MRO Delay Line

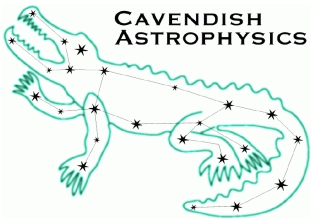
Production Delay Line Control Software Architecture

INT-406-VEN-1001

John Young
jsy1001@cam.ac.uk

rev 1.0

22 January 2010



Cavendish Laboratory
JJ Thomson Avenue
Cambridge CB3 0HE
UK

Change Record

Revision	*Date*	*Authors*	*Changes*
0.1	2010-01-15	JSY	Initial version.
1.0	2010-01-22	JSY	Removed open issue of saving configuration changes. Minor edits.

Objective

To describe the overall architecture of the production control software for the MROI delay line system. To outline the deliverables of the production delay line software contract.

Scope

This document provides an overview of the software to be delivered under the production delay line software contract. We define the components of the software, outline their functions, and describe the changes from the equivalent prototype software. We describe how the control software functions as an integrated system to close the required servo loops and to provide a high-level interface to the MROI ISS.

Reference Documents

RD1 Delay Line Production Software Development Statement of Work

RD2 Requirement Specifications for the MROI “production” delay line software INT-406-CON-0101

RD3 Network Message Protocols and Telemetry/Status/Logs File Format INT-406-VEN- 1007

RD4 Production Trolley Software Functional Description INT-406-VEN-1002

RD5 Production Shear Sensor Software Functional Description INT-406-VEN-1005

RD6 Production Metrology Software Functional Description INT-406-VEN-1004

RD7 Production Workstation Software Functional Description INT-406-VEN-1003

RD8 Delay Line Analysis GUI Functional Description INT-406-VEN-1006

RD9 Design of an MROI System INT-409-ENG-0020

Applicable Documents

AD1 Requirements Specification for the MROI Delay Line System INT-406-TSP-0002

AD2 Commissioning Plan and Performance Verification Milestones for the MROI INT-414-TSP-0004

AD3 (Prototype) Control Software Architecture INT-406-VEN-0101

AD4 Trolley Electronics Design Description INT-406-VEN-0112

Acronyms and Abbreviations

API	Application Programming Interface
DL	Delay lines
dlmsg	Cavendish delay line messaging protocol
FDR	Final Design Review
FT	Fringe Tracker
GUI	Graphical User Interface
ISS	Interferometer Supervisory System
MCBD	ISS Monitor and Configuration DataBase
MROI	Magdalena Ridge Observatory Interferometer
OPD	Optical Path Difference
PVM	Performance Verification Milestone
SOW	Statement of Work
SSH	Secure Shell
TBC	To Be Confirmed
TBD	To Be Determined

Table of Contents

1 Introduction and Context

Following on from the Final Design Review for the MROI delay lines, the need for "production" software to both commission and operate the delay-lines for their astronomical role has been identified. The team at the Cavendish Laboratory who have designed the delay-line system are now contracted to deliver this production software, the basis for the contract being the agreed Statement Of Work [RD1].

The delay line software provides real-time control, monitoring and data acquisition for the MROI delay line system [AD1]. The software also provides a data analysis capability for use in initial commissioning of the delay lines and infrequent alignment tasks.

The use cases for the production software can be summarized as:

- operation of the delay lines for the planned system commissioning tasks, i.e. the Performance Verification Milestones (PVMs) 1 to 13 [AD2];
- operation of the delay-lines for regular astronomical observations;
- acceptance tests of the second and subsequent production delay line trolleys.

The code to be delivered will be based on the prototype control and analysis software [AD3] that has been used to demonstrate the performance of the prototype delay-line trolley, but has been re-architected to integrate with the MROI supervisory control system (ISS) and to provide the functionalities needed for PVM testing and science operations.

Here we provide a high-level overview of the delay line control system, with pointers to other design documents that describe the components of the software in more detail. This document also highlights the key differences between the prototype and production versions of each software component.

1.1 Document Conventions

In this document we use the word **system** to refer to a *single* delay line as an entity, notwithstanding the fact that some hardware and software components are associated with multiple delay lines (see Fig. 3).

2 Delay Line System Overview

In order to set out the context for the production software, we first describe the complete delay line system and how it must be controlled.

The delay lines are required to introduce vacuum optical paths between zero and 380m using a single stroke, with an absolute precision of 10μm and jitter below 15nm over 10ms intervals.

