		Prompt user to confirm. If confirmed then: Causes the sequencer to halt as soon as possible. Alerts other systems of the abort.	Only available when sequencer is “Running” or “Paused”. Overrides the effect of a “Pause”. State changes to “Stopped”. Prompt user to: Abort just this item; or skip the rest of the observation definition, or restart the observation definition, or restart this item.
Options		Bring up “options” menu	This give access to various options that are mostly TBD. E.g. this section will allow the setting of flags to control communication with systems.

TABLE 1. The Sequencer Command Interface

4.4 Interface to the Remote Observer

This is the interface to the system as presented to a remote observer. This may be the same as the interface to the Observer, but might be a restricted subset, or a subtly different interface. The details of this will be decided late in the project when more is understood about the requirements.

4.5 Interface to the Scheduler/Chooser

The Scheduler/Chooser represents the system that will form the major user-interface to observing. The name is indicative of the fact that in the first instance it will be a relatively simple tool which will allow the user to choose which Observation to run next. A later version will aid in the choice and perhaps perform automatic scheduling of the Observations.

Either the *Scheduler* or *Chooser* will also be responsible for translating an Observation Definition as held in the Observing Database into the components required by each other major part of the system. One of these parts will be the Observing Sequencer and it will receive from the *Chooser/Scheduler* a *Sequence* which will contain all of the information required for the systems to be able to take the observation in question. This *Sequence* will most likely be in the form of a temporary disk file.

4.6 Interface to the other systems

The Telescope Control System, Instrument Control System, Data Handling System and the Facility Subsystems shown in Figure 1 form the other primary systems in the UKIRT Control System. The command interface to each of them is presented later, however it is also necessary to define the technology used.The interface to each of them is a messaging interface. The design presented here allows for the use of other technologies, though it is most likely that Drama message system will be used, perhaps in all cases. For those systems implemented using ADAM or EPICS the use of a translation task (ADITS and DES) will make them appear like a Drama task to the sequencer.

The information on how to communicate to a system will be table driven - the sequencer will read a file which will contain information similar to that presented in table Table 2.

The existing instruments are likely to remain as VMS-ADAM tasks and new instruments (Michelle, UIST) are currently planning to have a Drama-based instrument sequencer (running on Solaris), even though much of the instrument software has an EPICS interface. In addition it is plausible that new instruments might just have EPICS interfaces.

Instrument Name	Technology	"Task" Name	Implementation
CGS4	ADAM	irtcon.ukirt.jach.hawaii.edu!!cgs4	Issue an ADAM "obey" to the task with the relevant command name and parameter values
IRCAM3	ADAM	irtcon.ukirt.jach.hawaii.edu!!ircam3	Issue an ADAM "obey" to the task with the relevant command name and parameter values
MICHELLE	DRAMA	Michelle	Issue a "ditsobey" to the task with the command name and parameters
IMAGINE	EPICS	img:	Use channel access to put values into database records, whose names are constructed from the information the instrument has provided.

TABLE 2. System Interface technologies

The possible EPICS interface referred to here is more problematic and is not likely to be implemented in the first release of the sequencer, in fact probably not unless there is a demand. However, there would seem to be nothing preventing the idea.

4.7 The Sequencer Dictionary

This is a description of the sequence command set understood by the sequencer. It defines the name of the command, along with the parameters expected, the system it refers to (i.e. the system a command will be issued to) and whether or not the command may be executed in parallel with other commands. Some additional notes are useful:

- Any command for which parallel operation is allowed may be executed in parallel with any other such command.

Command	Sent Parameters	Returned Parameters	System Commanded	Parallel Operation?
INITINST			ICS	YES
DATUM	DeviceName		ICS	YES
LOADCONFIG	ConfigurationName		ICS	YES
SET	ConfigurationType		ICS	YES
TAKEFRAME	FrameTag, ObservationType		ICS	NO
MOVIE			ICS	NO
INITDHS	InstrumentName		DHS	NO
STARTFRAME	Observation Type	FrameTag	DHS	NO
STARTGROUP		GrpNum	DHS	NO
NEXTSEQ		GrpSeq	DHS	NO
ENDGROUP			DHS	NO
ENDFRAME	FrameTag		DHS	NO
TAKEHEADER	Stage		DHS	NO
MOVIE			ICS	NO
TARGETDESC	TargetDescriptionName		TCS	YES
SLEW			TCS	YES
NOD	BeamName, GuideState		TCS	YES
OFFSET	RAOffset, DECOffset		TCS	YES
GUIDE	GuideState		TCS	YES
ONSOURCE			TCS	NO
FOFFSET	FocusOffset		TCS	YES
TOFFSET	RAOffset, DECOffset		TCS	YES
XOFFSET	RAOffset, DECOffset		TCS	YES
ISU2	Position, Offset		TCS	YES
IRPOL	IrpolPosition		TCS	YES
DATUMPOL			IRPOL	YES
POL	PositionNumber		IRPOL	YES
POLDEG	Degrees		IRPOL	YES
FPX	XPosition		FP	YES
FPY	YPosition		FP	YES
FPZ	ZPosition		FP	YES

