Created:August 3, 1994
Modified:October 23, 1996

Gemini Controls Group Interface Control Document

ICD 7b — TCS Subsystem Interfaces

Philip Taylor

GSCG.grp.009-ICD07b/11

This document defines the parts of the Telescope Control System to subsystem interfaces that need consistent implementation.

1.0 Introduction

1.1 Purpose

This Interface Control Document (ICD) specifies the general way in which the Gemini Telescope Control System will communicate with its subsystems. It specifies those interfaces in the Gemini system which, while most of them exist entirely within the TCS, share a common implementation with similar interfaces found in the ICS and described in ICD/7a [10]. As such, these interfaces should be as consistent as is reasonable between the TCS and ICS to reduce development, maintenance, acceptance, and upgrade costs.

The interfaces involved are as follows:

- Control commands and responses from the TCS Master Control System to all TCS subsystems (PCS, SCS, AO, A&G, MCS, CRS, OIWFS and ECS)
- Status information from the TCS subsystems to the TCS Master Control System
- Alarms from the TCS subsystems to the TCS Master Control System

All of the above interfaces are to be implemented using the facilities provided with EPICS and are described in detail below.

While there are details of these interfaces that must differ between their use in the TCS and ICS, the fundamentals of the interfaces should be as consistent as possible to simplify maintenance and flexibility.

A summary of the required EPICS records that should be implemented in all TCS subsystems is given in the Appendix, Section 11.0 on page 21.

1.2 Scope

This document covers only the command, status, and alarms within the TCS and its subsystems. The movement of these entities from the TCS to other principal systems is covered by the *Gemini Software Design Description* [1], ICD/1a [4], ICD/1b [5] and ICD/2, [6].

This ICD defines the common features that must be present in the interface between the Telescope Control System and each TCS subsystem. Each TCS subsystem will have additional distinctive features that are not described here : this document only describes a common subset of the interface that the TCS can assume to be uniform amongst all subsystems. This ICD is not applicable to instrument control : interfaces between the Instrument Control System and its subsystems are described in ICD 7a — ICS Subsystem Interfaces, [10].

The specifics for each individual TCS to subsystem interface (such as the actual names of the commands and the parameters passed) are not given in this general software ICD. Individual interface details are covered in specific ICDs for each subsystem (see [18] - [25]).

No logging events are described here. A description of the interface between the Telescope Control System and its subsystems and the DHS logging system is given in ICD/4, [8].

The transfer of wavefront information across the Synchro Bus is not included in this ICD, but is described in ICD/5 [9] and ICD/10 [13].

The signalling of events on the event bus is not included in this ICD, but is described in ICD/11 [14].

The transmission of time data on the time bus is not included in this ICD, but is described in ICD/9 [12].

The signalling of interlocks is not included in this ICD, but is described in ICD/12[15].

1.3 Applicable Documents

The following documents should also be consulted:

[1] SPE-C-G0037, *Software Design Description*, Gemini 8m Telescope Project.

[2] SPE-I-G0009, *Software Programming Standards*, Gemini 8m Telescopes Project.

[3] gscg.sbw.063, T*he Official Gemini/EPICS Namespace List*, Steve Wampler & Steven Beard, Gemini 8m Telescopes Project.

[4] gscg.kkg.009, *ICD 1a — The System Command Interface*, Gemini 8m Telescopes Project.

[5] gscg.grp.024, *ICD 1b — The Baseline Attribute/Value Interface*, Gemini 8m Telescopes Project.

[6] gscg.kkg.010, *ICD 2 — Systems Status and Alarm Interfaces*, Gemini 8m Telescopes Project.

[7] gscg.grp.007, *ICD 3 — Bulk Data Transfer*, Gemini 8m Telescopes Project.

[8] gscg.grp.008, *ICD 4 — Logging Interface*, Gemini 8m Telescopes Project.

[9] gscg.grp.010, *ICD 5 — Wavefront Sensing Information Interface*, Gemini 8m Telescopes Project.

[10] gscg.grp.015, *ICD 7a — ICS Subsystem Interfaces*, Gemini 8m Telescopes Project.

[11] gscg.grp.012, *ICD 8 — Non-Conforming ICS Interfaces*, Gemini 8m Telescopes Project.

[12] sic_pbt_001, *ICD 9 - EPICS Time Bus Driver*, Philip Taylor, Royal Greenwich Observatory.

[13] gscg.grp.020, *ICD 10 — EPICS Synchro Bus Driver*, Gemini 8m Telescopes Project.

[14] gscg.grp.021, *ICD 11 — Event Bus*, Gemini 8m Telescopes Project.

[15] gscg.grp.017, *ICD 12 — Interlock System*, Gemini 8m Telescopes Project.

[16] *ICD 13 - Standard Controller*, Bret Goodrich and Andrew Johnston.

[17] gscg.grp.025, *ICD 16 — The Parameter Definition Format*, Steve Wampler, Gemini 8m Telescopes Project.

[18] *ICD 1.1.11/1.1.12 - The TCS/MCS Interface*, Andy Foster, Chris Mayer and Philip Taylor

[19] *ICD 1.1.11/1.5.4 - The TCS/CRCS Interface*, Chris Mayer

[20] *ICD 1.1.11/1.8 - The TCS/AOS Interface*, Brian Lecke and Chris Mayer

[21] *ICD 1.1.11/1.2.10 - The TCS/PCS Interface*, John MacLean and Chris Mayer

[22] *ICD 1.1.11/2.1.3 - The TCS/ECS Interface*, Chris Mayer

[23] *ICD 1.1.11/1.6.2 - The TCS/A&G Interface*, Chris Mayer and Malcolm Stewart

[24] *ICD 1.1.11/1.10 - The TCS/OIWFS Interface*, Chris Mayer

[25] *ICD 1.1.11/1.4.4 -- The TCS/SCS Interface*, Sean Prior

[26] TCS/PTW/6 -*Time*, P.T. Wallace, Rutherford Appleton Laboratory.