The design architecture adopted for the MROI delay lines incorporates a single-stroke movable carriage carrying retro- reflecting optics and running in an evacuated aluminium pipe. A 3-d cutaway diagram showing the basic elements of the design is presented in Fig. 1. Further details of the additional hardware components for the full delay line system are shown schematically in Fig. 2.

The optical assembly utilises a standard configuration of a parabolic primary mirror with a flat secondary located at the M1 focus, and is itself supported on flexures (in an inverted pendulum geometry) mounted on a motorised carriage. The location of the cat's eye is monitored by a commercial Agilent laser metrology system where the only modification has been to expand the laser beam to approximately 20mm diameter to accommodate the long propagation path. However, the design also

incorporates a number of unusual features which are significant departures from the norm. The most important of these are:

1. The motorised carriage runs directly on the interior surface of the aluminium pipe that provides the vacuum vessel for the system.
2. Because the aluminium pipes used to define the trajectory of the carriage are themselves not guaranteed to be straight, the secondary mirror is adjustable in tip and tilt so as to correct for the shear introduced by any lack of pipe straightness. The shear is detected by picking off a fraction of the return metrology beam and sensing its location on a CCD camera. Even at slewing speeds, only a low control bandwidth is necessary because the length scales on which the vacuum pipe is "bent" are relatively large (i.e. many tens of cm).
3. The whole of the optical path in the MROI delay lines is kept under vacuum. By operating at a pressure of ~1 millibar, problems associated with internal seeing and longitudinal dispersion are minimized while still allowing sufficient heat transfer via conduction to take place between the delay line electronics chassis and the vacuum pipe walls.
4. Because rails are not used to define its trajectory, the delay line carriage is unconstrained in rotation about the vacuum pipe axis. This "roll" degree of freedom is controlled both by design — the center of gravity of the delay line carriage is below the vacuum pipe centerline — and also through an on-board tilt sensor which measures the roll angle in real time so that the undriven rear wheel of the carriage can be steered to correct any roll errors. As is the case for the shear variations, only low-bandwidth control (a few Hz) is necessary.
5. The distribution of power and control signals to the carriage is performed in a "contactless" manner. Power is transmitted inductively, using a wire lying in the bottom of the vacuum pipe which slides through a long transformer core attached to the bottom of the carriage. On-board power storage to manage periods of high load is also made available through the use of a bank of super-capacitors. Control signals to the carriage use two separate radio links. The first is a commercial 2.4 GHz wireless Ethernet link used for high data-rate communication between the on-board micro and external control computers, while the second is a dedicated low-latency 900 MHz link used to close the servo-loop controlling the cat's-eye position, e.g. when following astronomical targets.
6. The tube that defines the separation between the cat's-eye primary and secondary mirrors is made from carbon fibre and is designed to have a very low coefficient of thermal expansion. Thus only occasional axial translation of the cat's-eye secondary mirror is needed to compensate thermally-induced changes in the cat's-eye focus. To adjust the focus, a loop is closed around an encoder mounted on the secondary stage under control of the trolley micro.

2.1 Optical Path Delay Loop (Cat's-eye and Carriage Control)

There are two main modes of operation for axial control of the delay line. In either case the position of the cat's-eye is measured by the laser metrology system, and the relative position of the cat's-eye and the carriage is measured by an onboard sensor. However, the measurements are used in somewhat different ways in the two modes.

2.1.1 Tracking

This is the most critical mode, used when recording science data, and involves two stages to the servo loop:-

Cat's-eye

The cat's-eye position is measured by the laser metrology system and compared (by the VME CPU) with the current demanded position (interpolated from positions sent slightly in advance from the workstation). The resulting error signal is sent via the dedicated low latency RF link to the trolley (bypassing the onboard micro), where it is amplified and used directly to drive the cat's-eye voice coil actuator.

Two small additional signals derived from the cat's-eye/carriage relative position sensor are also applied, in hardware. The first one is a proportional term used to reduce the effective stiffness of the "wishbone" leg flexure pivots.

The second is a velocity term to offset dynamic drag caused by the voice coil. In addition electronic travel limits for the cat's-eye are set by the relative position sensor; these clamp the voice coil drive signal if the limits are exceeded.