TABLE 3. The Sequence Command Dictionary

Command	Sent Parameters	Returned Parameters	System Commanded	Parallel Operation?
DRRECIPE	RecipeName		Internal	NO
SEQUENCE	SequenceName		Internal	NO
REPEAT	NumTimes, SequenceName		Internal	NO
BREAK			Internal	NO
BREAKPOINT			Internal	NO
WAIT	TimeSeconds		Internal	NO

TABLE 3. The Sequence Command Dictionary

- Any command issued to the "ICS" is expected to be available for any Instrument Control System. However, the sequencer will only be initialised for using one at any time, so there is no need to distinguish which system is intended.
- The word "internal" under the heading "System Commanded" indicates that the Sequencer handles the command itself.
- The dictionary is not (and can never be) complete. New commands can be added straightforwardly. This allows easy access to instrument engineering functionality, and also leaves room for new observing concepts.

4.8 Parallel Commands

A requirement of the sequencer is that, for efficiency, such commands as can be should be executed in parallel. Clearly commands that take data cannot be executed in parallel (in most cases) and some commands take a short time (e.g. NAME). It would seem that the commands that cause a reconfiguration of a system (e.g. SET, SLEW, DATUM, FPZ, POL) can be executed in parallel with each other. The sequencer should be able to implement such parallelism fairly straightforwardly, as long as the dictionaries describing the commands flag which of them can be executed in parallel. Table 3 indicates the commands that can be executed in parallel. It can be seen that most commands can be executed concurrently, with the except of those that take data. However, there is an additional rule not mentioned in the table:

- Only one command may be issued to any single subsystem at a time

Though it may technically be possible in some cases to break this rule it would not in most situations make much sense, and it also makes a clear, simple, definition.

4.9 Sequencer Functions

This section lists the functions that the sequencer must be capable of in order to meet the requirements set out in [O-5] and the design summarised in Section 4.0.

4.9.1 Token substitution

Individual Sequencer commands may have parameters and it must be possible to use “tokens” (variables) instead of actual values, and for these tokens to be replaced at execution time.

4.9.2 Parameter gets/puts

Some Sequencer commands may require the return of simple parameter information as the result of execution in a remote task. It must be possible to “get” this return value and also then to use it elsewhere as a “put” to another task. A possible illustration of use is the example of getting a dataLabel for datafiles from the DHS and then using it as input to the Instrument System instruction to take a data frame.

4.9.3 Action initiation/completion

The Sequencer must be capable of initiating actions in other tasks and then detecting their completion. In the case of more than one action proceeding simultaneously (Section 4.8) the Sequencer must be able to detect and distinguish the multiple completions.

An errors resulting from the commands must be handled correctly and reported. In most cases this should cause the Sequencer to pause.

4.9.4 Aborts

When the Sequencer receives an Abort command it is required to abort the sequence it is currently running as soon as possible. Exactly how best to achieve this is TBD. For some Sequence commands it will need to send an abort to the system it is commanding. For others (*e.g*. TAKEFRAME) it will also need to inform other systems (in the case the DHS would need to receive an ABORTFRAME command).

4.9.5 Posting of information

The Sequencer must be able to make certain information available to other systems, either on demand or continuously. This information is of two kinds:

1. Information about the Sequencer, what is the name of the sequence it is currently executing, what is its state, etc.
2. Information it is asked to make available by the Sequencer Command POST.

4.9.6 Generation of labels/numbers?

Not sure if this is required, it might be performed by the DHS and returned.

4.10 Standard Sequencer Commands

The Sequencer has a small number of simple standard internal sequence commands that it recognises. These are considered in this section.