[27] ocs.kkg.031, *“Preliminary Sequence Command Specifications”*, Kim Gillies, Shane Walker & Steve Wampler, NOAO/Gemini 8m Telescopes Project.

[28] cics_smb_002, *“Core Instrument Control System — Introduction & Specification”*, Steven Beard, Royal Observatory Edinburgh.

[29] EPICS *Alarm Handler User’s Guide*, Martin Kraimer, Ben-Chin K. Cha, Mark Anderson, Janet Anderson, September, 19, 1991, Los Alamos National Laboratory.

[30] EPICS *Alarm Configuration Tool and Alarm Handler User Interface System Requirements Definition (DRAFT)*, Janet Anderson, Ben Chin-Cha, August 30, 1993. Los Alamos National Laboratory.

[31] EPICS *Display Manager User’s Guide*, Los Alamos National Laboratory.

[32] EPICS *Archiving Reference Manual* (DRAFT), Roger Cole, Los Alamos National Laboratory. [N.B. This document is in preparation. The Gemini project has a draft copy. Contact the author for up to date information].

[33] EPICS *Channel Access Reference Manual*, Jeffrey O. Hill, EPICS Group, Los Alamos National Laboratory.

[34] EPICS *Input/Output Controller Record Reference Manual*, J.B. Anderson, M.R. Kraimer, October, 1992.

[35] EPICS *IOC Application Developers Guide*, Los Alamos National Laboratory.

[36] EPICS *State Notation Language and Run-time Sequencer Users Guide*, Andy Kozubal, Los Alamos National Laboratory

[37] CAPFAST *Electronic Circuit Design CAE User’s Manual*, Phase Three Logic Inc., Beaverton, Oregon.

[38] *genSub EPICS Record Reference Manual*, Andy Foster, Royal Greenwich Observatory.

1.4 Abbreviations and Acronyms

A&G	Acquisition and Guidance
AO	Adaptive Optics
CA	Channel Access
CAD	Command Action Directive
CAR	Command Action Response
CICS	Core Instrument Control System
CRS	Cassegrain Rotator control System
DC	Detector Controller (software system)
DHS	Data Handling System
ECS	Enclosure Control System
EPICS	Experimental Physics and Industrial Control System
FITS	Flexible Image Transport System
GCS	Gemini Control System
GIS	Gemini Interlock System
ICD	Interface Control Document
ICS	Instrument Control System
IOC	Input Output Controller
IS	Instrument Sequencer
LAN	Local Area Network
M1	Primary Mirror
M2	Secondary Mirror
MCS	Mount Control System
OCC	Optical Components Controller
OCS	Observatory Control System
PCS	Primary mirror (M1) Control System
PDF	Parameter Description File
SAD	Status/Alarm Database
SCS	Secondary mirror (M2) Control System
SDD	Software Design Description.
SIR	Status/Information Record (EPICS)
TCP/IP	Transmission Control Protocol/Internet Protocol
TCS	Telescope Control System
TCSS	Telescope Control System Subsystem
VSM	Virtual System Mode
UDP/IP	User Datagram Protocol/Internet Protocol
WAN	Wide Area Network

1.5 Glossary

VME	A real-time system obeying the ANSI/IEEE 1014-1987 Versatile Backplane Bus standard.
VxWorks	The Real Time Operating system from Wind River

1.6 Stylistic Conventions

References to documents in Section 1.3, “Applicable Documents,” on page 2 are given like this; [1].

2.0 Overview

An overview of the whole Gemini Control System (GCS) is given in the Gemini *Software Design Description*, [1]. The GCS is made up of principal software systems: the Telescope Control System (TCS), the Data Handling System (DHS), the Observatory Control System (OCS), and up to four Instrument Control Systems (ICS). In addition, these principal systems contain low-level subsystems which are responsible for a particular area. The TCS is responsible for the following TCS subsystems:

- Acquisition and Guidance subsystem (A&G)
- Onboard Instrument Wavefront Sensor (OIWFS)
- Adaptive Optics subsystem (AO)
- Cassegrain Rotator Control subsystem (CRS)
- Enclosure Control subsystem (ECS)
- Mount Control subsystem (MCS)
- Primary mirror (M1) Control subsystem (PCS)
- Secondary mirror (M2) Control subsystem (SCS)