Carriage

The carriage drive motor is controlled by the trolley CPU, using measurements from the relative position sensor, to keep the carriage centred under the cat's-eye so the "wishbone" legs are upright. This is important for best noise rejection. A demanded velocity term sent via the Ethernet RF link is added to reduce tracking error.

2.1.2 Slewing

This mode is used for rapid re-positioning of the trolley. The cat's-eye voice coil is driven directly by the relative position sensor to hold it fixed relative to the carriage. The drive motor can then be ramped up to full speed until the desired position is approached and then slowed before reverting to tracking mode. This is accomplished by the VME CPU reading the laser metrology and sending velocities to the trolley micro.

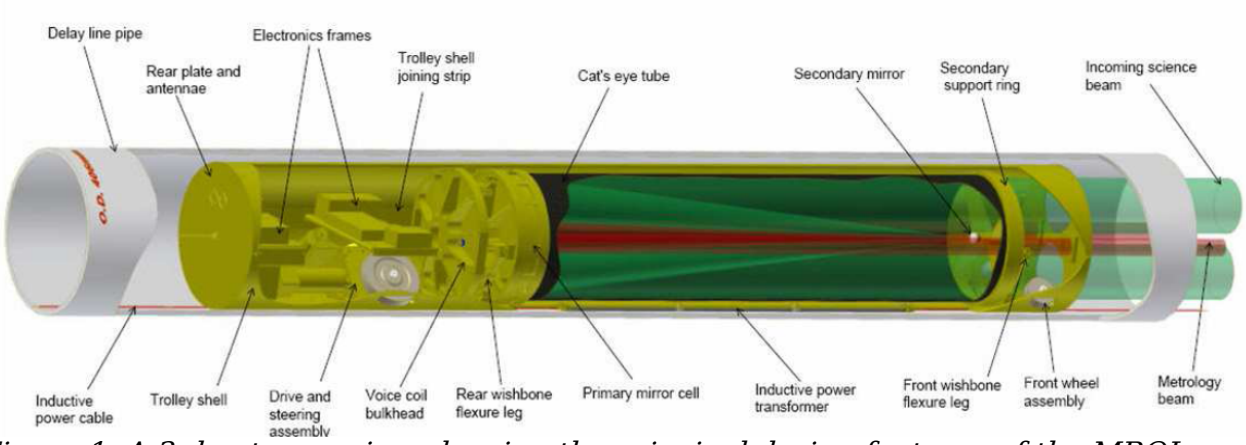


Figure 1: A 3-d cutaway view showing the principal design features of the MROI delay line system with the optical assembly mounted within its concentric mechanical carriage and the whole trolley running within an evacuated pipe. In this schematic the incoming science beam enters at right, is retro-reflected by the cat's eye, and returns to the right underneath the incoming beam. The input and output metrology beams enter and exit at right too, but remain in the same horizontal plane. The carriage motor, control electronics and communications and power systems are all located to the left of the delay line trolley, behind the primary mirror and voice-coil bulkhead. The supports carrying the vacuum pipe are not shown here.

