MOUNT CONTROL SYSTEM - CONTROL SYSTEM DESIGN DESCRIPTION

John Wilkes and Chris Carter

Royal Greenwich Observatory
Madingley Road
Cambridge
CB3 OEZ

ISSUE : 3

14 January 1997

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. PURPOSE

This document is a deliverable of the critical design phase of the Gemini mount control system work package. Its purpose is to formally describe the proposed design of the hardware and software that implement the mount control system functions.

The audience of this document is:
- The CDR reviewing panel.
- Individuals working on subsequent phases of the project.
- Individuals working on connected Gemini work packages

ISSUE	DATE
For Review	25 October 1995
1	15 December 1995
2 (For Review)	25 November 1996
3	14 January 1997

Table 1.1 - Issue Record

1.2. SCOPE

The mount control system provides the basic ability to slew and track the telescope. It also interfaces to a number of secondary systems that are required to provide services. The MCS is intended as an engineering interface to the mount and its subsystems - it is intended that the telescope could be run from here for initial set-up and engineering work. It is not intended as an interface where the telescope would meet specification.

1.3. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

The following is a list of abbreviations used in this document and related MCS documents.

A&G	Acquisition and Guiding
ABS	Polyacrylonitrile-butadiene-styrene
ADC	Analogue to Digital Converter
ADR	Assembly Design Review
ATP	Acceptance Test Plan. This consists of a Test-Procedure Specification as defined in IEEE Std 829-1983.
BNC	British Naval Connector
CAD	Command Action Directive
CAN	Controller Area Network
CAR	Command Action Response
CDR	Critical Design Review
CPU	Central Processing Unit
CRCS	Cassegrain Rotator Control System
CSDD	Control System Design Description
CSS	Control System Simulator
CW	CounterWeights
DAC	Digital to Analogue Converter
DC	Direct Current
DHS	Data Handling System
DPRAM	Dual Ported Random Access Memory
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
dSPACE	digital Signal Processing And Control Engineering (German company supplying the HWILS system)
DXF	Drawing eXchange Format. File format used by AutoCAD, ORCAD etc.
e.g.	for example

ECC	Error Checking and Correction
ECS	Enclosure Control System
EMI	Electro-Magnetic Interference
EPICS	Experimental Physics and Industrial Control System
EPLD	Erasable Programmable Logic Device
etc.	et cetera - ‘and the rest’, ‘and so on’
FCS	Functional Control System
FDE	Friction Driven Encoder
FEA	Finite Element Analysis
FST-2	A two board FAST amplifier by Kollmorgen Inland Motor. FAST - Flexible Amplifier Servo Technology)
GIS	Gemini Interlock System (formerly ISS - Interlock Safety System)
GPO	Gemini Project Office
GPS	Global Positioning System
GUI	Graphical User Interface
H/W	Hardware
HWILS	Hardware In the Loop Simulation
Hz	Hertz
i.e.	that is
I/F	Interface
I/O	Input / Output
I/P	Input
ICD	Interface Control Document
ICS	Instrument Control System
IDD	Interface Design Document
IGPO	International Gemini Project Office
IOC	Input Output Controller
IP1	Implementation Phase 1
IP2	Implementation Phase 2
IP3	Implementation Phase 3
IRIG-B	Inter-Range Instrument Group (format B)
ISS	Instrument Support Structure
ITB	Inner Topple Bracket
kHz	One thousand hertz.
kN	One thousand Newtons.
LAN	Local Area Network
LSB	Least Significant Bit
LVDT	Linear Variable Differential Transformer
M1	Primary Mirror
M2	Secondary Mirror
MCS	Mount Control System
MEC	Mount Engineering Console
MHz	One million hertz.
ms	millisecond (One thousandth of a second).
MSB	Most Significant Bit
NCB	Node Control Board
NCU	Node Control Unit
O/P	Output
OCS	Observatory Control System
OPI	OPerator Interface
OSS	Optical Support Structure
OTB	Outer Topple Bracket
PA	Power Amplifier
PC	Personal Computer
PCB	Printed Circuit Board
PCS	Primary (Mirror) Control System
PDR	Preliminary Design Review
PLC	Programmable Logic Controller
PLD	Programmable Logic Device

PMAC	Programmable Multi-Axis Controller
PRS	Package Requirements Specification
PSU	Power Supply Unit
PTP	Package Test Procedure. This consists of a Test-Design Specification and a Test-Case Specification as defined in IEEE 829-1983.
PVT	Position Velocity Time - a PMAC mode of operation, see reference [4], page 3-125.
rad.	radian (Angular unit of measurement)
RAM	Random Access Memory
RGO	Royal Greenwich Observatory
RMS	Root Mean Square
RPM	Revolutions Per Minute
RTD	Resistance Temperature Detector
Rx	Receive
S/W	Software
SAD	Status and Alarms Database
SDD	Software Design Description
SDR	System Design Review
SIC	Standard Instrument Controller - WP that described generic Gemini hardware. See reference [5].
SIR	Status Information Record
SISO	Single Input - Single Output
SMCS	Secondary Mirror Control System
SRS	Software Requirements Specification
STP	Screened Twisted Pair
SW	Service Wraps
TAI	International Atomic Time (In French !)
TAV	Tacho AVeraging (circuit)
TBD	To Be Determined
TBEG	Telescope Building and Enclosure Group
TCS	Telescope Control System
TF	Transfer Function
TI	Texas Instruments
TTL	Transistor Transistor Logic
Tx	Transmit
VAT	Value Added Tax
VCC	Velocity Command Combining (circuit)
VE	Virtual Encoder
VME	Versa Module Eurocard (The meaning is obsolete)
WP	Work Package
WPD	Work Product Document
wrt	with respect to

1.4. REFERENCES

[1] REV-C-G0036 “Mount Control System Design Review Report”
[2] “Mount Control System Software Design Description” Andy Foster
[3] # 9414257-GEM00067 “Telescope Requirements Document”
[4] Delta Tau Data Systems “PMAC User’s Manual & Software Reference”, Version 1.13 PLUS addendum (Versions V1.14 and V1.15) and relevant accessory manuals.
[5] “The Gemini Standard Controller Hardware Documentation” Andrew Johnson
[6] TC-C-G0022 “Gemini Non-linear Servo Simulation” Mike Burns and John Wilkes
[7] MCSCJC08 Revision 1.1 “MCS VME Crate Wiring Schedule” Chris Carter
[8] TN-C-G0040, MCSJDW10 Issue 3 “Prediction Of Servo Error Using Simulink Model” John Wilkes
[9] TN-C-G0041, MCSJDW11 Issue 4 “Encoder Tests” John Wilkes
[10] MCSJDW12 Issue 2 “Tape Encoding System Requirements Specification” John Wilkes
[11] MCSJDW14 Issue 1 “Main Axis PMAC Set Up” John Wilkes
[12] MCSJDW16 Issue 1 “Counterweights And Service Wraps PMAC Set Up” John Wilkes

[13] "Determination and correction of quadrature fringe measurement errors in interferometers" Peter L. M. Heydemann, APPLIED OPTICS Vol. 20 No. 19, 1 October 1981

[14] "Optical fringe subdivision with nanometric accuracy" K. P. Birch, PRECISION ENGINEERING Vol. 12 No. 4, October 1990

[15] PCSPM01 "Primary Mirror Control System: Node Box Software Design Description" Paul Martin

[16] PCSJFM04 "PCS IOC/NCU CAN Protocol" John Maclean

[17] IEEE Std 1016-1987 "IEEE Recommended Practice for Software Design Descriptions"

[18] MCSJDW09 "MCS Preliminary Design Review Report" John Wilkes

[19] "Mount Control System - Control System Design Description" Issue 1, John Wilkes and Chris Carter

[20] "Subsystem Circuit Diagrams" - A collection of '.DXF' files containing miscellaneous MCS circuits, including:

- FIDUCIAL.DXF
- OVP_CCT.DXF
- EL_TAV.DXF
- AZ_TAV.DXF
- EL_VCC.DXF
- AZ_VCC.DXF

[21] "Monitoring and Metrology Subsystem Circuit Diagrams" MM_*.DXF, a collection of '.DXF' files containing the monitoring system circuits

[22] Kollmorgen Inland Motor "FAST Drive, Flexible Amplifier Servo Technology, User's Manual" MN-223 REV (Preliminary 2/3/95)

[23] PCSJFM10 "C167 Node Control Board Hardware Manual" John Maclean

[24] 87-GP-1002-0000 Altitude Cable Wrap Drawing Set

[25] 87-GP-1001-0000 Azimuth Cable Wrap Drawing Set

1.5. OVERVIEW

1.5.1. Philosophy Of Critical Design

In its truest form, a critical design of a given product should be sufficient information to allow the product to be built and operated to fulfil all given requirements. In the case of the software components of a product, this information normally *is* the finished product. It is therefore sensible to split a product into two parts, hardware and software, and treat each differently for the purposes of a critical design. This has been done in the MCS.

SOFTWARE

This includes the EPICS software (see reference [2]) and the lower level software items such as the PMAC user written DSP code and programmable set up.

The critical design of the software components shall include completed algorithms of how the software shall be implemented and a subset of the actual code.

HARDWARE

All elements of hardware shall be completely defined by circuit diagrams, wiring schedules, layouts and packaging.