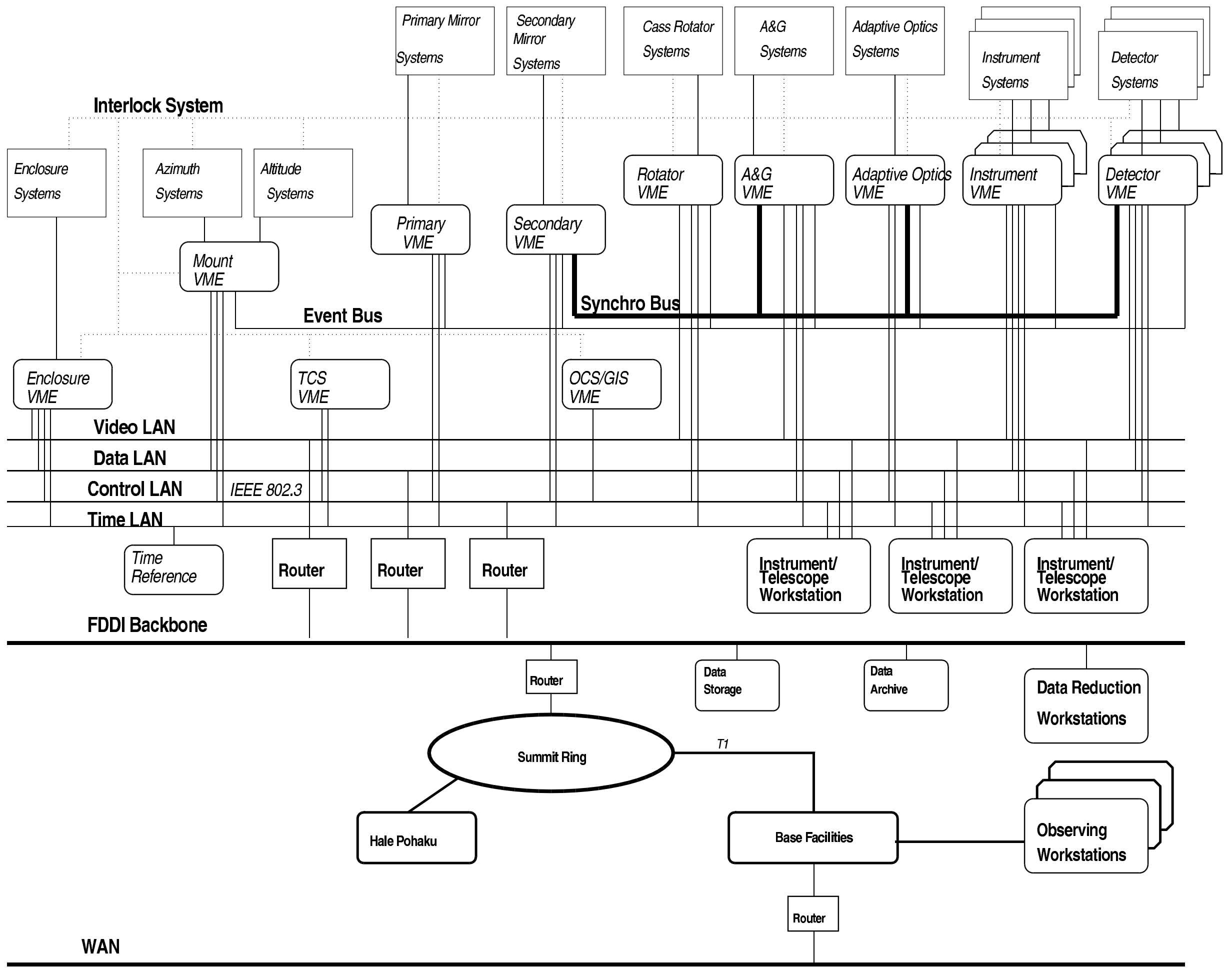
The TCS Master control system is responsible for managing the interface between these systems and other Principal Systems (OCS, DHS and ICS). Commands to the TCS subsystems will all be issued via the TCS Master Control system - Principal Systems should not issue commands directly to TCS subsystems.

2.1 System Hardware Architecture

Figure 1 on page 6 shows the full Gemini system hardware architecture, including the TCS and its subsystems.

The systems that must conform to this ICD are VxWorks/VME based using EPICS for internal communications and control. The TCS is distributed across multiple VxWorks systems, one for the TCS Master Control System and one for each TCS subsystem. The Standard Controller documentation [16], provides details on recommended VME hardware.

FIGURE 1. The Full Gemini System Hardware Architecture



2.2 Communication Architecture

2.2.1 Control and Status Interface

The TCS and its subsystems communicate internally using EPICS Channel Access across a high performance LAN (the Control LAN). Communications are established using UDP, but run using TCP/IP.

2.2.2 Time Bus

Accurate time is distributed to the TCS and all its subsystems using the Time Bus with time supplied from Bancomm 635/637 cards. The TCS contains the bc637 card with built-in GPS receiver and is thus the time master. Each TCS subsystem contains a bc635 card using time distributed to it from the TCS via the Timebus. See document TCS/PTW/6 [26] for further details of the TCS and the Time Service.

Access to the Timebus from TCS subsystems should be made via calls to routines in the TCS Time System library (`timelib`) which is distributed as part of the Gemini common software environment and should be loaded in each TCS subsystem IOC. See Appendix to ICD 9 [12] for a full description of the `timelib` routines.

2.2.3 Synchro Bus

The Synchro Bus is used for fast transfer of data between TCS subsystems : the TCS itself is not connected to the Synchro Bus. See ICD 10 [13] for a description of Synchro Bus facilities.

2.2.4 Event Bus

The Event Bus is used to transmit Chop/Nod event information between subsystems, for example between the SCS and ICS subsystems. See ICD 11 [14] for a description of Event Bus facilities. The TCS itself is not connected to the event bus.

2.2.5 Gemini Interlock System

The Gemini Interlock System disables and enables specific devices and is connected to the TCS and all its subsystems. TCS subsystems will be able to raise interlocks via the GIS but the TCS itself will only be able to detect that an interlock has been set. See ICD 12 [15] for a description of the Gemini Interlock System.