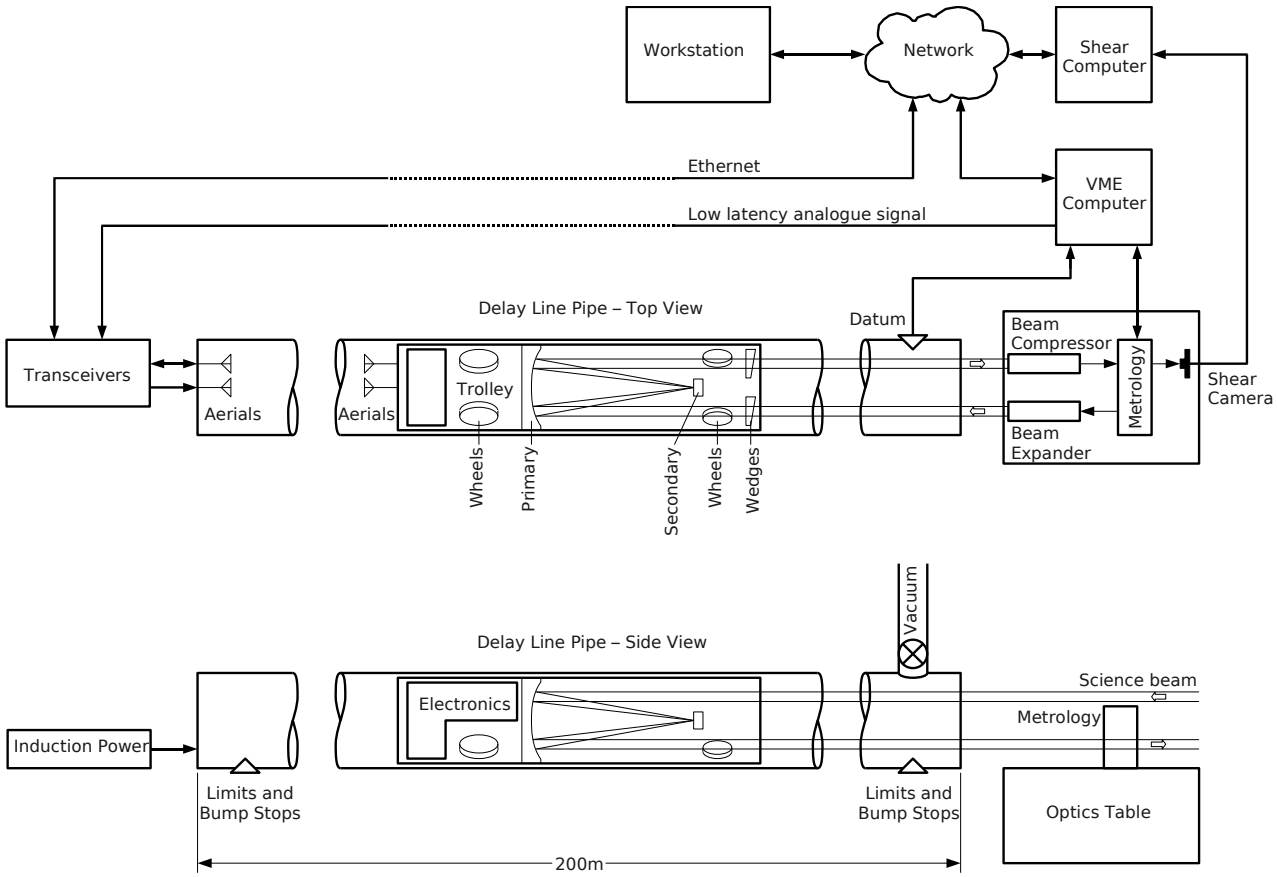


Figure 2: Schematic top and side views of a complete delay line system, showing the physical locations of the major components. The interfaces to the NMT-provided systems (ISS and FT) are not shown.

3 Control Software Architecture

The delay line control system is a distributed, event-driven system, comprising software running on the following computers:

- A “workstation” PC (shared between all trolleys) to act as a system controller, and provide a user interface for standalone testing of the delay line and interrogating delay line telemetry. In the production software this computer also accepts commands from the ISS and transmits monitor data to ISS data collector(s).
- A VME-bus CPU (shared between all trolleys) to read the metrology signal and hence control the cat’s-eye. In the production software this computer also applies position offsets sent by the MROI Fringe Tracker.
- A low-power PC104 single-board micro on each trolley, to control onboard functions with undemanding timing requirements, and to send telemetry to the workstation.
- A rack-mounted PC connected to each shear camera, to capture camera frames and compute shear corrections which are applied to the cat’s-eye secondary tip-tilt stage.

The actions of these computers are coordinated by means of a custom network messaging protocol, “dlmsg”, which is described in detail in RD3. This protocol defines several categories of message.

1. Commands:
   Sent by the workstation to subsystems in response to user input or system commands received from the ISS
2. (Command) Data:
   Information needed in real-time to close the servo loops described in Sec. 2
3. Status (transmitted from subsystem to the workstation):
   - Subsystem state information
   - Information to display for the user (including delay line performance metrics)
   - Command acknowledgements
   - Diagnostic log messages and fault notifications
4. Telemetry:
   Diagnostic information transmitted from subsystems to the workstation

The components of the production software are specified in the SOW [RD1]. The roles of each component are outlined in the sub-sections below, together with brief descriptions of the differences from the equivalent component of the prototype software. This material should be read in conjunction with Fig. 3 which illustrates how these components relate to the overall control architecture for the delay-lines.

3.1 Trolley software

The trolley software [RD4] resides on-board each delay-line trolley and interfaces to on-board hardware. This software is responsible for closing the roll loop, and applying correction signals from the shear sensor in order to close the shear loop. The trolley software also implements trolley actions commanded by the workstation such as changes in drive velocity and secondary focus.