4.10.1 BREAK

On encountering the Sequence command BREAK the sequencer will react exactly as if it had received the PAUSE command. That is it will:

1. Halt execution of the Sequence (any commands presently executing in parallel will be completed)
2. Place itself into the PAUSED state
3. Indicate this state to its user interface
4. Resume execution of the Sequence only on receipt of a CONTINUE command.
5. Receipt of other commands will result in the same action as indicated in Table 1.

4.10.2 BREAKPOINT

In normal execution the BREAKPOINT Sequence command is ignored by the sequencer. It should be used by its user interface to highlight points where the sequence may be "sensibly" broken (see [O-14]). However, by use of the command "Stop_At_Break" a BREAKPOINT may be made active. In this case the Sequencer will halt when it reaches this BREAKPOINT. If the parameter "GotoEnd" which arrives with the "Stop_At_Break" command is TRUE then the Sequencer will move to the next END Sequence command and put the Sequencer into the PAUSED state. If FALSE then the Sequencer will go into the STOPPED state.

4.10.3 WAIT <numsecs>

The Sequencer will wait for <numsecs> seconds before continuing with the execution of the sequence.

4.10.4 END <grpnum>

Specifies the END of a related group of commands, usually a "group" of observations. Is this the only "END"? What happens? (GRPNUM set to 0). Is this a command sent to the DHS?

4.11 Minimum Interface to the Instrument Control System

This is the interface that the Sequencer expects any instrument it commands to provide. For a more complete specification of the interface that an Instrument Control System is required to meet see [O-15]. All instruments expecting to be controlled by ORAC must meet the interfaces defined in that document. In this section we present the interface in the context of the Sequencer, and therefore the full details appropriate to the designer of an instrument control system are not presented. The instrument must provide these commands, even if they are implemented as null operations. However, it is possible for the instrument to provide additional commands, which must be entered into the relevant instrument configuration table for the sequencer. This allows the addition of new commands and the use of engineering commands.

Much of this interface is already defined (though not documented) for existing instruments (IRCAM3 and CGS4), but is not clearly delineated because the interface effectively exists as part of the same task (AIM) that carries out the observation sequencing. This interface is basically a command and parameter interface which lists the commands which must be obeyed by the instrument system in order to be configured and take data. Table 4 defines that interface more clearly.

This table is not exhaustive and defines what is possible with current instrumentation. It is intended that a table of this type will be available for each instrument, defining its abilities, and thus it should be possible to extend the dictionary understood by the sequencer by simply defining new commands for instruments.

Command	Parameters	Action	Comments
INITIALISE		Initialise the instrument	This is a NEW command, though existing instruments do allow for it.
DATUM	DeviceName	Causes the instrument sequencer to datum the component “DeviceName”	Device Name “ALL” is a special name to datum all devices.
LOADCONFIG	ConfigurationName	Reads in the configuration given in the file referred to by “ConfigurationName”.	This does not cause a configuration of the instrument
SET	ConfigurationType (BIAS|DARK|FLAT|ARC|OBJECT|SKY)	Set the instrument to a given configuration type.	This may be one of “BIAS”, “DARK”, “FLAT”, “ARC” or “OBJECT”
TAKEFRAME	FrameTag, ObservationType (BIAS|DARK|FLAT|ARC|OBJECT|SKY)	Take a data frame of the type indicated by the parameter, with the given FrameTag	This will move to the configuration relevant to the parameter even if the relevant “SET” was not issued first. Note that a “SKY” is the same as an “OBJECT” except for its label in the header.
ABORTRAME	FrameTag	Abort the data frame currently being acquired. Frametag is given as confirmation of the correct frame.	

TABLE 4. Instrument Interface Commands

This solution depends on there being simple configuration files that the instrument sequencer can read (see Section 4.5). The “instrument sequencer” is the task that is in overall command of the instrument, sequencing its functions.

It should be noted here that the ORAC Sequencer will not provide access to detailed, specifically engineering, functionality of instrument control systems. Those systems will be responsible for providing their own user interfaces and status displays. Detailed control of the instrument should only be necessary in unusual circumstances or if something has gone wrong. It has been agreed with the JAC that this detailed instrument control should only be available to the telescope operator and not to the observer (though of course instrument summary status displays must be available to the observer).

4.12 Interface to the Telescope Control System

The telescope control system already exists and is not changing in the near future, so this interface is already defined [X-11]. The telescope system runs under VMS-ADAM, and will be on a separate machine. Thus the Sequencer will need to be able to issue ADAM messages to the remote machine using the TCP/IP protocol. As ADAM now runs on Unix this is technically possible and is in use by some projects. There may remain some problems with the reliability of this communication path. Another option is to use the Drama ADITS task, which allows Drama tasks to control ADAM tasks. This option will be investigated.