There is some exception to this with regard to the tape encoder interface electronics since this was a late addition to the MCS work package, the actual circuit diagrams of the head interfaces have not yet been defined.

1.5.2. Format Of An Control System Design Description

The format of this document is based loosely upon reference [17], modified to take into account that this document is mainly concerned with the description of hardware rather than software.

The design of the control system is broken up into a number of subsystems and each has a set of *attributes*. The values given to these attributes form the design of the control system. The following attributes have been identified as relevant to a control system design:

- Name
- Purpose - Or Requirements.
- Subordinates - Can the subsystem be broken down into still smaller subsystems.
- Internal Dependencies - Relationships with other subsystems.
- Internal Interfaces - How the subsystem communicates with other subsystems.
- External Dependencies - Or Resources. External systems that are required by the subsystem.
- External Interfaces.
- Function - Or Design Detail. What the system does and how it does it.

Each subsequent section of this document presents a *design view* of the MCS control system. Each design view defines of a subset of the subsystem attributes.

Section 2 presents the *Decomposition Description* design view. This outlines how the MCS is organised internally and how it fits in with the rest of the Gemini Control System. The Name, Subordinates and Internal/External Dependency attributes are defined in the Decomposition Description.

Section 3 presents the *Detail Description* design view. For each subsystem, the requirements, as specified in the MCS PRS (see reference [1]), are restated. The detail of how these requirements are to be met is then explained.

Section 4 presents the *Interface Description* design view. This presents the hardware and software Internal and External Interfaces.

2. DECOMPOSITION DESCRIPTION

This section describes the role of the MCS within the Gemini control system, and the dependencies upon external systems. Also described is the partition of the MCS into a number of functional subsystems.

2.1. SYSTEM PERSPECTIVE

The MCS is part of the Gemini Telescopes Project. It is responsible for the interface between the telescope computer system and the mount hardware. It exists alongside a number of other control systems that carry out a similar task in other areas of the telescope. Figure 2.1 shows, in block diagram form, how the MCS fits into the overall Gemini control system.

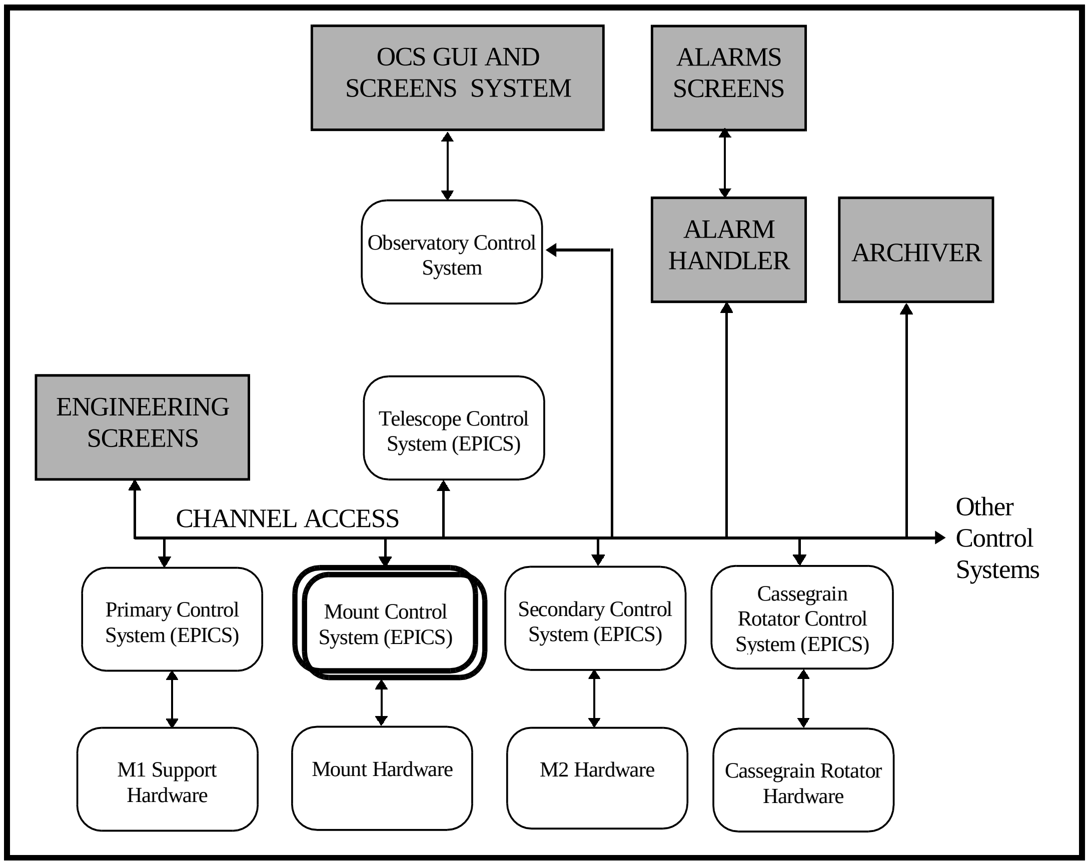

OCS GUI AND SCREENS SYSTEM
ALARMS SCREENS
Observatory Control System
ALARM HANDLER
ARCHIVER
ENGINEERING SCREENS
Telescope Control System (EPICS)
CHANNEL ACCESS
Other Control Systems
Primary Control System (EPICS)
Mount Control System (EPICS)
Secondary Control System (EPICS)
Cassegrain Rotator Control System (EPICS)
M1 Support Hardware
Mount Hardware
M2 Hardware
Cassegrain Rotator Hardware

Figure 2.1 - Relative Position Of Mount Control System

2.2. DEPENDENCIES

Figure 2.2 shows how the MCS is broken up into subsystems and the relationships between those subsystems. The shaded blocks indicate the MCS functions which will be implemented with a mixture of hardware and software. The software, indicated by the circles in Figure 2.2, is to written using VxWorks, C and EPICS and run on a Motorola VME 68000 based computer. The design of this software is the subject of a separate document, reference [2].

The remainder of this document presents the proposed design of the MCS hardware, indicated by squares in Figure 2.2.

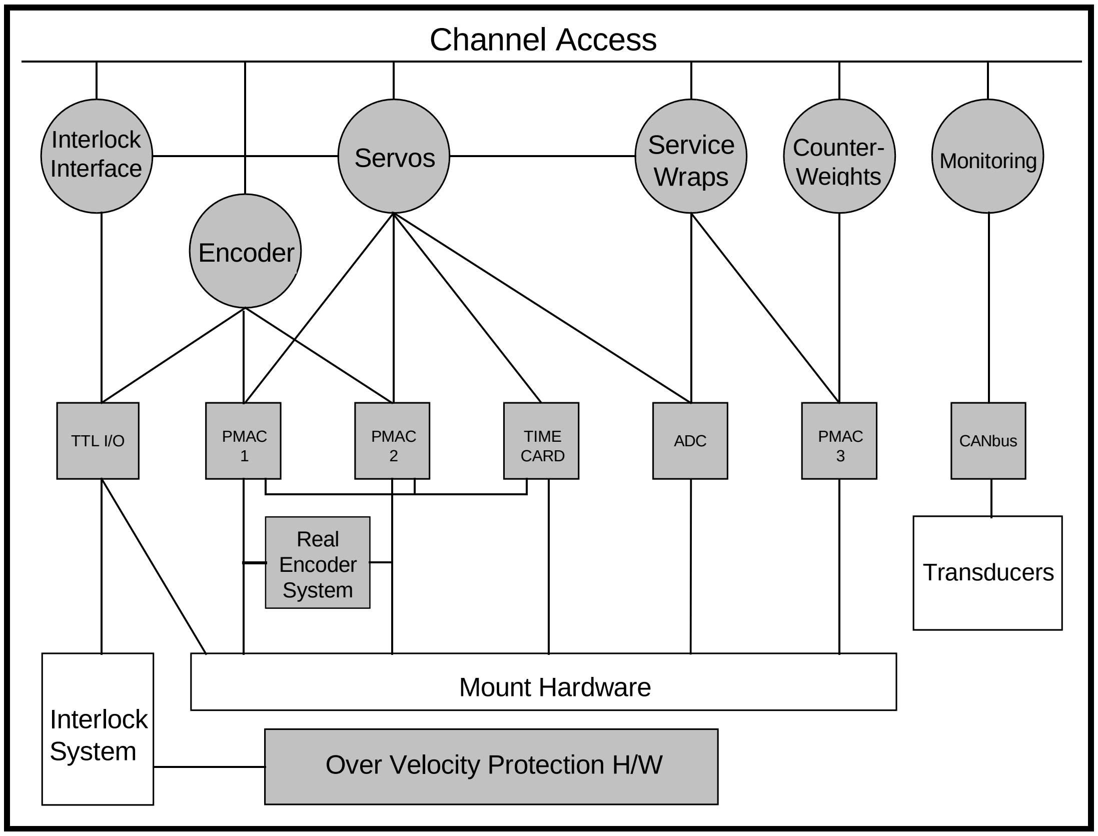


Figure 2.2 - MCS Internal Dependencies

2.3. GENERAL CONSTRAINTS

- The MCS will be implemented in a dedicated VME crate located near the mount base.
- The VME crate control CPU (MVME-167 M68040) will run the VxWorks operating system. The MCS software will be implemented using the EPICS database system.
- There will be a two-headed Sun Workstation (not provided by the MCS work package) permanently connected to the mount control VME crate via ethernet in order to control the system from the enclosure floor. A light-tight cover will be provided for this console so as not to impact the required enclosure darkness during observing.
- The serial port of the VME CPU will be monitored in order to provide diagnostics during system initialisation. This will be done with a dedicated VT100 type serial terminal or (more likely) with a terminal emulator running on the sun workstation.
- The MCS WP is only responsible for specifying the cabling to/from the MCS IOC along with the specific connectors to be used. It is not a requirement of the MCS WP to provide length and routing information, this is the responsibility of the Gemini Systems Group.
- No signal conditioning or transducers are to be provided by the MCS WP. The Monitoring subsystem is simply an ADC function.
- The power amplifiers, drives and encoders for the service wrap-ups are to be provided by a separate WP.
- The hardware used shall be based on the work products of the *Standard Instrument Controller* by Andrew Johnson/RGO (see reference [5]).
- There shall be an additional two spare slots within the VME card frame for the possible addition of a high speed reflective memory bus.