The production trolley software is functionally identical to the prototype trolley software, and much of the existing code is expected to be carried over.

3.2 Shear sensor software

The shear sensor software [RD5] performs capture and centroiding of CCD images in order to provide closed-loop control of the delay line output beam shear.

The production shear sensor software adds one function to the prototype shear sensor software, the ability to transmit shear camera video to the workstation, for onward transfer to the ISS as monitor data (note that shear camera video will still be recorded directly by the shear sensor software, rather than by the workstion). This new capability will require an extension to the current telemetry message format; a proposed design for which is described in RD3. We expect the majority of the existing shear sensor code to be carried over to the production software.

3.3 VME metrology software

The metrology software [RD6] closes the fast position loops of the delay lines by sampling the Agilent metrology hardware and transmitting correction signals over the low-latency wireless links to the trolleys' cat's-eye drive electronics. The metrology system will incorporate a low-latency point-to-point Ethernet interface to the Fringe Tracker system for fringe tracking and fringe searching. This link will use the RTnet real-time Ethernet stack. The production metrology software will run under the Xenomai real-time Linux flavour, whereas the prototype metrology software runs under QNX.

The production metrology software will be a complete re-write of the protype software in order to support up to 10 delay lines and accommodate the change of real-time operating system. Some code from an existing metrology emulator (which implements inter-subsystem messaging for multiple trolleys) will be re-used.

3.4 Workstation software

The workstation software co-ordinates the overall delay-line system and manages communications with the MRO ISS. There are three separate application programs:

- a **system controller** [RD7] to provide supervisory control of the delay lines, responsible for generating trajectory demands for the metrology software and managing the state of the delay line subsystems. This will provide interfaces to the ISS, implemented using the C code-generation framework, for receiving delay line system commands and transmitting system and subsystem status and telemetry (including shear camera video) as monitor data;
- an **engineering control GUI** [RD7] with telemetry recording capability and live displays of system and subsystem status;
- a standalone **analysis GUI** [RD8] that reads telemetry recorded by the engineering control GUI. The production version of the analysis GUI will have enhanced functionality to support the defined use cases [RD2].

It will be possible to start and stop the engineering control GUI while the system controller software is running, and hence use the GUI's display and recording facilities as needed while the delay lines are being supervised by the ISS.

The workstation control software will be re-structured to create separate system controller and engineering control GUI applications, and to interface with the NMT-provided C code-generation framework for communicating with the ISS.

Open Issue: It will be possible to re-use a substantial fraction of the existing prototype C code, although re-coding in C++ may be desirable.