In addition to the technology the command and parameter interface needs to be defined. Table 5 lists the commands and parameters this interface requires. Many of these already exist but some do not. The new commands and parameters required are noted.

As noted some of these new commands will require changes to the telescope control system. These changes are summarised here and presented for comment:

1. The telescope system must be able to read the contents of a “Target Description”. The details are TBD, but an initial version might contain the name and position of the target and the name and position of an offset guidestar. These must be held by the telescope system and usually used in response to commands that are related to acquiring a target.
2. The telescope system should be able to respond to the new OFFSET command and work out the actions required to best carry out the request. This will normally be moving the cross-head and telescope separately as in the “SLIDE” command to execute the movement as quickly as possible. However, if the movement will exceed the cross-head range, or take the guide-star off the dichroic then other approaches should be used. Details of this are TBD.

4.13 Interface to Other subsystems

The other major subsystems that Orac presently has to interface to are the Fabry-Perot (FP) etalon system and the infra-red waveplate (IRPOL) system. Both of these systems are presently VMS-ADAM based and thus will probably be controlled using the same solution as for the telescope system. The command interfaces are, however, substantially simpler. Table 6 and Table 7 outline them.

4.14 Interface to the Data Handling System

The data handling system will be designed to act as a server, storing data by request from the instrument data acquisition (for Michelle and later instruments this is presently the Edict system). However, it will be expected to provide labels (unique names) for the data sets and also to put them together with header information made available by the rest of the system, thus it must be informed by the Orac system, specifically by the Sequencer, that data frames are about to be acquired. Orac will need to make some header information of its own available.

There will also be a requirement to ensure that a data frame is connected with the correct set of header information. Therefore a “frametag” will be given to the dataset. This will be affixed to the header information and also given to the Instrument system, which must then return it to the data handling system with the data. (Query - how to do this for older instruments?)

The data handling system also needs to know whether or not data that is sent to it is for quick-look display only, for storage only or for both. Again this information will be given to the instrument and returned to the data handling system with the data frame.

Command	Parameters	Action	Comments
TARGETSLEW	ObjectName	Defunct	To be replaced by SLEW
TARGETWAIT	ObjectName	Defunct	To be replaced by the TARGET-DESC command.
TEL	BeamName	Defunct	To be replaced by NOD
NOD	BeamName (MAIN|OFF-SET)	Send the telescope to the “Beam Name” (MAIN or OFFSET)	MAIN and OFFSET should be already set up. May be extended to allow multiple NOD positions
CHOP	BeamName	Defunct	To be removed
TOFFSET	RAOffset, DECOffset	Offset the telescope to the given offset positions	Renamed from TOFF. (Is it still needed given the OFFSET command?)
XOFFSET	RAOffset, DECOffset	Offset the cross-head to the given offset positions	Renamed from XOFF. (Is it still needed given the OFFSET command?)
GUIDE	GuideState (ON|OFF)	Set the autoguider into the required state (ON or OFF)	Modified from present GUIDE command
NOGUIDE		Defunct	Replaced by parameters to GUIDE command
SLIDE	RAOffset, DECOffset	Defunct	To be replaced by OFFSET
SLIDE SLIT	Offset	Defunct	To be replaced by OFFSET
TARGETDESC	Description-Name	Telescope System reads the target description file and determines location of target and guide stars (plus any other information...)	NEW COMMAND. Will require changes to the telescope system to read and hold the information.
SLEW		Slew to the target currently setup	NEW COMMAND. To be used following a TARGETDESC command.
OFFSET	RAOffset, DECOffset	Offset the pointing of the telescope to the relevant offsets	NEW COMMAND. A replacement for TOFF and XOFF where the onus will be on the telescope system to decide how best to implement the offsetting
ONSOURCE		Wait for a telescope move to complete (to within tolerance)	NEW COMMAND (to ensure telescope is ready). Is this required??
FOFFSET	FocusOffset.	Offset the telescope focus position	NEW COMMAND (for “focus runs”)
ISU2	Position, Offset	Move the position of ISU2	This will move ISU2 to a different instrument port. For proper automation this will require knowledge of which instrument is where.

