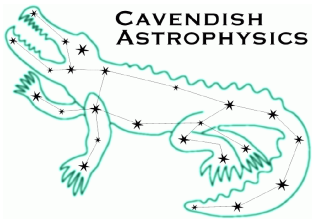
MRO Delay Line

VME Software Functional Description

The Cambridge Delay Line Team

rev 0.3

11 July 2007



Cavendish Laboratory
JJ Thomson Avenue
Cambridge CB3 0HE
UK

Objective

To describe the overall architecture of the metrology system software running on the VME CPU.

Scope

This document describes the architecture and functional behaviour of the prototype control software running on the VME system. The role of this sub-system is to close the delay line metrology loop by sending a correction signal to the cat's-eye over the low-latency link, the correction being obtained by comparing the measured cat's-eye position from the metrology system with a demanded trajectory received from the workstation.

This document should be read in conjunction with:

- “System Overview”, which provides a high-level description of the metrology loop and explains how the cat's-eye is controlled on-board the trolley
- “Metrology System and VME Hardware Design Description”, which describes the hardware used to measure the cat's-eye position
- “Workstation Software Functional Description”, which describes how the trajectory demand is calculated and transmitted to the VME system

1 Introduction

The VME system supports the following (axial) system modes of the delay line (these are introduced in the “System Overview” document). We now outline the behaviour of the VME system in these modes.

1.1 Stop/Idle Mode

In this mode the trolley is stopped by the workstation sending a **DirectSlew** command with a demand velocity of zero to the on-board trolley micro. This activates the cat's-eye local loop which locks the cat's-eye in the middle of its travel. The VME system is not required to do anything in this mode.

1.2 Datum Mode

In this mode the VME system transmits a sequence of slew velocities to the trolley micro in order to precisely position the cat's-eye at the datum switch. The metrology count is then reset.

1.3 Follow Mode

In follow mode, the VME sub-system receives a 10 Hz-sampled trajectory (position and velocity) demand from the workstation, interpolates it to 5 kHz sampling, and compares this with the actual cat's-eye position obtained from the laser metrology system. Within the follow system mode, the VME/trolley micro combination may be in either track or slew mode as determined by the VME CPU:

1.3.1 Track Mode

If the position error is within fixed limits, the VME system switches the trolley to track mode (which deactivates the cat's-eye local loop) and closes the metrology loop by outputting a control voltage to the cat’s eye via the low latency link. The current velocity demand is transmitted to the trolley micro so that the carriage follows the cat's-eye motion.

1.3.2 Slew Mode

If the position error exceeds track mode limits, the trolley is switched to slew mode (which activates the cat's-eye local loop, locking the cat's-eye in its centre position). The VME system calculates a velocity based on the magnitude of the position error, which is sent to the trolley in order to reposition it so that eventually track mode will be activated.

2 Metrology Hardware

The metrology hardware is described fully in the document “Metrology System and VME Hardware Design Description”. We include an outline description here to help clarify the interaction between the metrology hardware and software. The metrology system is based on a commercial laser interferometer – the Zygo displacement measuring system (ZMI 2000). and the reader is referred to the Zygo web site (www.zygo.com) for more technical details and the principle of operation.

The hardware consists of:-

- Laser head (7702),
- Linear interferometer (7002),
- Retroreflector(7003),
- Various mirrors,

- Beam expanders,
- Fibre optic pickup,
- Measurement board (2002),
- Datum switch (Sony SET-P15-1).

A schematic layout is shown below in Figure 1. Optical fibres connect the receiver to the measurement board and connect the laser to the reference input on the measurement board.
The datum switch is a magnetic sensor operated by a magnet on the trolley. This is connected to a electronic latch which can be reset.
The VME chassis contains:-

- Processor (Concurrent Technologies CPU - x86 with Ethernet port, keyboard, mouse, VGA port, etc),
- Timing board (Symmetricom TTM635VME),
- Measurement board (Zygo 2002 as listed above),
- Industry pack carrier board with DAC and PIO,
- Interface board.