2.3 TCS Context Diagram

Figure 2 on page 8 shows a context diagram as seen by the TCS system. The Gemini systems appear to the interface as entities in the outside world. This diagram may be compared to the “Gemini Control System” data flow diagrams in [1] and [28].

.

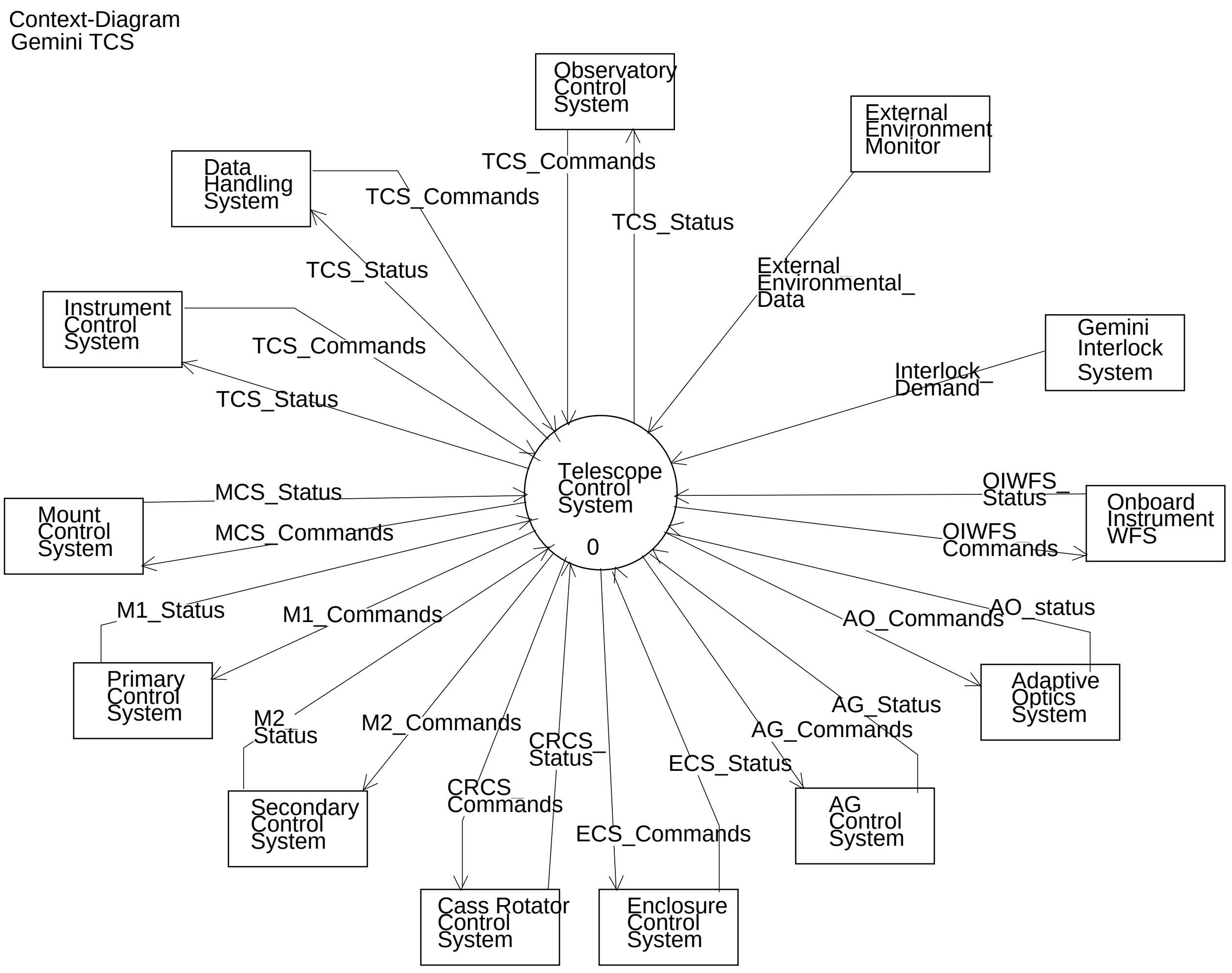


FIGURE 2. TCS Context Diagram

2.4 TCS General Events and Responses

The events that may happen on the TCS Subsystem Command and Responses interface, and the responses to those events, and shown in Table 1 on page 9

.

TABLE 1. Events and Responses

Event	Response
The TCS issues a command to a subsystem.	The subsystem checks the arguments given with the command and either accepts or rejects it. Acceptance or rejection of the command is acknowledged. (At this point the command initiation transaction is complete). If the command is accepted it is obeyed. Command completion is acknowledged when the command completes (see the event "A TCS subsystem acknowledges command completion." on page 9).
The TCS updates a public process variable contained within the EPICS database belonging to a subsystem.	The subsystem uses the new value in the way it has been programmed to. N.B. The subsystem response depends on how it is set up to use the process variable. In most cases the TCS will already have activated a command (such as FOLLOW) requesting that the subsystem monitor the variable, and the variable will contain a parameter (such as a position) which the subsystem must attempt to follow.
The TCS begins monitoring a process variable in a subsystem.	The subsystem does nothing, as the monitoring is handled transparently by EPICS Channel Access. The TCS receives updates on the value of the process variable. This may be by scan or on changes outside a 'dead-band'.
The TCS begins monitoring the alarm state of a process variable in a subsystem.	The subsystem does nothing, as the monitoring is handled transparently by EPICS Channel Access. The TCS receives any alarms triggered in that process variable.
A TCS subsystem acknowledges command completion.	The TCS receives the acknowledgement, together with an indication of the success or failure of the command and a message describing the command status.
A TCS subsystem changes a process variable.	If a CA client has been set up to monitor the state of that process variable then it will receive an update. If no CA clients are monitoring the process variable nothing will be transmitted across the subsystem interface.
A TCS subsystem signals an alarm.	The OCS alarm handler, which should be monitoring the alarm state of all the public process variables in each subsystem, receives the alarm and reports it. If the TCS has been set up to monitor that particular alarm it will also receive notification and can take appropriate action.