2.4. HARDWARE REQUIREMENTS

The Gemini Mount Control System is based around one 21-slot VME crate that will be installed at the telescope mount base; two crates are being prepared for the Hawaiian and Chilean telescopes (known as Gemini North and Gemini South respectively).

The table below details the VME cards required to populate one crate.

Quantity	Description	Purpose	Width (Slots Occupied per card)	Part Number
1	Motorola processor card	*Runs the VxWorks operating system and hosts the EPICS environment*	1	MVME167–33B (33MHz MC68040 with 16MB ECC DRAM)
1	Xycom TTL I/O card	*Provides digital I/O capability*	1	XVME–240
3	Delta Tau PMAC VME card	*Implements servo control of the telescope axes and other subsystems. (1 azimuth, 1 elevation, 1 counterweights & service wraps per telescope)*	2	PMAC VME (8 axis, 60MHz, dual-ported RAM)
5	Delta Tau A/D Converter Accessory Board	*Provides 4 channels of A/D conversion. Used with the Heidenhain tape system and the LVDTs.*	None	ACC28
1	Delta Tau I/O Expansion board (Option 3: Port A, Port B; 24 latched high-true TTL inputs)	*Provides digital I/O capabilities. Used for fiducial and absolute encoder interfaces. (1 per PMAC card)*	1	ACC14V (Option 3)
1	CANbus Industry-pack carrier card (plus one TIP-810 Industry pack)	*Provides a CANbus interface for the Monitoring subsystem*	1	VIPC-610 (plus TIP-810)
1	Datum Inc. / Bancomm time card (IRIG-B I/O)	*Provides a highly stable time reference*	1	BC635 VME
2	Xycom Analogue (A/D) Input module	*General purpose analogue input capability*	1	XVME–566

Table 2.1 - Hardware Requirements

See reference [7] for details of VME slot allocation.

3. DETAIL DESCRIPTION

3.1. MAIN SERVOS SUBSYSTEM

3.1.1. Introduction

There are two servo systems needed to point and track the telescope line of sight - elevation and azimuth. A third axis, the Cassegrain rotator, is needed to maintain field orientation and is the subject of a separate work package under the control of the Instrument Group.

The mechanics of the drive (power amplifiers, motors, tachos, encoders etc.) have been defined by the TBEG. Both the azimuth and elevation axes employ friction-type drive systems to supply the required torque for all telescope slewing, tracking, and offsetting operations. To simplify design and manufacture, the two drive systems use identical motor housings and motors (i.e. identical rollers, bearings, motors, and housing design). Each unit is hydraulically pre-loaded against the drive track. The azimuth drive consists of four drive units and the elevation drive consists of two drive units, one either side of the tube. Each drive unit consists of two motors. A fuller description of the drive configuration is given in reference [3].

3.1.2. Purpose

The following requirements shall be met by the MCS hardware.

The elevation and azimuth servo systems shall accept position information from the encoder subsystem at the servo rate.

It shall be the responsibility of the hardware to close the servo loops.

The hardware of the servo subsystem shall be implemented using a digital signal processor programmed with suitable servo algorithms.

The sampling frequency of the servos (the servo rate) shall be sufficient to allow the servo algorithms to be effective. This is expected to be at least 1 kHz.

The servo subsystem shall implement a scheme to reduce the available torque demand to the motors in the event, or possibility, of drive slippage.

The servo subsystem shall provide a demand signal proportional to the required velocity to each of the drive amplifiers at the servo rate. The drive amplifiers are Inland FST-2; there is one per motor and each implements a velocity loop using feedback from the tacho-generators.

An interface shall be provided so that the drive current of each motor can be monitored by the software at some rate, up to 50 Hz.

An interface shall be provided so that the information from each tacho-generator can be monitored by the MCS software at some rate, up to 50 Hz. There is no longer a requirement to monitor at the servo rate since the tacho loops are to be closed in the power amplifiers.

3.1.3. Function

3.1.3.1. Implementation

The hardware chosen to implement the position loop controller is the PMAC VME card provided by Delta Tau. One of these cards will be used for each axis and it will implement the encoder (see Section 3.2) and servo algorithms. The set up of these cards is quite complex and is described in detail in reference [11]. Note that the default setting for the servo rate in PMAC is approximately 2 kHz. Unless a reduction in servo rate is required by another system (e.g. the encoders), and the reduction is proved, by simulation, to have no effect upon servo performance, the sampling frequency will be not be changed from the default value.

3.1.3.2. Servo Specification

The error budget, after tip/tilt correction, agreed with the IGPO is 20 milli-arcseconds RMS, on axis, at zenith. This increases with zenith distance, *Z*, according to the following equation:

$$E = 0.02 \times \sqrt{1 + \sin^2(Z)}$$

Where *E* is the permitted error, on axis, in arcseconds.

The on axis error is calculated from individual axis errors by using the following equation:

$$E_T = \sqrt{E_E{}^2 + [E_A \times \sin(Z)]^2}$$

Where: E_T is total on axis error.
E_A is total azimuth error.
E_E is total elevation error.

Reference [8] uses the non-linear model to estimate the performance of the servos with and without tip/tilt correction. The results are summarised in Table 3.1 and Figure 3.1.

RMS milli-arcseconds	Without Tip/Tilt	With Tip/Tilt
Azimuth	55	23
Elevation	27.7	0.05

Table 3.1 - Summary Of Error Predictions

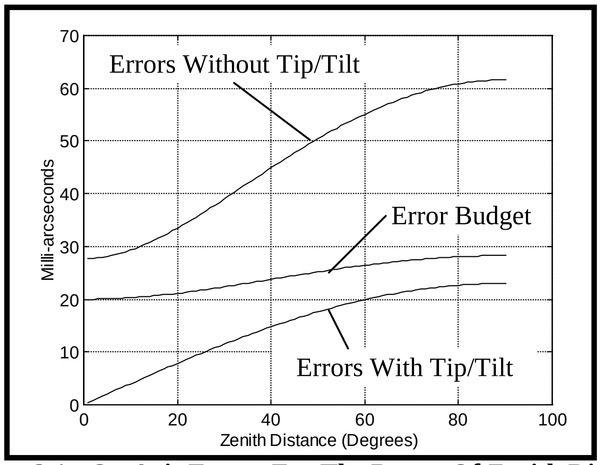


Figure 3.1 - On Axis Errors For The Range Of Zenith Distance

3.1.3.3. Interfacing

3.1.3.3.1. Interface With MCS Software

The MCS shall receive position demands at 20 Hz from the TCS, or a one-off position demand from the TCS or MEC. The MCS software will cause the telescope to track these positions with a combination of “jog” commands and a calculated trajectory based upon interpolation. See reference [2] for more details.

HOW TO PASS THE DEMANDS TO PMAC

Forty locations shall be defined within PMAC's dual-ported RAM, twenty for position demands and twenty for associated velocity demands. This shall be known as the *move argument buffer*. Every 50 ms the EPICS software will receive a demand from the TCS and calculate the next ten position and velocity demands. These are then downloaded to PMAC card using half of the defined block of memory (i.e. 20 locations, ten for positions and ten for velocities). The PMAC card will run a motion program in PVT mode (See reference [4], page 3-125) with the time interval set to 5 ms. This program will consist of an infinite loop of twenty move commands, each one taking its position and velocity arguments from the move argument buffer. The buffer needs to be twice as big as each download (i.e. enough for twenty move commands) so that EPICS can be writing to one half while PMAC is reading from the other. A suitable PMAC motion program is shown in reference [11].

Note that at present, it is not clear whether both position and velocity demands are required, or whether the interpolated rate of 200 Hz is necessary. However, the method proposed provides the capability to update both position and velocity demands at a rate of 200 Hz. Changing the interpolation rate will affect the size of the move argument buffer and the PVT time interval.

HOW TO SYNCHRONISE PMAC TO TAI

The motion program within PMAC can be started from an external hardware signal. The software shall set up the time card so that a suitable signal occurs exactly at the desired start time. Tests have shown that PMAC motion programs can be started in this way with a time error of less than 1 ms.

PMAC will remain in sync by using a reference signal, also from the time card, as a time base (see reference [4], pages 3-42 & 3-171).