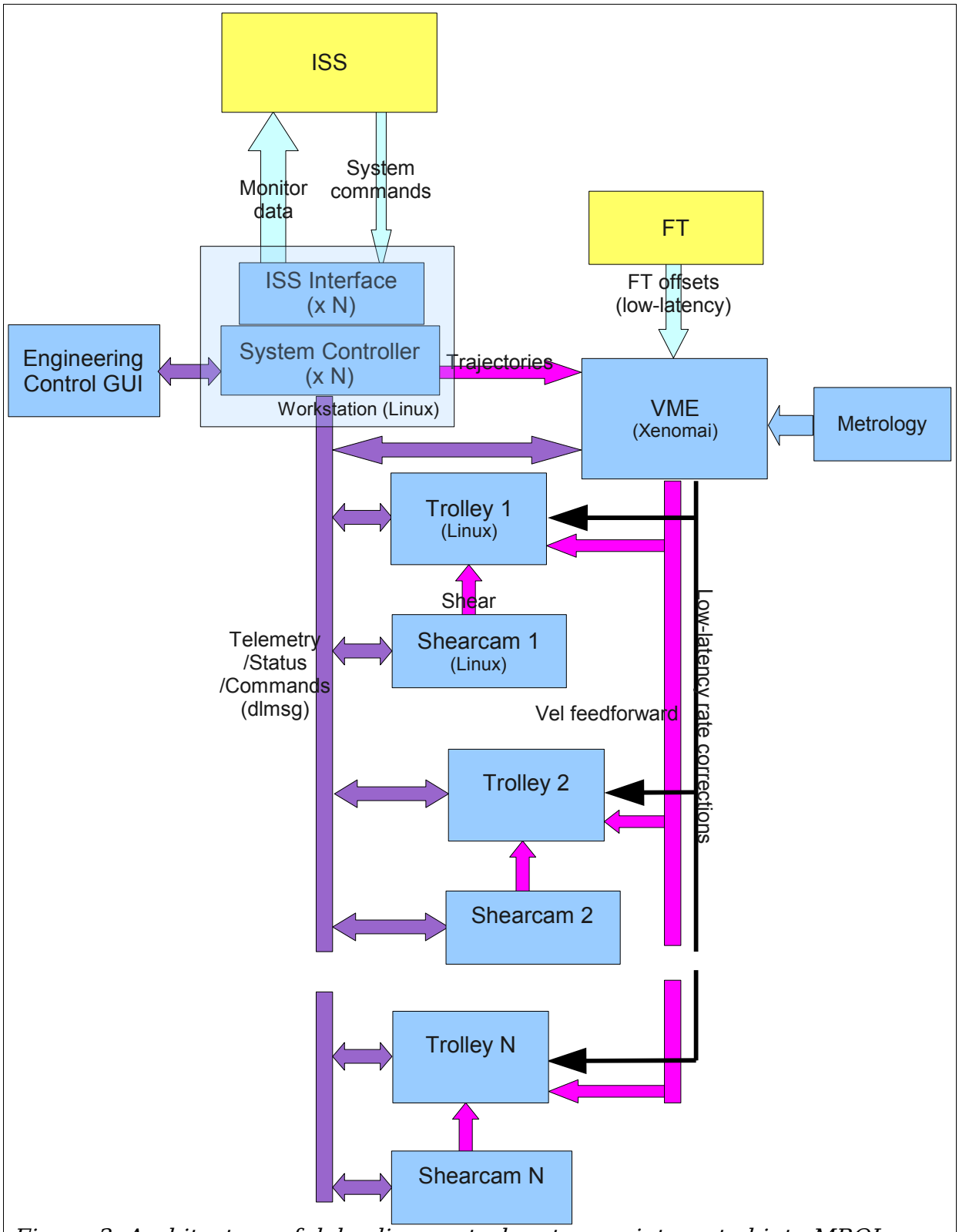


Figure 3: Architecture of delay line control system as integrated into MROI. Different coloured arrows refer to different types of communication (see text). The components shown in yellow correspond to NMT-supplied deliverables (which interface to the Cambridge software). Telemetry, status and command message flows are shown as purple arrows. Command data streams (labelled individually) are shown in magenta. The thin black arrows represent control signals transmitted via interfaces other than Ethernet; these signals are described in AD4.

3.5 Baseline solution software

This software component is used to refine an initial baseline model, using fringe acquisition data for a set of stars widely distributed across the sky. We anticipate that this will be supplied as a software library that can be integrated into the ISS by NMT, together with a simple user interface that can be used prior to this integration taking place. The baseline solution software will be described at the Final Design Review.

4 Delay Line System Implementation

From the perspective of the MRO Interferometer Supervisory System (ISS), the delay lines will appear as standard MROI systems (Fig. 4). This will be accomplished using a C code-generation framework being written by NMT that implements the custom TCP/IP protocols described in RD9. The facilities provided by the C framework will be integrated into the existing (mostly single-threaded) event-handling system used by the workstation software. An example of how we expect to do this is given at http://www.mrao.cam.ac.uk/~jsy1001/SomeSystem/html/

In addition to the supervised mode used for regular operations, the delay line system will also be able to operate in a standalone mode, independent of the ISS. In this mode the user will be able to command the delay lines using the supplied engineering control GUI.