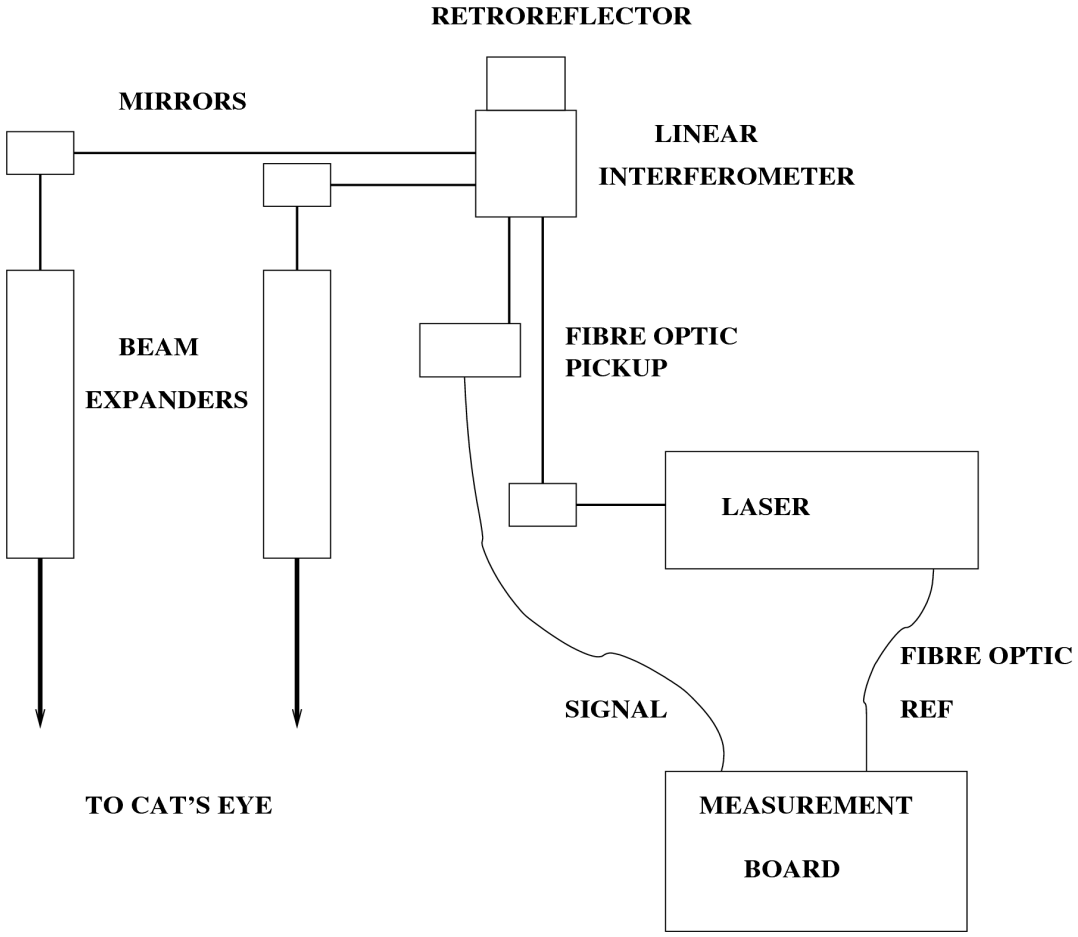


FIG 1
Figure 1: Schematic layout of the metrology system hardware

3 Development and Execution Environment

The operating system is QNX 6.1. This was chosen because it is a real time operating system with a fast, deterministic interrupt response time and also because we have previous experience with QNX at COAST.

The interrupt performance is essential in optical interferometers as the cat's eye must be positioned and track to sub-micron accuracy. At maximum tracking speed of 15mm/sec a timing error of 10μsec translates to an error of 0.15 microns in cat's eye position. QNX is UNIX-like and follows various industry standards such POSIX and TCP/IP. The system is used to both develop and run the software, which is coded in C.

4 Program Architecture

The software is divided between several threads or processes which run concurrently. One thread deals with transmission of messages to the workstation and trolley micro; another deals with interrupts and there are two threads to deal with command and command data messages received from the workstation. The advantage of using threads is that if there is no interrupt/message than the other threads are free to run but immediately an interrupt occurs or message arrives that thread will run.

4.1 Main program

When the main program starts it sets up a common data area for use in the interrupt routine, initializes VME hardware, instantiates status and telemetry objects, and establishes a socket connection with the trolley in order to send command data messages. The various threads are then started: two lower-priority threads to deal with command and data messages respectively, and a higher-priority thread to handle interrupts. Note that eventually there would also be another thread to handle communication with the fringe tracker. The main program thread then loops, sending messages when the appropriate intervals have elapsed, as follows:

- Status messages are sent to the workstation at 0.1s intervals.
- Command data messages are sent to the trolley at 0.1s intervals. These are either **Slew** or **Track** messages depending upon the value of a flag set by the interrupt routine. Both message types contain a demand velocity for the carriage. In track mode this is equal to the demand velocity received from the workstation, whereas in slew mode the velocity is calculated by the VME system on the basis of the current position error.
- Telemetry messages are sent to the workstation at 1 second intervals.