HOW TO PROVIDE A DATA LOGGING FUNCTION

To help with problem diagnosis and system analysis it is extremely useful to be able to log values of important parameters (e.g. servo error, raw encoder values, etc.) at a specified frequency for later inspection. It is proposed to make use of PMAC's data gather function and dual-ported RAM in order to provide the data logging function (see reference [4], page 3-220). This is capable of logging up to 24 parameters at any frequency up to the servo rate. However, for real time data logging, the performance will depend upon :

- The amount of DPRAM and processor time taken up by the servo and virtual encoder functions.
- How fast the VME processor can read and store contents of the DPRAM.

In order to make efficient use of memory resources, the parameters to be logged and the logging frequency will be user definable.

3.1.3.3.2. Drive Configuration

Although multiple motors are used to drive the telescope (eight for azimuth and four for elevation), each axis is controlled using a single position loop and single velocity loop. The single position loop is closed on the relevant PMAC card. Each power amplifier closes a velocity loop with the feedback coming from the average of all the tacho signals, as long as all the velocity controllers are the same, this is effectively a single velocity loop. A block diagram of this scheme is shown in Figure 3.2. A reduced version of this scheme (two motors only) has been successfully employed on the friction driven test rig.

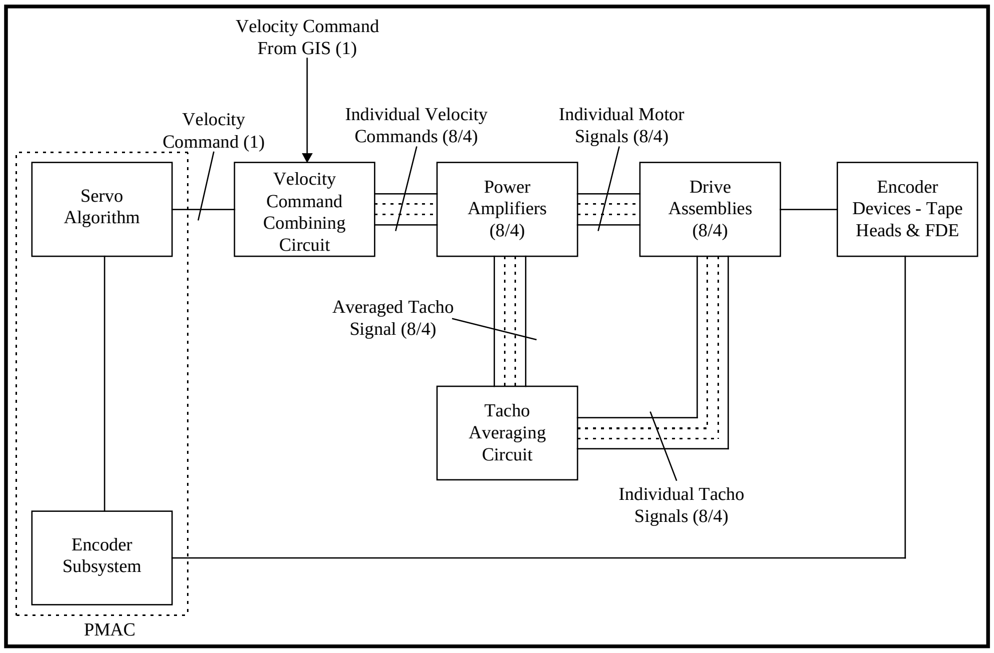

Velocity Command From GIS (1)
Velocity Command (1)
Individual Velocity Commands (8/4)
Individual Motor Signals (8/4)
Servo Algorithm
Velocity Command Combining Circuit
Power Amplifiers (8/4)
Drive Assemblies (8/4)
Encoder Devices - Tape Heads & FDE
Averaged Tacho Signal (8/4)
Tacho Averaging Circuit
Individual Tacho Signals (8/4)
Encoder Subsystem
PMAC

Figure 3.2 - Main Axis Servo Loops

The required set up for the PMAC card is described in reference [11].

The definition of the connections and cable types for the main drive systems (i.e. power amplifiers and motors etc.) is not a formal part of the scope of the MCS work package. However, in the absence of any other formal definition, the MCS/Drive system interface, from the point of view of the MCS, is described below. Note that these settings and connections have been based upon experiments on the friction drive test rig.

HARDWARE CONFIGURATION

Figure 3.3 and Figure 3.4 show the overall wiring scheme for the elevation and azimuth axes respectively. Note that all tacho signals and the tacho average signal are monitored by the MCS - to avoid confusion, this is not shown in the diagrams.

TAV is the Tacho AVeraging circuit. This is an op-amp circuit that takes the individual tacho signals and produces an averaged tacho signal, fanned out eight times (four for elevation).

VCC is the Velocity Command Combining circuit. This is an op-amp circuit sums the MCS velocity demand and the GIS velocity demand and fans the result out eight times (four for elevation). This allows the GIS to move the telescope.

Full circuit diagrams of TAV and VCC are given in reference [20].

DAx is Drive Assembly No. x, including motor, tacho and encoder. ADC is one of the MCS XYCOM 566 VME cards. PSU is a ±15V supply. FST-2 is one of the Power Amplifiers.