TABLE 5. Interface to the telescope control system

IRPOL	IrpolPosition (IN|OUT)	Move the IRPOL wave-plate module (to IN or OUT)	NEW COMMAND.
PEAKUP		Use the telescope PEAKUP facility to "peak-up" the counts from the source.	NEW COMMAND.

TABLE 5. Interface to the telescope control system

Command	Parameters	Action	Comments
DATUMPOL		Datum the IRPOL module	NEW: Is this required?
POL	Position Number	Move the IRPOL module to waveplate position given by "Position Number"	
POLDEG	Degrees	Move the IRPOL module to an angle "Degrees" degrees from datum.	

TABLE 6. Interface to the IRPOL system

Command	Parameter	Action	Comments
FPX	X position	Move the FP to the given X position	
FPY	Y position	Move the FP to the given Y position	
FPZ	Z position	Move the FP to the given Z position	

TABLE 7. Interface to the FP system

The data handling system also needs to know where to store data, etc. OK for one TAC (Time Allocation Committee) number, what about queue-scheduled observing or split nights?

TABLE 8. Interface to the Data Handling System

5.0 Backwards Compatibility

The ability of Orac to run the existing suite of UKIRT instrumentation (CGS4 and IRCAM3 in particular, but also including UFTI) is a requirement of the project. The interfaces outlined above are primarily aimed at new instruments, but their ability to control the existing instruments is sometimes clear. This section outlines what is required to meet this particular requirement.

Command	Sent Parameters	Returned Parameters	Action	Comments
INITIALISE	InstrumentName		Informs DHS of startup and which instrument this sequencer is handling	
STARTFRAME		FrameTag	Informs data handling system that a new data frame is about to start.	
STARTGROUP		GrpNum	A new group of frames is beginning, generate a group number	This might be a sequencer function, but probably DHS.
NEXTSEQ		GrpSeq	Return the next sequence number for a group	Not sure if this is a DHS function or a sequencer function
ENDGROUP	GrpNum		A group has completed.	Not sure if this is a DHS function.
ENDFRAME	FrameTag		This data frame is complete, you may store when all the data is received from the detector.	At this point all of the header items should be obtained except those that will come with the data frame from the detector.
TAKEHEADER	Stage		Take the header items for this Stage	
ABORTFRAME	FrameTag		Informs the data handling system that the frame it has been informed of has been aborted.	The data handling system will throw away this dataset.

5.1 The Instrument Control System

CGS4 and IRCAM3 do not have instrument sequencers as separate tasks that meet the interfaces described in Section 4.0, However, the AIM tasks does more or less meet this interface. Loading the AIM task under an alias relevant to the instrument and then not using the EXEC abilities within it would come very close to being what is required. The changes required here to make it fit should be small.

The same could be done for UFTI, which will be fitting in to the IRCAM3/CGS4 system.

The exact details of what is required will be considered at a later date.

6.0 Engineering Functionality

The Observing Sequencer requires some engineering functionality, most of which will not normally be available to the observer. This functionality falls into two main categories:

1. Functions that will be used in controlled engineering - laboratory testing, calibration, etc.
2. Functions that aid in the tracing of faults and errors during operations.

In attempting to meet the requirements the following functionality must be present in the sequencer:

1. The table-driven design outlined above means that it is possible to drive most functions of an instrument control system from the sequencer, even if the function is not listed in Table 3 as one required for normal observing. Adding the function to the table for the instrument will make it available in the command set.
2. In addition Table 3 does already include some commands that will normally be used during engineering only. These commands allow for repetitive testing of systems with some degree of control.
3. The observing sequencer may have an engineering user-interface that is different from the one used for normal observing, though this is not yet clear. This interface will allow the direct loading of sequences, and most of the same abilities to control the execution of the sequence (STOP, PAUSE, ABORT etc.). It will also allow the on-line editing of the current sequence.
4. The sequencer must have the ability to simulate any of the subsystems that it may control. This simulation only needs to be simple - i.e. it will not attempt to control the subsystem, merely simulate a successfully completed command. An adjustable time-delay might be useful.
5. Logging of commands received and the subsequent actions taken must be present, possibly at different verbosity levels.

7.0 Handling of Errors

Three types of errors can be identified in the context of the sequencer, errors in attempting some internal function, errors communicating with another system and errors returned from another system. The response of the Sequencer to each of these scenarios is presented here.

7.1 Internal Errors

These are errors which happen in functions that are internal to the Sequencer. Examples would be a failure to read a file (a Sequence or a configuration file) or some failure in attempting to allocate memory.