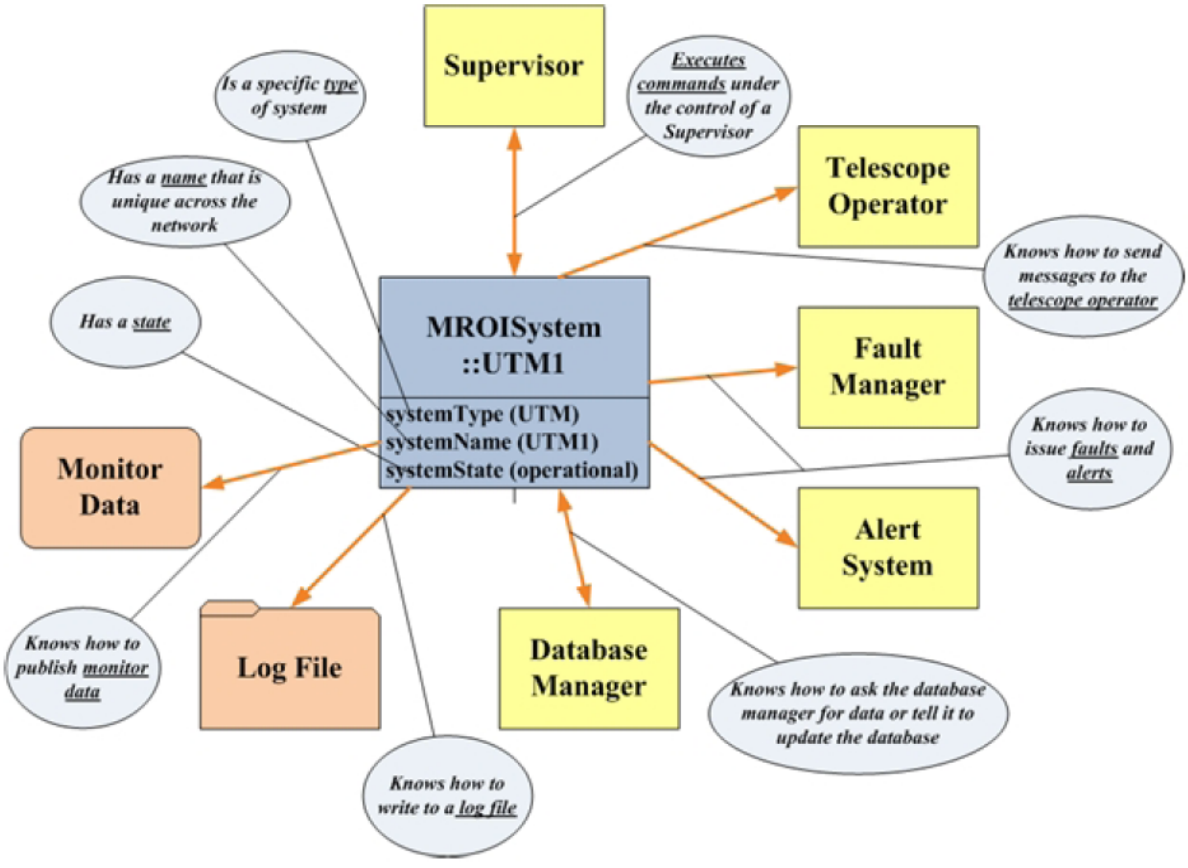


Figure 4: Diagram from RD9 showing the characteristics of a system that conforms to the MROI standard for interfacing to the ISS.

The delay line systems exposed by the workstation software will conform to the standard characteristics of MROI monitoring systems [RD9] as follows:

- The system type is “Delay Line.”

- Up to 10 instances may be created at startup, with unique names (TBD).
- All subsystems will obtain their configuration data by reading ASCII "key=value" format configuration files written by the ISS Executive during system startup (the same mechanism will be used to supply configuration data to the Unit Telescope control software), using data extracted from the MCDB.
- Any configuration parameters that are changed while operating the delay lines (e.g. the shear sensor fiducial point) will be saved to the MCDB by the workstation software, using the C system framework to communicate with the ISS Database Manager over a TCP/IP socket connection.
- The workstation system controller software will implement the standard state model (see next section), making use of the C system framework. The delay line system modes (FOLLOW, STOP/IDLE, DATUM, DIRECT SLEW) defined in RD2 will be sub-states of the operational state.
- The workstation software will accept commands sent by the ISS using the standard ISS TCP/IP-based protocols, making use of the C system framework. In general these commands will be handled as if they had originated from the workstation engineering GUI. A preliminary list of system commands is given in Sec. 5.
- The workstation software will transmit monitor data to the ISS according to the standard ISS protocols, making use of the C system framework. The available monitor points will be the unique items from the status and telemetry lists (for all delay line subsystems) defined in RD3. The workstation software will implement deactivation and decimation (see RD7) of individual monitor points.
- The workstation will receive log messages from all subsystems (including those generated within the workstation software). These logs will be written to a local file. Logs above a certain level (severity) will be transmitted to the ISS using its standard protocols, making use of the C system framework.
- The workstation will receive fault notifications from all subsystems (this includes faults generated within the workstation software). All faults will be transmitted to the ISS, using the C system framework and standard ISS protocols. For a small subset of the defined faults (to be discussed), the workstation will send an accompanying alert notification.