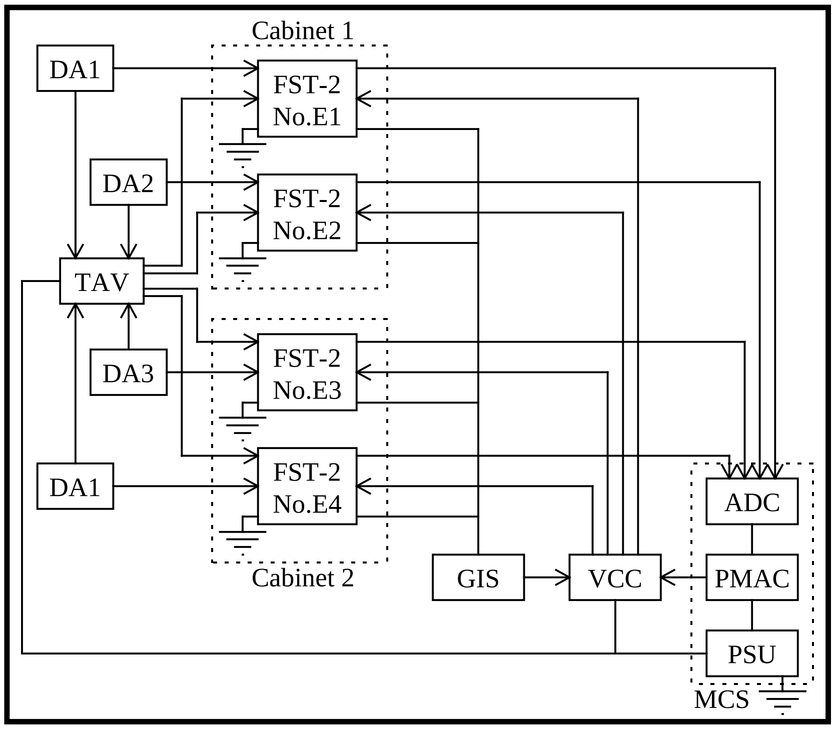


Figure 3.3 - Overall Wiring Scheme For Elevation Axis

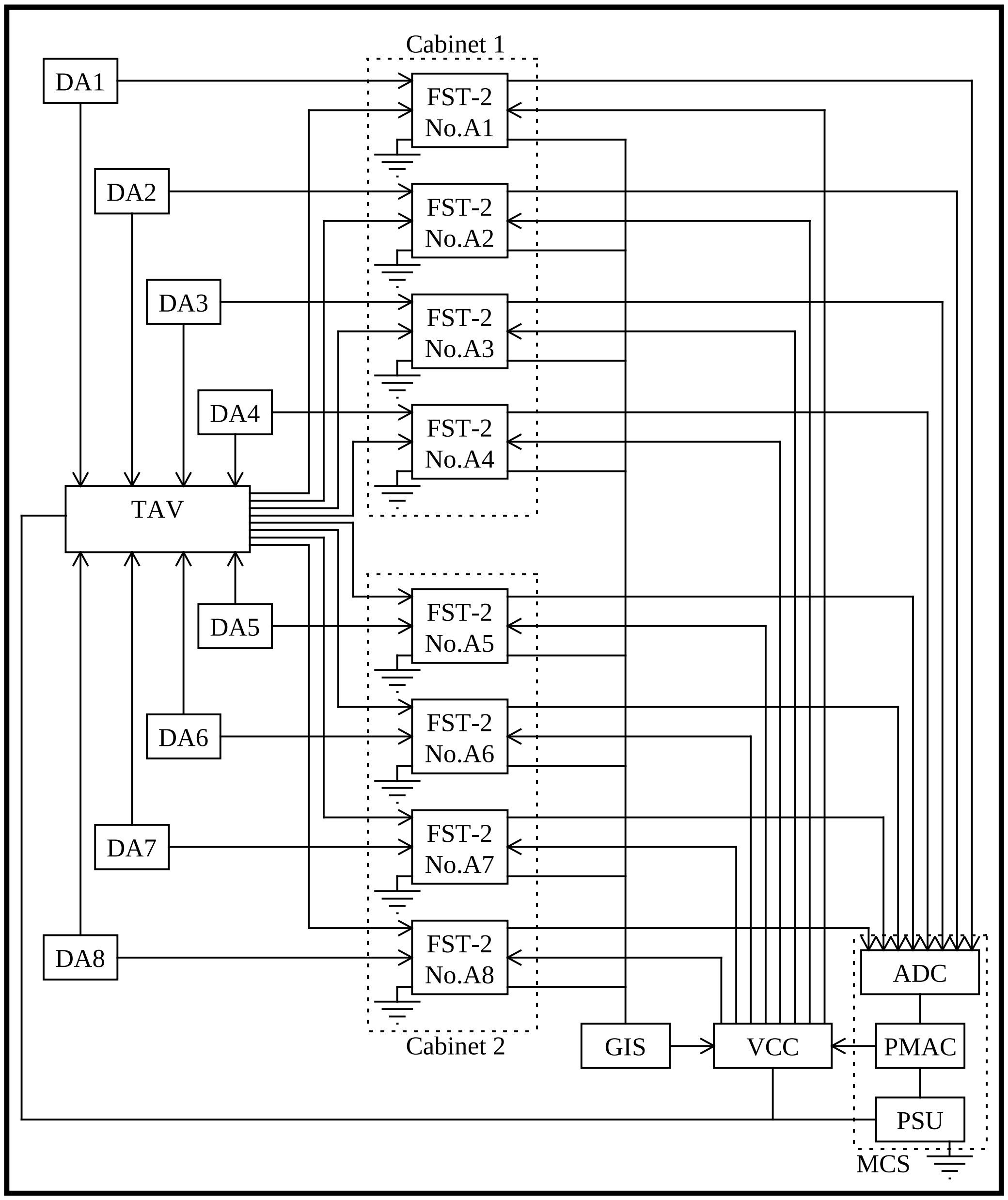


Figure 3.4 - Overall Wiring Scheme For Azimuth Axis

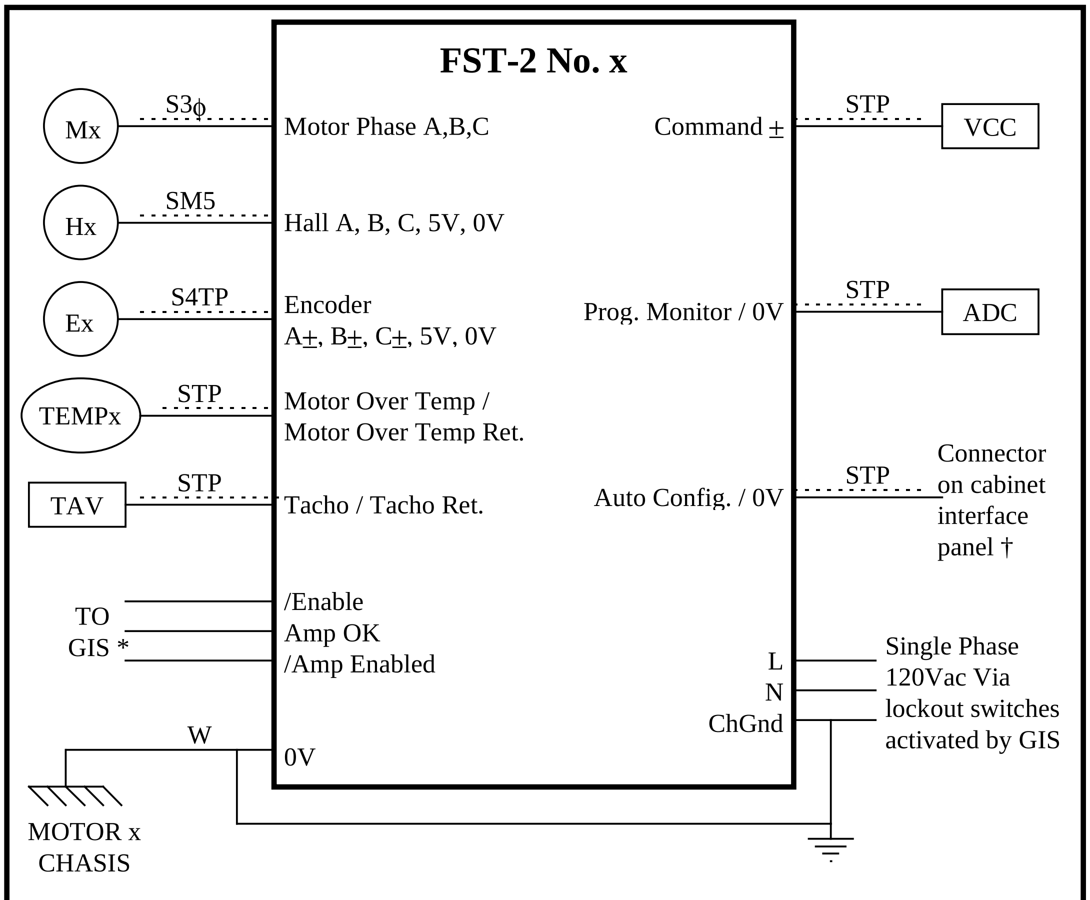
Figure 3.5 shows how each FST-2 amplifier should be connected. The following abbreviations are used in this diagram:

S3ϕ	Screened 3 phase motor current cable.
SMx	Multi-core cable with x conductors and an overall screen.
SxTP	Multi-core cable with x twisted pairs and an overall screen.
STP	Screened twisted pair.
W	Single earth wire.
Mx	Motor associated with drive assembly x.
Hx	Hall sensors associated with the motor of drive assembly x.
Ex	Encoder associated with drive assembly x.
TEMPx	Temperature sensor associated with the motor of drive assembly x.

Note that where a STP is specified, it does not rule out the possibility of combining of many STPs into one SxTP. This decision is ultimately up to the engineer responsible for overall telescope cabling.

Note that in addition the cable types defined in this diagram:

- The cables between tachos and the tacho averaging circuit shall be STPs, shield grounded at the circuit end.
- The cable between PMAC and the velocity command combining circuit shall be a STP, shield grounded at circuit end.



All cable screens should be connected to 0V.
* Details of the interface with the GIS should be defined by the GIS documentation.
† This connector will not normally be mated. It is for engineering purposes only.

Figure 3.5 - Detailed Wiring Of FST-2 Amplifiers

FST-2 PROGRAMMABLE PARAMETERS

Table 3.2 lists the required values of the FST-2 programmable parameters. After these parameters have been programmed (via a RS232 link) a ‘SAV0’ command should be issued to save the parameters to non-volatile memory. For more details on each parameter see reference [22].

Parameter	Value	Purpose.
ANL	0	Analogue input enabled.
CCP	-32422	Current Compensation.
COM	2, 20, 65, 0*	Communications mode.
CPL	6638, -6776	Current loop pole.
CPN	2000†, 32767	Current loop gain.
DGT	0*	Digital command.
ENA	1	Enable H/W enable signal.
ENC	170, -24576‡	Encoder Scaling - 2048 pulses/rev with 4x interpolation.
HLL	0‡	Hall Code Phasing.
ILM	100*	Current Limit.
IND	0*	Index Location - Not used.
LOP	1	Loop rate (1 for tachometer velocity mode).

MOD	3	Loop Mode (Tachometer velocity mode).
MON	0	Monitor select (Current monitor).
MTF	-10000*	Motor temperature fault
PCP	0*	Position loop compensation - Not used.
PDP	0*	Position loop damping gain - Not used.
PHA	0*	Phase advance - Not used.
PHO	0*	Phase offset.
PHS	12, 30	Auto-phasing Parameters - 1.2 A for 7.5 s
PIN	0*	Position integral gain - Not used.
POS	0*	Reset position feedback - Not used.
PPL	0, 0*	Position loop pole - Not used.
PPN	0*	Position loop proportional gain - Not used.
RET	0*	Disable auto retry mode.
RMS	100, -100, 20, 27	RMS current limiting
SCL	6390 / 12780	Absolute current limit - 10.4 A / 20.8 A for az / el.
SIN	0	Commutation mode - Trapezoidal on start up, switching to sinusoidal after first hall code transition.
TOF	0*	Torque offset.
VCP	0†	Velocity loop compensation.
VIN	0†	Velocity loop integral gain.
VOF	0*	Velocity offset.
VPL	-32767, 0†	Velocity loop pole.
VPN	100†	Velocity loop proportional gain.
VSC	±4200†, 16384, -256	Velocity scaling

Table 3.2 - FST-2 Programmable Parameters

NOTES:
* Default values.
† To be tuned. Suggested starting values are given.
‡ Exact values calculated by auto-configure process. Expected values given.

3.1.3.4. Further Servo Development

3.1.3.4.1. Hardware Simulator

The actual telescope will, obviously, not be available during controller development. A hardware in the loop simulation system will be used to imitate the telescope hardware. A HWILS system takes a mathematical model of the system plant and simulates it in real time. This simulation is then interfaced to the controller, enabling development away from the real physical plant.

The hardware for the MCS HWILS system is based upon the products of the German company, dSPACE, and a reduced version of the Gemini non-linear servo model by Mike Burns, see reference [6]. The hardware has been installed and the model has been successfully downloaded.

The system will be used for the following tasks:

- Testing and developing correct operation of controller during start-up, slewing, pointing and tracking modes.
- Optimisation of sampling rates.
- Identification of failure modes and the development of correct recovery behaviour.
- Simulation of encoder non-linearities and effect on tracking performance.
- Development of suitable engineering tests, logging and calibration procedures.
- After integration on site, system identification using part of HWIL hardware and subsequent analysis can adjust and improve model. Controller development can be undertaken without access to telescope.