TABLE 1. Events and Responses

Event	Response
A TCS subsystem IOC crashes or hangs up.	The OCS alarm handler, which should be monitoring the alarm state of all the public process variables in each subsystem, receives an alarm from the TCS and reports it. If the TCS is monitoring any process variables or waiting for any outstanding commands it will detect a connection loss and raise an alarm indicating that the subsystem is no longer connected.
A TCS subsystem IOC is rebooted.	The subsystem rebuilds its EPICS database, then executes its start-up procedure, equivalent to executing the INIT sequence command. It then becomes available again in its start-up state. The TCS can reconnect to a rebooted subsystem without having to be rebooted itself (this is a property of EPICS).

Note that only the TCS issues commands. A subsystem can only cause an event by changing a process variable or issuing an alarm. (It can also log information as specified in ICD/4, [8]).

3.0 EPICS Implementation

Besides the basic EPICS channel access implementation and EPICS utilities this interface also relies on the implementation of:

- Command Action Directive (CAD) and Command Action Response (CAR) records, whose implementation is described in ICD/1b, [5].
- Status Information (SIR) records, whose implementation is described in ICD/2, [6].
- `genSub` records, used to pass an array of data between subsystems. whose implementation is described in reference [38].

These records have been developed as part of the Gemini Project and are included in the standard Gemini EPICS distribution.

3.1 EPICS Record Names

Since EPICS provides a global namespace for its records, every record in the Gemini Control System must have a unique identifier. This is accomplished by using a unique prefix for records in each TCS subsystem's EPICS database. (For further details see [3]) Each record name incorporates the two letter prefix and a colon in front of the record name. A standard set of database name prefixes has been defined for the TCS and its subsystems as follows

TABLE 2. TCS subsystem database name prefixes

Subsystem	Prefix
Telescope Control System	TCS
A&G	AG
Adaptive Optics	AO
Cassegrain Rotator	CR
Enclosure	EC
Primary Mirror	M1
Secondary Mirror	M2
Mount	MC
On-Instrument Wave Front Sensor	??

3.1.1 Record Name Structure

In subsequent sections, record names are shown with the prefix `<sys:>` representing the actual subsystem name prefix. The names of subsystem records mentioned in this document should be restricted to this simple form without additional name components due to Capfast diagram hierarchical naming conventions. The record names should be set to the required value using Capfast properties where necessary : for example, using the Capfast property `name` to specify the name of the record. (See the `e2sr` documentation for further details).

4.0 TCS Sequence Commands

Sequence commands (TEST, INIT, RESET, VERIFY, ENDVERIFY, GUIDE, ENDGUIDE, OBSERVE, PAUSE, CONTINUE, STOP, ABORT and PARK) are commands that are initiated by the OCS and which all Gemini Principal Systems must be capable of obeying. See reference [27] for further details. The TCS will pass on some of these commands to its subsystems. Many sequence commands will cause no action in the TCS and so will not be passed on to TCS subsystems. The sequence commands shown below in Table 3 should be implemented in all TCS subsystems.

TABLE 3. TCS Subsystem Sequence Commands

Sequence Command	Description
TEST	The TCS subsystem will perform self tests to ensure its systems are healthy.
INIT	The TCS subsystem will execute its complete initialisation sequence, in which it will: read its hardware setup files and any lookup tables, datum all the mechanisms which need to be datumed and reset itself to the start-up state.

TABLE 3. TCS Subsystem Sequence Commands

Sequence Command	Description
RESET	The TCS subsystem will reset itself to the start-up state. No configuration files will be read and no mechanisms will be moved. After a RESET the hardware setup parameters should revert to the values they had at boot time but all mechanisms are in a state where they need to be redatumed.
PARK	The TCS subsystem will move its mechanisms to a configuration in which it can safely be switched off.

5.0 TCS subsystem Commands

In addition to the sequence commands passed on from the TCS, each TCS subsystem has commands of its own, many of which will be specific to the particular hardware of that subsystem and should be documented in the appropriate ICD (See [18] - [25]). Subsystem commands come in two varieties; “One-off” and “Continuous”, as described below.

Normally the following two commands will be implemented in all subsystems, as follows

- **MOVE**. A one-off command to move the principal subsystem mechanism to a specified position, then maintain that position. The mechanism will not necessarily be braked or locked at the demanded position.
- **FOLLOW**. A continuous command which causes the subsystem’s principal mechanism to follow a continuous stream of demanded positions (not every subsystem will require this command).

5.1 One-Off Commands

These are commands which cause some discrete and well-defined action to take place. The action starts, is obeyed and then completes. Attributes and parameters are passed for one-off commands using the argument field of the appropriate CAD record. Typically this will consist of fields containing the demanded position(s) of a subsystem mechanism.

An example of a one-off TCS subsystem command is MOVE to move the main mechanism controlled by the subsystem to the position specified by the parameters, then stop at that position. One-off commands are implemented by issuing a CAD START directive then waiting for a command completion response.