Open Issue: For a small subset of the defined faults, the workstation will be able to send an alert notification to the ISS. The delay line team would appreciate some further discussion of the kinds of fault which should generate an alert. Please note that there are no conditions that require operator intervention to prevent damage to DL hardware.

- We do not see any need to transmit messages to the Telescope Operator.

4.1 System Bootstrap

The bootstrap process is managed by the ISS Executive. The IP addresses of the delay line computers are retrieved from the MCDB by the Executive.

The first part of the process is sequenced by a Bourne Shell script which is run by the Executive. In this sequence, the SSH protocol is used to execute commands on remote systems and to copy files over the network.

- Executive ensures delay line workstation computer is powered up (perhaps by asking the operator to power it up if there is no remote boot facility)
- Executive retrieves workstation configuration from MCDB, writes ASCII

configuration file, copies file to delay line workstation

- Executive starts system controller process on workstation. Command line arguments are used to specify:
  - Which delay lines to start
  - The numbers of the main and data TCP/IP ports for each DL, on which to listen for connections from the ISS
  - Whether the metrology and/or shear sensor systems are present
- When the system controller process starts, the requested delay line systems are created as software objects and enter the UNDEFINED state
- DL systems automatically create server (listening) sockets on the main and data ports, and enter the STARTED state
- Executive ensures the delay line subsystem computers are powered up
- Executive retrieves subsystem configuration from MCDB, writes ASCII configuration files, copies files to delay line subsystems
- Executive starts processes on delay line subsystem computers, subsystems periodically attempt to connect to workstation

The delay line systems can now accept TCP/IP socket connections from the Executive. The Executive connects to the DL systems to complete the bootstrap process:

- Executive connects to DL systems on their main ports
- Executive commands DL systems to initialize
- DL systems enter the INITIALIZING state, in which:
  - Workstation opens server socket to listen for connections from delay line subsystems
  - Hence subsystems succeed in connecting to workstation
  - Once the metrology system has connected to the workstation, the workstation opens a command data connection to the metrology system for transmission of trajectory data
- Once all of the inter-subsystem connections needed to operate a particular delay line have been established, that system enters the INITIALIZED state
- Executive initiates collection of delay line monitor data by ISS Data Collector(s)
- Executive commands DL systems to enter the OPERATIONAL state

If an inter-subsystem socket connection is subsequently broken, a fault notification will be sent to the ISS. Reconnection will be attempted automatically.

5 Preliminary system command list

The following commands can be sent from the ISS or the engineering control GUI to the workstation system controller object managing a particular delay line.

Command	***Parameters***	***Description***
SiderealFromCat	Float64[10]	Follow sidereal trajectory for baseline [bc,bx,by,bz] and target [ra,dec,pmra,pmdec,plx,rvel]
ConstAccel	Float64[4]	Follow trajectory through {[position], [velocity], [time]} at constant [accel]

Command	***Parameters***	***Description***
ConstAccelJoin	Float64[1]	Continue from current trajectory at constant [accel]
ConstVel	Float64[3]	Follow trajectory through {[position], [time]} at constant [velocity]
ConstVelJoin	Float64[1]	Continue from current trajectory at constant [velocity]
Position	Float64[1]	Follow fixed position trajectory at [position]
SetOffset	Float64[1]	Set intra-night offset to [value]
Idle	none	Set system mode to STOP/IDLE: Stop OPD loop and zero carriage velocity
Datum	none	Slew trolley to datum and zero metrology
DirectSlew	Float64[1]	Slew trolley using command direct from workstation at [velocity]
SteeringOn	none	Turn steering servo on
SteeringOff	Float64[1]	Turn steering servo off and move to [angle]
TipTiltOn	none	Turn tiptilt servo on
TipTiltOff	Float64[2]	Turn tiptilt servo off and move to [X] and [Y]
SetShearFiducial	Float64[2]	Set shear sensor zero position to [X] and [Y]
FocusPos	Float64[2]	Move focus to [position] until [timeout]
FocusOffset	Float64[2]	Move focus [distance] from current position until [timeout]
FringeTrackOn	none	Apply FT offsets
FringeTrackOff	none	Don't apply FT offsets
ResetFTOffset	none	Zero total FT offset
LogVideoOn	Int16[1]	Activate shear video logging, saving one image in every [number]
LogVideoOff	none	Deactivate shear video logging