In response to these errors the Sequencer should fully report the error and any known reasons, suggest a correction (if appropriate) and, if in the middle of executing a sequence, halt it (treat as a PAUSE) and await further instruction.If in the middle of parallel commands it should complete the other commands that are in execution. Corrective action may be required and it may be a “user error”.

7.2 Communication Errors

These must occur whilst executing a Sequence. The sequencer will report the failure and halt (again treat as a PAUSE). It should report the reason for failure (as much as is known) and the Sequence command that failed. If in the middle of parallel commands it should complete the other commands that are in execution. Corrective action by the Observers will almost certainly be required.

7.3 Task Return Errors

These must occur whilst executing a Sequence. The sequencer will report the failure and halt (again treat as a PAUSE). It should report the reason for failure (as much as is given by the task in question) and the Sequence command that failed. If in the middle of parallel commands it should complete the other commands that are in execution. The system that returned the error should report more diagnostic information on its status screen. Corrective action may be required - this may also be a "user error".

8.0 Requirements Trace

This summarised how the design presented here meets the requirements set out in [O-5]. For the most part references are made to relevant sections of this document where the solution is described.

FO1. Must control current telescope system/allow access to full API of current TCS [X-11]. *Section 4.12*.

FO2. Must be able to control current and planned instruments. *i.e.* Michelle, UIST, UFTI, CGS4 and future instruments. *Section 4.11 and Section 5.1.*

FO3. Sequencer must be able to drive each instrument stand-alone (*i.e.* without other systems such as the TCS). *Section 6.0. This may be done easily by putting communications with other systems into simulation*.

FO4. Must read the output of the preparation system, but might be implemented by a translator in the Chooser/Scheduler. *Section 4.5. The Sequencer will not be able to read the output of the preparation system, but the translation provided by the Chooser/Scheduler will be accessible.*

FO5. Must co-ordinate with UKIRTDR to ensure DR gets correct recipes. *Section 4.14. The coordination is slight in that all that the DR system requires is the name of a DR Recipe in the data header.*

FO6. Must be able to *group*[1] related observations in such a way that UKIRTDR can recognise the groups. *TBA*

FO7. Should be able to run some of the sequence commands in parallel with others where this is possible (*e.g.* slewing the telescope and configuring an instrument). *Section 4.8*.

FO8. Must be able to sequence multiple (concurrent?) observations on multiple sources. (NB Desire to do calibrations in parallel with other observing). *Not directly addressed in the design, but the intent is that more than one incarnation of the Sequencer will be possible, each attached to a different instrument. See also [O-14].*

FO9. Observation sequence must be easily interruptible (visual feedback useful). *Now addressed in [O-14].*

FO10. The premature ending of a *group* must be communicated to UKIRTDR in some way. *Addressed in [O-12].*

FO11. Must be clear what part of the observation sequence is currently executing - a visual feedback of loops and similar constructs (sensible breakpoints?) would be useful. *Now addressed in [O-14].*

FO12. Must be able to dynamically modify current observation sequence (particular example - increase number of pairs). *Now see [O-14].*

FO13. Must allow for easy and rapid instrument switching. *Still TBA, however, the fact that more than one Sequencer may be run is the most likely solution.*

1. These are not necessarily the same as the existing idea of groups in the CGS4 system.

FO14. Must allow for separation of data for different observing programmes within a night and from one night to the next. (This implies close collaboration with the author of the data storage task.) *Still TBA, perhaps a function of the UKIRT DHS?*

FO15. Must allow for re-use of observations to enable calibrations to be shared between observing programmes. (Collaboration with data storage again?). *Still TBA*

FO16. Basic system to sequence and obtain data on a single source to be delivered with Michelle. *The delay in the delivery of Michelle makes it likely that the complete system will be available by the time of Michelle delivery.*

FO17. Aim to share philosophy and code with JCMT OCS system. *There is scope for some code reuse and sharing at the Scheduler/Chooser level.*

FO18. Changes required to CGS4, IRCAM and UFTI software to enable control from this sequencer are not part of the ORAC project. *A statement, rather than a requirement.*

9.0 Unresolved Issues

The following is a list of the known issues that are yet to be fully resolved.

1. Ensuring that switching instruments is rapid and easy. (But might be straightforward?)
2. Flagging and separation of data from different programmes.
3. Possible reuse of calibration observations.