5.2 Continuous Commands

Continuous commands are those that require continuous action from the subsystem until further notice. An example of a continuous TCS subsystem command is one to request that the telescope tracks a target. In general these commands are executed by sending a continuous stream of demands to a subsystem which is monitoring those demands. The monitoring activity of the subsystem is switched on and off by issuing a “one-off” command using an EPICS CAD record.

The principal continuous command that most TCS subsystem will implement is FOLLOW, which moves the main mechanism controlled by the subsystem using a stream of demanded positions output by the TCS as an array of data. Subsystems will only implement this command when the position update rate is higher than that appropriate for a one-off MOVE command.

5.2.1 Continuous Command Data

The demand data stream generated by the TCS for a continuous command consists of an array of values in a single output field of a `genSub` EPICS record. The `genSub` record is an extended version of an `mosub` record where the type and number of elements in each field can be specified. For further details see reference [38]. The set of data variables transferred must be consistent (all applicable at the same time) even when the producer and consumer systems are running asynchronously on separate IOCs. The subsystem should have a corresponding `genSub` record called `followA` to receive the demanded data array.The output data array from the TCS is comprised of the following items:

TABLE 4. Continuous Command Data Items

Item	Type	Description
Timestamp	DOUBLE	A timestamp recording the time at which the demand was sent from the Telescope Control System to the subsystem. This will be used to monitor the elapsed time between the sending and receipt of the demand value. The TAI timescale should be used.
Target time	DOUBLE	The target time at which the demands are to be applied. The TAI timescale should be used.
Trackid	DOUBLE	Unique identifier for the current stream of demands (see Section 5.3 and Section 5.3.2 below).
Demand(s)	DOUBLE	A set of one or more demanded position values.

As the Control LAN is non-deterministic, variable delays will occur between sending and receiving the position data. Normally, the target time should be later than the time of receipt of the data. If a delay occurs such that the target time value is earlier than the time of receipt then the associated demand values should be used by the subsystem to calculate the next demand positions by extrapolation.

The subsystem should verify that the data received is valid and not excessively delayed and indicate this status in a SIR record called `arrayS`. Erroneous input data, large numbers of late arrivals or excessively large delays, should result in an alarm being raised via this record.

5.3 Continuous Command Synchronisation

The TCS continuously generates timestamped subsystem position demands at 0.05s intervals. When it wants a subsystem (such as the MCS) to track it sends the subsystem a FOLLOW command and updates the track identifier in the demand data array. These demands will normally be changing sufficiently slowly that the MCS will, in the absence of some error condition, always be "in position". When the TCS wants to move the telescope to a new position on the sky it will send another FOLLOW command and the Az/El demand will change abruptly. This abrupt change will often, but not always, be sufficiently large that the MCS will be unable to remain in position.

The TCS needs to be able to detect when its subsystems are in position after sending a FOLLOW command so that it can inform the OCS that the command that initiated the change in position has completed. It is not sufficient for the TCS simply to monitor the “in position” status of the subsystem since there is no way for the TCS to know whether the subsystem status refers to a demand before or after the change in position. The problem cannot be resolved in the subsystem either because the change in demand is not synchronised with the arrival of the corresponding FOLLOW command.

The solution is to add an identifier (type DOUBLE) to the timestamped demands that changes each time the telescope is moved to a new position; the subsystem must copy this identifier to an EPICS analogue output record called `trackid`. The TCS can then monitor this record in order to determine which stream of demands (or “track”) the subsystem is currently attempting to follow.

5.3.1 `activeC` CAR Record

The subsystem must maintain an EPICS CAR record called `activeC` which reflects the status of the subsystem mechanism moving to a demanded position.For a detailed description of the algorithm to set `activeC` when used with a continuous command, see Section 5.3.3 below. For a definition of the values that `activeC` may take, see Section 6.3 below.

5.3.2 `trackid` Record

The subsystem must maintain an EPICS analogue output record called `trackid` containing the identifier of the stream of demands that the subsystem is currently acting on. The value of this record must be updated with a new value as soon as it is received in the stream of demands. For details see Section 5.3.3 below.

5.3.3 `activeC` Algorithm

The algorithm for determining the `activeC` status is as follows:

`demandid` == `trackid` input with the latest position demand from the TCS.
`currentid` == `trackid` value currently known to the subsystem.

```
if ( currentid != demandid )            (1)
{
  currentid = demandid;
  activeC = CAR_BUSY;                   (2)
  if ( inPosition )
    activeC = CAR_IDLE;                 (3)
}
else
{
  if( inPosition )
      activeC = CAR_IDLE;               (4)
  else if ( activeC == CAR_IDLE )
           raise inPosition alarm;      (5)
```

```
          else if (error during slew)
              activeC = CAR_ERROR;          (6)
          else
              activeC = CAR_BUSY;
 }
```

(1) A new trackid has entered through the `genSub` record. Copy this new value into the local value `currentid`.

(2) Now slewing to the new position : always set to `CAR_BUSY`.

(3) It was a small move and the mechanism is in-position immediately.

(4) Now in-position for the first time after a slew.

(5) The subsystem had previously reached the demanded position but has now gone out of position, for whatever reason.

(6) A serious error has occurred that will prevent the mechanism achieving the demanded position.

Note that an error of going out of position during tracking but with no apparent hardware fault is reflected by the `inPosition` status item, not the `activeC` CAR record (see below).