The hardware simulator consists of the following products:

<u>HARDWARE</u>

1 x DS1003-192	- TMS320C40 Processor Board (dSPACE).
1 x DS2002	- 32 Channel ADC (dSPACE).
1 x DS4001	- Digital-I/O and Timer Board (dSPACE).
1 x P.C.	- DELL GXMT 5120

<u>SOFTWARE</u>
Matlab (MATHWORKS).
Simulink (MATHWORKS).
The Real Time Workshop (MATHWORKS).
Real Time Interface to Simulink C Code Generator (dSPACE).
TRACE - Data Acquisition Software (dSPACE).
COCKPIT - Graphical Instrument Panel (dSPACE).
Texas Instruments C Compiler for C40 (TI).

The hardware simulator models the main axis hardware from power amp input to virtual encoder output. In order to interface the HWILS to PMAC, digital word interfaces to PMAC will be required (PMAC ACC14). Figure 3.6 shows how the hardware simulator connects to the MCS for servo development.

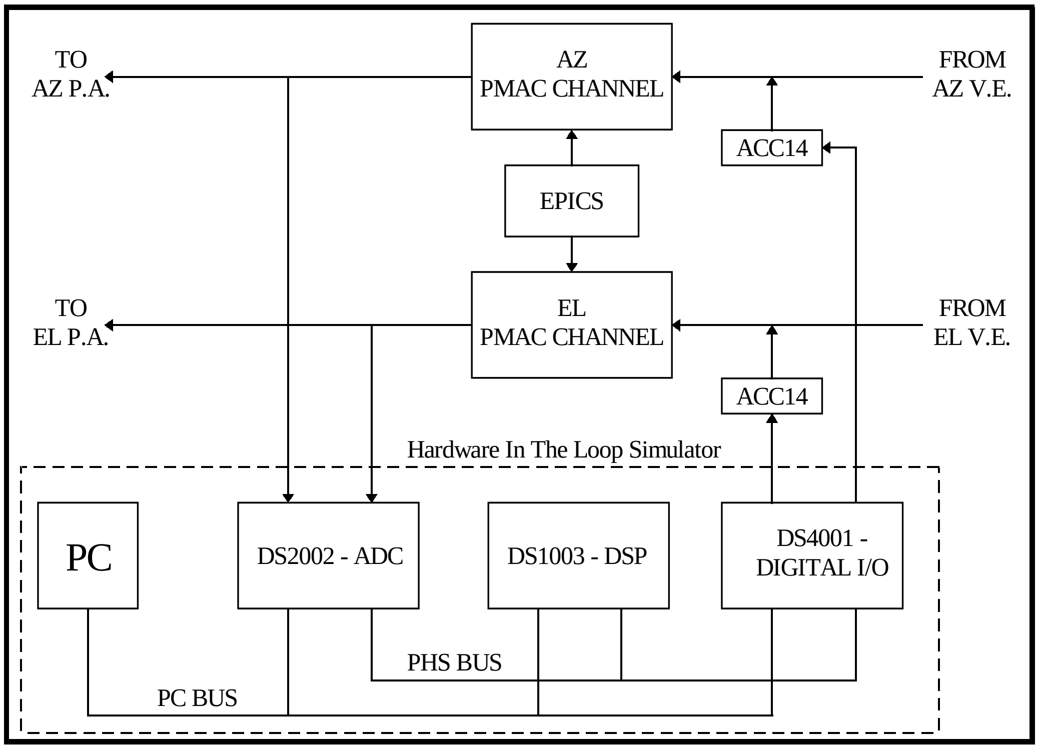


Figure 3.6 - HWILS Configuration

3.1.3.4.2. Test Rig

As part of the MCS design, a friction drive test rig has been assembled. The test rig is an adaptation of the friction-driven encoder test rig built by the IGPO. This consists of a two metre diameter steel wheel mounted vertically in a frame. The rig was modified to include two brushless motors with tachometers and commutation encoders mounted so that they friction drive against the outer diameter of the wheel. Figure 3.7 shows schematically how the test rig drive system is connected. The proposed configuration of the telescope drive components, as detailed in Section 3.1.3.3.2, is based upon this scheme.

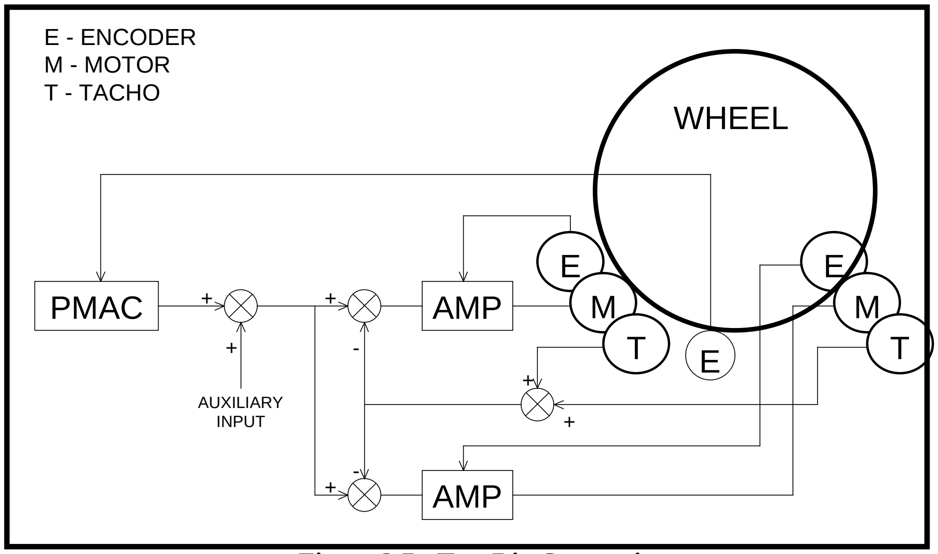


Figure 3.7 - Test Rig Connection

Some measurements of motor torque and drive wheel slippage have been made using the test rig. These tests are in no way comprehensive and are not complete, however, the following observations have been made:

- A mis-match in tacho loop parameters leads to a corresponding imbalance in motor torques. However, the torque sum remains the same indicating that the although one motor may be doing more work, both motors continue to work together rather than against each other.
- The drive wheels do slip even if the theoretical friction torque is NOT exceeded. This slip seems to occur when the wheel changes direction and is more or less repeatable. Note that this low level of slippage does not warrant a removal of torque. The reduction of torque requirement is meant to apply when the drive wheels are slipping uncontrollably (wheel spinning) due to some fault situation and this will be achieved by monitoring tacho and encoder signals in the software.
- Over the test period (≈ 3 months), the drive rollers have worn quite badly. This wear occurred mainly on one side of the roller, see Figure 3.8. It is expected that this is due to an imbalance of the loads on the motor shaft, non-uniformity of the line contact and misalignment of the roller/wheel interface. Also the ratio of hardness between the aluminium roller and the steel wheel is quite large (98 and 150 Brinell respectively). Note that the equivalent hardnesses on the telescope are 42 - 45 and 48.5 - 55.5 Rockwell ‘C’ for roller and track respectively. Despite this, it is still a worry that the roller can get damaged so quickly.

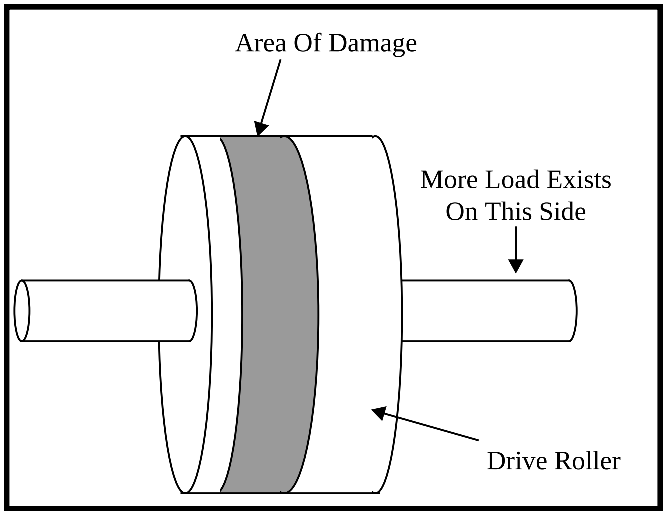


Figure 3.8 - Drive Roller Damage

FURTHER TESTS

Command Passing And Data Logging.

The test rig will play an important rôle in the development of the MCS software interface. The move argument buffer and data logging ideas will be tested out on the test rig drive system.

Comprehensive Slippage Tests

Although extensive slipping (e.g. wheel spinning) is now not expected, it would still be useful to look at the mechanism of wheel slippage to see if it can be detected and even eliminated.

Tests should be conducted with various drive wheel materials, hardnesses and pre-loads. The following variables should be logged:

- Motor currents
- Pre-load Forces
- Tacho signals
- Velocity demand
- Commutation encoders
- Main position encoder

These data can then be analysed to see if slippage of the drive wheels can predicted. The data can also be compared with data from the slippage model to verify and refine this model.

Drive roller wear for different roller materials and pre-loads can be compared.