The operation of the main program thread is pre-empted by the other threads described below.

4.2 Interrupt Routine

The interrupt routine comprises two parts: the interrupt service routine (ISR), which runs when a hardware interrupt occurs, and a second part which runs when the ISR returns.

The ISR uses the complete resources of the processor and therefore runs deterministically so that a minimum time elapses between sampling the measurement board and outputting a drive voltage to the cat's eye. Floating point operations are not available in the ISR and so all arithmetic is done using integers, performing carries explicitly where necessary.

The timing board generates a signal at 5 kHz rate which samples the Zygo measurement board and is also connected to the "SYSFAIL" signal on the VME bus. This method of generating the CPU interrupt was chosen, rather then a VME bus interrupt, because the "SYSFAIL" signal is fed directly to the processor interrupt lines and so gives minimum response times.

The ISR immediately reads the measurement board, then compares the metrology value with the previous value to check if the measurement board has overflowed[1]. The routine then calculates a position error by

1 The Zygo measurement board is designed to count ±21.2 m and has 36bit storage so for 200 metres an

subtracting the position demand for the time the interrupt occurred (inferred by counting interrupts) from the measured position. If fringe tracking is enabled, the tracking offset is added to the the position demand before calculating the error. Provided the delay line is in follow mode, the subsequent ISR behaviour depends on the magnitude of the position error:

- If the error exceeds fixed limits, slew mode is flagged, causing the main thread to send **Slew** messages to the trolley micro
- Otherwise, track mode is flagged, causing the main thread to send **Track** messages to the trolley micro. The position error is converted into a rate demand by adding the product of the demand velocity and the interrupt period (0.2ms) and the result transferred to the DAC for transmission to the cat's-eye

If a command to go to the datum has been received the measured position of the cat's eye is ignored and the trolley is switched to slew mode and driven to the datum position as described in the next section.

Finally the demanded position and velocity for the next interrupt are calculated. This is done by interpolating the trajectory demand received from the workstation assuming a constant velocity over each 0.1s interval between supplied values (this is sufficiently accurate given the small accelerations required to follow a sidereal trajectory).

If the interrupt is coincident with a GPS second tick, the interrupt count is reset and a "second" flag is set.

The following operations are performed when the ISR returns:-

- Telemetry data are stored in a buffer for use by the main program thread
- Every tenth of a second, a new demand position and velocity are read from the received message buffer for action by the next ISR
- If the "second" flag is set:
    - The "telemetry buffer full" flag is set indicating that this block of data should be sent to the workstation and the storing of telemetry data is switched to a second buffer
    - The appropriate set of demand positions and velocities to use in the subsequent second are selected based on the time stamps of the buffered messages

4.3 Datum Routine

The datum routine is entered on receipt of a command from the workstation. This is necessary at start up and if the metrology loses count for any reason.

The VME system commands the following sequence of operations in order to seek the datum:-

1. Reset the datum latch circuit
2. Slew at maximum speed towards datum
3. Stop the trolley when the datum latches
4. At this point the trolley will be past the datum switch
5. Reset the latch and drive slowly back until the datum latches
6. Continue driving the trolley until clear of the datum switch

---

extra 4 bits are required which are added in software. The board is programmed to carry on counting after an overflow has occurred.

7. Drive very slowly towards datum until switch operates
8. Reset metrology counter and re-close cat's eye servo loop

If the trolley is already past the datum when commanded to go to the datum then the trolley will drive into the pre-limit (see "Limits Design Description"). The workstation will then drive the trolley out beyond the datum by a direct slew command to the trolley micro and then command the VME to restart the above sequence.

4.4 Message Routine

The format for sending and receiving messages has been described fully in the document "Control Software Architecture".

Received command messages are checked for which trolley they apply to and the appropriate flag set.

In the case of received command data messages, each message is stored in a circular buffer that can store up to ten messages. Messages are sent by the workstation at 1Hz and each contains ten positions and velocities, plus timestamp information specifying when the positions/velocities should be realised. When a "second" event is detected in the ISR, the code that runs when the ISR returns checks the time stamps of the messages in the buffer and selects the appropriate set of positions and velocities to use in the subsequent second. Hence data can be sent sometime in advance and performance will not be affected by network delays.