6.0 Status Information

Status information is provided by each TCS subsystem through EPICS CAR records and public EPICS SIR records. The SIR records should be incorporated as part of the Status and Alarm Database (SAD).

The status information provided by each subsystem should be declared in a Parameter Description File (PDF) as specified in ICD/16, [17].

Each TCS subsystem should provide SIR records in the Status and Alarm database that includes at least the following records :

6.1 `<sys>:state`

This SIR record will take one of the integer values Booting, Initialising or Running and reflects the state of the subsystem. The activities corresponding to these states are shown in the following table.

TABLE 5. `state` SIR record values

Symbolic Name	Numeric Value	Description
`BOOTING`	0	The state at `iocInit` and whilst any database default values are being set

TABLE 5. `state` SIR record values

Symbolic Name	Numeric Value	Description
`INITIALISING`	1	The subsystem will be checking its own hardware during this phase
`RUNNING`	2	The subsystem is ready to accept commands. Commands will only be accepted when the subsystem is in this state.

6.2 <sys>:health

This SIR record will reflect the overall health of the subsystem as an integer value. This value will be set by the subsystem by monitoring all relevant mechanism status. Defined health values are :

TABLE 6. health SIR record values

Symbolic Name	Numeric Value	Description
`GOOD`	0	All subsystem mechanisms are operating correctly.
`WARNING`	1	Some parts of the subsystem have problems; the subsystem is functioning but may not perform to specification.
`BAD`	2	Subsystem is unusable.

6.3 <sys>:activeC

This CAR record reflects the status of the subsystem mechanism moving to a demanded position. It should be used in conjunction with all subsystem commands that cause the demanded position(s) to change e.g. `FOLLOW`, `MOVE` and `ZEROSET`. See Section 5.3.3 for a full description of its use when a continuous stream of demands is being followed. It may take one of the following values :

TABLE 7. activeC CAR record values

Symbolic Name	Numeric Value	Description
`CAR_BUSY`	4	The subsystem has received a new demand but has not yet achieved the demanded position.
`CAR_IDLE`	1	The subsystem has achieved the demanded position within the required tolerance. Any subsequent drift outside the limit will not change this value.
`CAR_ERROR`	3	The subsystem has detected a serious hardware failure that will prevent it reaching the demanded position.

6.4 <sys>:inPosition

This SIR record will reflect the difference between the actual mechanism position and the position demands associated with any command (e.g. `MOVE` or `FOLLOW`). The value is an integer TRUE or FALSE, reflecting whether the current position error is within the allowed tolerance. The tolerance value for the position error may vary at different times.

An alarm should be raised in this record when its value goes to FALSE and the subsystem mechanism is currently tracking the current stream of demands (i.e. `activeC` has value `CAR_IDLE`). No alarm should be raised if `activeC` has value `CAR_BUSY` or `CAR_ERROR`.

Defined values for the status values are as follows:

TABLE 8. inPosition SIR record values

Symbolic Name	Numeric Value	Description
`FALSE`	0	The current difference between demand and achieved position is greater than the maximum tolerance.
`TRUE`	1	The current difference between demand and achieved position is less than or equal to the maximum tolerance.

6.5 <sys>:present

This record will be used to indicate that the subsystem is currently operational. It may be of any type and must be in a database resident in the subsystem IOC (*not* the Status and Alarm Database). Its type and values are undefined : the record exists simply to provide a record residing in the subsystem IOC with known name that can be monitored by the TCS to ensure that the subsystem is still active.

6.6 <sys>:arrayS

This SIR record will be used to reflect the status of the demand data from the TCS to the subsystem. As described above in Section 5.3, the demand data array also includes timestamps showing the time the data was sent and the time at which the demanded position is to be applied by the subsystem. This SIR record should raise an INVALID DATA alarm if the demanded positions are illegal or the delays between transmission and receipt are excessive.

Defined values for the status values are as follows:

TABLE 9. arrayS SIR record values

Symbolic Name	Numeric Value	Description
`VALID`	0	The demand data array from the TCS contains valid data and the time delays from transmission to receipt are within tolerance.
`INVALID`	1	The demand data array from the TCS contains invalid data.
`TIMEOUT`	2	The time delays from transmission to receipt of the data array from the TCS are out of tolerance.

7.0 Debugging

7.1 Debugging Modes

The TCS may direct any subsystem to operate in any of the following debugging modes:

- NONE — There is no debugging. The subsystem operates normally.
- MIN — There is minimal debugging. The subsystem (perhaps) provides a commentary on some of its actions by means of the EPICS logging system but it does not carry out any operation that might severely affect its performance.
- FULL — Full debugging. The subsystem (perhaps) provides a commentary and carries out extensive checks on its operation, which may result in a large degradation in performance.

The exact meaning of the above debugging modes is the responsibility of whoever builds the subsystem. They may also invent further levels, as long as the three basic ones are recognised. The commentary reported by a subsystem running in debug mode should be sufficiently explicit for a programmer familiar with the subsystem to diagnose a problem. (For example the operator may try temporarily running a faulty system in a debug mode and report the log file to a remote programmer).

Subsystems should initialise themselves at debug level NONE unless specifically directed to do otherwise.

7.2 DEBUG Command

Each TCS subsystem must provide this command which switches the subsystem to the requested debugging level from the set: `NONE`, `MIN` or `FULL`. All three debugging level values must be accepted by all TCS subsystems even though, as described above, each debugging level may not be distinct.