Effective Gear Ratio Mismatch

The altitude axis of the telescope consists of two drive discs with a drive unit on each disc. Tolerances in manufacture may mean that there could be an effective gear ratio mismatch between the two discs and drive units. This is not thought to be a problem because of the single averaged tacho configuration, and electrical differences in the tacho loops do not cause problems. However, it would be easy to make a couple of drive rollers with different diameters to test this theory.

3.1.3.4.3. Factory tests.

The factory pre-assembly of the first telescope is expected to be complete by mid February 1997. The assembly will remain intact for one month when tests on the elevation drive can be carried out.

DRIVE CABINETS

Before then a prototype of one of the drive cabinets will be built and shipped to France along with as much of the interconnecting cabling that can be realistically made in advance. There are two drive cabinets for each telescope and these house the FST-2 amplifiers and associated hardware. Each cabinet will contain 6 amplifiers, four for azimuth and two for elevation. Since only four amplifiers are needed for the elevation axis, a single prototype cabinet containing four amplifiers is sufficient for the factory tests.

All electrical components other than wires and cables will be procured by the IGPO and delivered to the RGO, where the prototype cabinet will be wired according to a wiring schedule supplied by IGPO. The cabinet shall be delivered to Telas in time for the elevation tests.

After the elevation tests are finished, final designs for the finished telescope drive cabinets will be produced.

OTHER HARDWARE

In addition to the drive cabinet, the following hardware shall be shipped to Telas in time for the elevation tests:

- VME enclosure with at least one PMAC card and relevant accessories.
- Prototypes of the VCC and TAV circuits.
- A P.C. with PMAC Executive software
- Relevant test equipment (e.g. power supplies, frequency response analyser, etc.)
- Miscellaneous tools (e.g. soldering iron, screwdrivers, etc.)
- Miscellaneous electronic supplies (e.g. components, wires, connectors, etc.)

TEST OBJECTIVES

The following objectives have been set for the month of elevation tests:

- Get drive correctly wired up.
- Create a makeshift GIS interface. Unfortunately, the GIS is still in a very early stage of design and will not be available for the factory tests. The GIS functionality will be implemented manually.
- Get velocity loop working - each amplifier has to be set up correctly.
- Get Position loop working.
- Perform frequency response tests. Results can be used to optimise velocity and position loops.
- Perform tracking error tests.

3.2. ENCODER SUBSYSTEM

3.2.1. Introduction

The encoders for the elevation and azimuth axes are a combination of mechanical switches, magnetic position sensors, tape and friction driven incremental encoders. The intent of the encoder subsystem is to hide the details of the physical encoding scheme and provide a device-independent virtual encoder to higher level systems. The MCS will contain two virtual encoders; one each for the azimuth and elevation axes.

It is also a requirement of the MCS work package to procure the tape encoders. Two systems were considered and a decision was made using a series of the tests and a tender exercise. The chosen system is to be supplied by Heidenhain, for more details see references [9] & [10].

3.2.2. Purpose

The encoder subsystem shall receive the following inputs from the mount hardware :

Tape encoder head outputs, four for azimuth and two for elevation - up to 5 kHz pulse stream plus interpolation sinusoids.

Translation sensors for elevation - analogue signals.

One FDE output per axis - up to 2 MHz pulse stream.

Up to 72 azimuth and 22 elevation fiducials - one signal from each fiducial.

An elevation tilt switch. The signal from this will indicate which half of the elevation range the axis is in at MCS IOC power-up.

A signal indicating the position of the elevation limit striker bracket. This bracket can be rotated to provide a different set of lower end, mechanical, elevation limits.

Two azimuth topple brackets. The position of these brackets shall be used to resolve the azimuth position ambiguity at MCS IOC power-up. The valid topple bracket states are summarised in Table 3.3.

AZIMUTH ANGLE	OUTER BRACKET	INNER BRACKET
-270° < AZ < -90°	Not Toppled	Toppled
-90° < AZ < 90°	Not Toppled	Not Toppled
90° < AZ < 270°	Toppled	Not Toppled

Table 3.3 - Azimuth Ranges

There is no longer a requirement for the MCS to read the azimuth absolute encoder. This encoder is intended to back up the topple bracket signals for safety purposes, and as such is read by the GIS only.

It shall be possible to easily extend the number and type of encoders read by the encoder subsystem. Note that the encoder subsystem shall accept input in one of the following forms:

- Incremental Pulses - Quadrature or Up/Down.
- Digital word.
- Analogue voltage ±10 V.

Incremental pulses input are preferred since the other two methods require extra hardware and extra slots in the VME card frame.

There is no longer a requirement to receive an input from the FDE load cell. This is now a function of the Monitoring & Metrology subsystem.

The encoder subsystem shall receive a configuration command from the MCS software.

The encoder subsystem shall implement one of a variety of algorithms in order to produce an output. These algorithms shall include, but not be limited to:

- The average of the outputs of all four azimuth tape encoder heads.
- The average of the outputs of any two azimuth tape encoder heads.
- The average of the outputs of both elevation tape encoder heads with correction made for axis translation.
- the output of any one elevation head with correction made for axis translation.
- The output of any one of the tape encoder heads.
- The output of the FDE.

It shall be possible to easily change and extend the algorithms to include new encoders.

There may also be systematic corrections to apply to certain encoder inputs (probably only the tape encoders, since friction-driven encoder errors will not, in general, be repeatable). The encoder subsystem shall apply these in real time.

The encoder subsystem shall have a calibration mode in which the systematic corrections for each encoder input can be derived.

It shall be possible to up/down load the look-up tables that implement the systematic corrections. These tables shall be in a readable form.

In order to avoid position ambiguity and to allow zero setting to occur at any position, the virtual encoder shall have a resolution of at least 30 bits. A 32 bit resolution is preferred.

NOTE: For the Gemini telescope azimuth axis, the required resolution is 0.005 arcsecs over ±270° range which is to one part in 388,800,000 or 29 bits. To allow an encoder zero set to occur anywhere, the effective range of the encoder becomes ±540°, hence the need for an extra bit.

It shall be possible to directly connect a computer to the encoder subsystem in order to run diagnostics and tests.

The encoder subsystem shall provide a virtual encoder output (≈2 kHz, 32 bit parallel word) to each axis of the MCS servo subsystem.

NOTE: Given a maximum velocity of 2 °/s, an encoder resolution of 0.005 arcsecs and a sampling frequency of 1 kHz, the maximum number of encoder pulses in one sample period will be 720. This can be covered by 10 bits, but to resolve any direction ambiguity an extra bit is added, hence the interface between the VE and the servo system need only be 11 bits wide.

An interface shall be provided so that the following outputs, for each axis, can be monitored by the MCS software at some fixed rate, up to a maximum of 50 Hz:

Slow virtual encoder output - 32 bit parallel word.

4x tape head output - 32 bit parallel words.

FDE output - 32 bit parallel word.

An indication of fiducial position.

Position of the Azimuth Topple brackets.

Elevation displacement transducer readings.

For the requirements of the Heidenhain encoding system itself, see reference [10].

3.2.3. Function

3.2.3.1. Implementation

The computation required by each encoding system (azimuth and elevation) will be implemented in the relevant PMAC card as user written DSP code running in motor #1. For more details on the set-up of these cards see reference [11]. The relevant parameters required by these computational routines will be supplied by the EPICS database, see reference [2].

The interface circuitry required by the different systems will be produced on printed circuit boards and housed in boxes mounted in or around the VME enclosure.

3.2.3.2. *Interfacing*

Precise details of how the encoder hardware interfaces to the PMAC cards is shown in reference [11].

3.2.3.2.1. Fiducial Interface

The encoder subsystem has to receive individual binary signals from a total of 94 separate electromagnetic fiducial switches; 72 on the azimuth axis and 22 on the elevation axis. Each switch provides a 12V output pulse at a maximum activating distance of 8mm. These signals are to be interfaced to the encoder subsystem via the azimuth and elevation PMAC cards.

Due to the large number of fiducials, the switch signals are encoded into a more compact binary form before transmission to the PMAC cards. Because the fiducial encoders are relatively remote from the MCS crate, signal transmission is performed using a differential twisted-pair scheme. An additional circuit at the crate converts these differential signals back into conventional TTL-compatible form. The physical arrangement is shown in Figure 3.9.

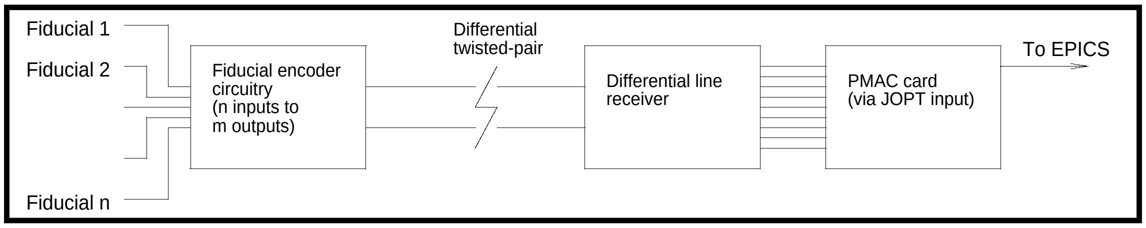

Fiducial 1
Fiducial 2
Fiducial n
Fiducial encoder circuitry (n inputs to m outputs)
Differential twisted-pair
Differential line receiver
PMAC card (via JOPT input)
To EPICS

Figure 3.9 - The fiducial encoding system for one axis

<u>FIDUCIAL ENCODING SYSTEM</u>

The individual fiducial signals are converted into a straight binary code plus even parity. This coding scheme results in an 8-bit azimuth code (7 bits to uniquely identify the fiducial; 1 bit for parity) and a 6-bit elevation code (5 bits to uniquely identify the fiducial; 1 bit for parity).