8.0 Error, Alarm and Logging System

Errors may be reported in the following ways:

- By reporting events via the EPICS alarm handler, as described in ICD/2, [6], and [29]. This method is particularly useful for reporting spontaneous events that do not have to be connected with a command.
- By rejecting a command through the “Accept/Reject” and “Reason” fields in an EPICS CAD record, or by reporting the failure of a command through the “Status” and “Message” fields of an EPICS CAR record, as described in ICD/1b, [5].
- By logging error messages using the standard Gemini logging system, as described in ICD/4, [8], and [32]. This mechanism can be used to elaborate on an error reported using the other two methods (especially in a debugging mode), but it should *not* be used alone.

9.0 System Attributes

9.1 Maintainability

9.1.1 Interface Design Recommendations

The interface to the TCS subsystems should be through the EPICS database structure and remain consistent with the EPICS database interface presented in [33]. In order to reduce the complexity of maintaining this database, the database should be developed, modified, and maintained using the CAPFAST EPICS database development tool, [37]. The design should reflect the hierarchy that is natural to the application.

To simplify maintenance the interface should obey the following rules:

- All messages between the TCS and subsystem should be exchanged by EPICS channel access only. Database access should not be used (this will not normally be possible as the TCS and its subsystems will be running in separate IOCs).
- Commands should be issued using EPICS CAD records.
- Status of actions resulting from CAD records should be reported using EPICS CAR records.
- Status information should be made available through public SIR records.
- Each system’s public information should be documented in a Parameter Definition File (PDF), as specified in ICD/16, [17].

9.1.2 Adaptability and Enhancement Potential

If additional EPICS database records and support code are needed, then these should be discussed with the Gemini Project Office to avoid duplicating work done elsewhere.

10.0 Development and Test Factors

10.1 Project Control

Final authority over the suitability of a particular application for this interface resides with the Gemini Project Office.

10.2 Deliverables

The interfaces described in this report are all directly implemented using EPICS. Consequently, applications of this interface should be in the form of an ICD which conforms to these guidelines and includes a full description of the functionality accessible through the interface.

10.3 Acceptance Testing

Gemini systems must be able to run in a mode that allows their communication with other Gemini systems to be tested. A simulator should be provided to mimic the behaviour of this interface.

10.3.1 Simulation Modes

A subsystem may operate in any of the following simulation modes:

- VSM — The subsystem is to operate in ‘Virtual System Mode’ (a generalization of the concept of the Virtual Telescope Interface). Here, actions may be initiated and checked, but the subsystem does no further computations.
- FAST — The subsystem’s event processing and responses are enabled, and some internal computations may be performed, but response times are not realistic. Responses from lower subsystems are simulated.
- FULL — Full simulation, events and responses are enabled and subsystem responses take realistic time. The subsystem performs all internal computations, but responses from lower subsystems are simulated with realistic timings.
- NONE — There is no simulation. The system is to operate normally.

These modes are to be provided as the system is developed in the order shown here. The VSM mode provides a way for the TCS to check command flows and completeness. FAST mode allows for the testing of interfaces in an integrated environment, while FULL permits the testing of a subsystem without lower subsystems operating.

The subsystem simulation mode is defined at boot-time and cannot be changed until another reboot is performed - dynamic simulation mode changing is not supported.

NOTE: Subsystems should *not* automatically fall back into a simulation mode if their hardware is not present or fails to respond. Such failures should *always* result in an error unless the subsystem is explicitly directed into a simulation mode. This rule is necessary to ensure that simulation modes are not used accidentally, which can waste observing time.

11.0 Appendix : Required EPICS Records

The following table summarises the minimum set of EPICS records required by this ICD to be implemented in all TCS subsystems. The subsystem record name prefix is here shown as `<sys>:`. See Section 3.1 for a list of the standard prefix for each subsystem.

Record Name	Record Type	Description	Reference
`<sys:>test`	CAD	Sequence command to perform subsystem self-test	Table 3 on page 11
`<sys:>init`	CAD	Sequence command to initialise subsystem	Table 3 on page 11
`<sys:>reset`	CAD	Sequence command to reset subsystem	Table 3 on page 11
`<sys:>park`	CAD	Sequence command to park subsystem mechanism(s)	Table 3 on page 11
`<sys:>move`	CAD	Subsystem (one-off) command to move mechanism to a specified place and stop	Section 5.0
`<sys:>follow`[a]	CAD	Continuous command to follow a stream of position demands sent from the TCS in an array.	Section 5.2 and Section 5.0
`<sys:>debug`	CAD	Set the debug level in the subsystem	Section 7.2
`<sys:>followA`[a]	`genSub`	Receive array of position demand data from the TCS	Section 5.2.1
`<sys:>activeC`	CAR	Status of the subsystem mechanism moving to a demanded position	Section 5.3.1, Section 5.3.3 and Section 6.3
`<sys:>trackid`[a]	AO	Identifier for the current stream of demands being acted on	Section 5.3.2 and Section 5.3.3
`<sys:>state`	SIR	Report the state of the subsystem IOC	Section 6.1
`<sys:>health`	SIR	Report the health of the subsystem's mechanisms	Section 6.2
`<sys:>inPosition`	SIR	Whether the current mechanism position is close enough to the demanded position	Section 6.4
`<sys:>arrayS`[a]	SIR	Status of continuous position demand data	Section 6.6
`<sys:>present`	`(any)`	Used to monitor whether the subsystem IOC is running	Section 6.5

a. These records are only required for subsystems that implement continuous commands (e.g. the FOLLOW command).