All incoming fiducial signals are opto-isolated before being passed to the encoding circuit. This is convenient as it handles the conversion of a 12V signal to a logic-level signal, in addition to providing isolation. Further, should the type of fiducial switch be changed in the future, any change in input signal level can be simply accommodated.

The encoding of the fiducial signal is carried out by an EPROM and a diode matrix. Although not the simplest scheme, it is flexible in that changing the fiducial coding simply requires a reprogrammed device. The encoded outputs are then converted into differential signals, before being transmitted over twisted-pair to a receiver circuit at the PMAC card. At the crate end, a differential line receiver converts the signals back to TTL levels before passing them to the PMAC input.

The circuit diagram of the generic fiducial encoder (for both azimuth and elevation axes) can be found in the supplementary documentation.

<u>FIDUCIAL INTERFACE</u>

The fiducial encoding system now no longer requires an Accessory 14 card in order to interface to PMAC; instead, the data is read through the `JOPT` (`J5`) connector on PMAC, once it has been received by the differential line receiver circuit. This receiver circuit also generates a ‘fiducial passed’ signal that is used to trigger PMAC to record the fiducial value. This is derived from a logical OR of the received fiducial data lines, and is sent to the `HMFL1` input on PMAC.

3.2.3.2.2. Encoder Interface

The various devices required to provide the encoding system shall be interfaced to the relevant PMAC card as shown in Figure 3.10 (azimuth) and Figure 3.11 (elevation). Full details of encoder connection and encoder table definition can be found in reference [11].

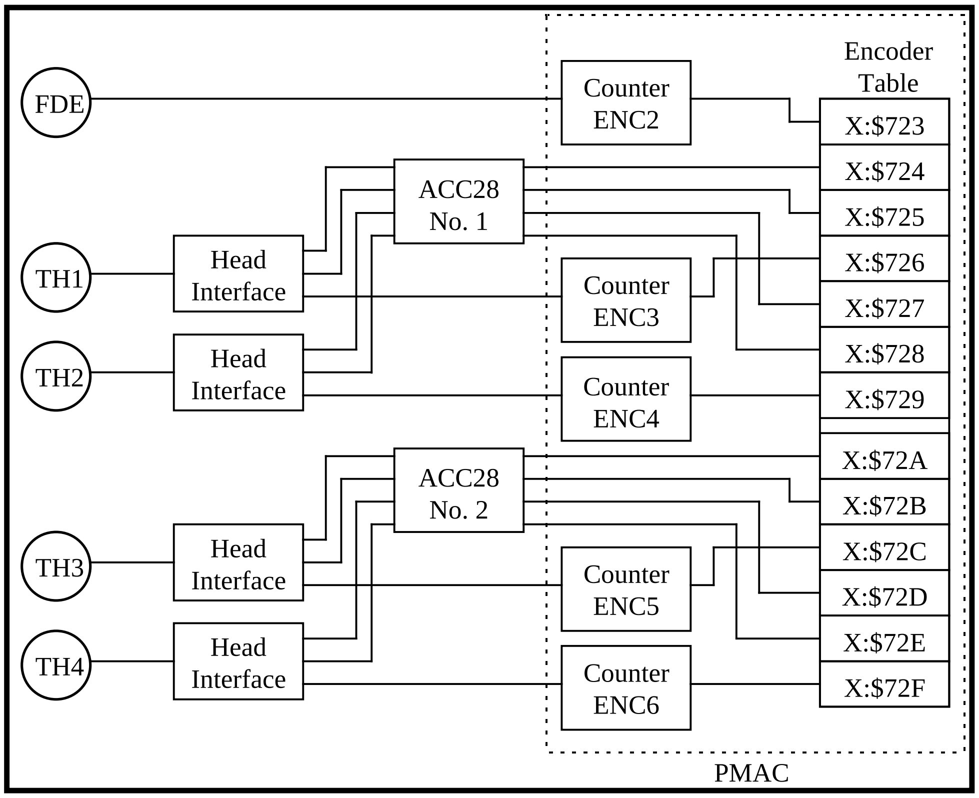


Figure 3.10 - Azimuth Encoder Interface

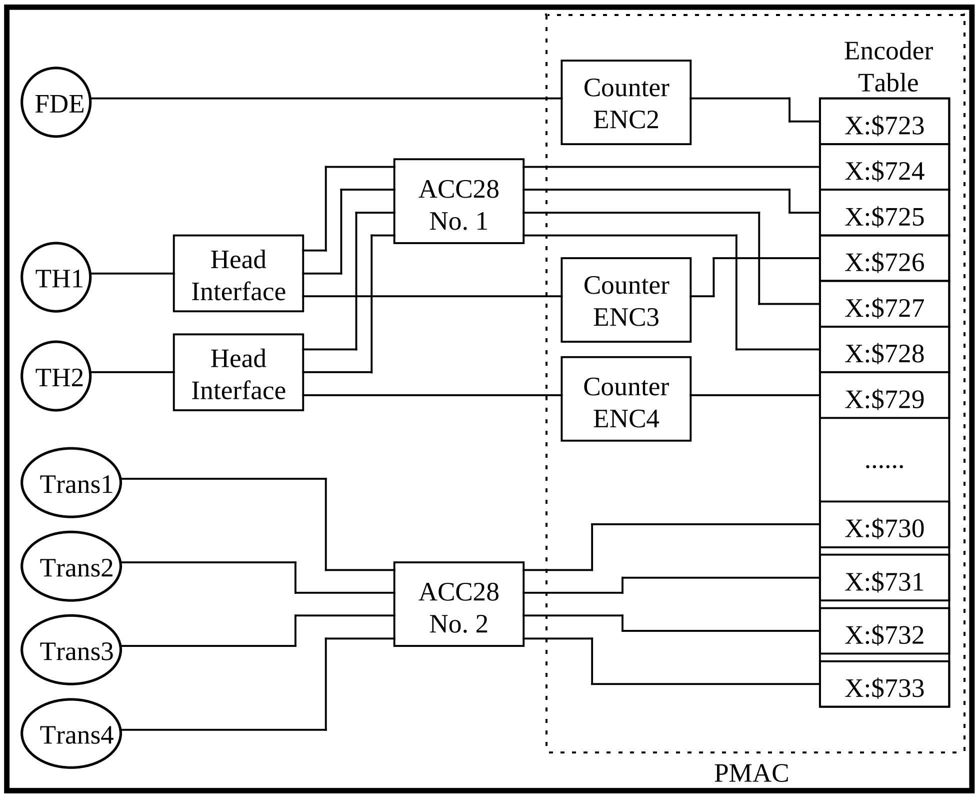


Figure 3.11 - Elevation Encoder Interface

NOTES:

1. Head Interface Block

   In order to derive a position value from each Heidenhain encoder head, we need to count pitches (every 40 µm) and interpolate between these pitches using the two, quadrature, sinusoids. The blocks marked “Head Interface” represent an electronic circuit that will amplify the head signals (sine and cosine) to the correct level for input to the ACC28. The circuit also provides a quadrature square wave output, in phase with the head signals, for input to a PMAC counter. This circuit will be housed in a box along with the ACC28 boards.

2. Encoder Table Units

   - Counter Channels

   Output of each encoder table entry is counts ×32; so each bit represents $^{1}/_{32}$ of a count. For the FDE channel, each bit represents the resolution of the FDE divided by 32. For the tape pitch counting channels, each bit represents 1.25 µm on tape or ≈65 milli-arcseconds on elevation and ≈54 milli-arcseconds on azimuth.

   - ADC Channels

   Output of the encoder table is a signed 21 bit integer (sign extended to 24 bits) representing ± 5V input to the ACC28.

3.2.3.2.3. Software Interface To PMAC

It is necessary for the PMAC card to interface with the EPICS database running on the VME processor card in the MCS crate. PMAC commands and data can be passed using the VME mailbox registers. In essence the operation will be similar to controlling the PMAC card from a PC using the PMAC Executive program. Fast data transfer will be carried out using the dual-ported RAM interface. Data logging will be implemented using PMAC’s data-gather function as described in Section 3.1.3.3.1.

3.2.3.3. Operation

The recent choice of the Heidenhain optical tape as the primary encoding system for the Gemini telescope has clarified the role of the virtual encoder. Each PMAC axis card will now contain two distinct sections of 'user written' code – the first being a tape-head interpolation program, the second the virtual encoder as described in previous documents. When the virtual encoder reads its 'compensated' tape head values, it is doing so from the output of this compensation code.

The two sections of code have been developed independently; the tape interpolation code as a result of encoder evaluation work. Sections 3.2.3.3.2 and 3.2.3.3.3 describe the operation of the tape interpolation and compensation algorithm and the virtual encoder respectively. The actual DSP assembly language is shown in APPENDIX A. The variable space used by these algorithms is part of PMAC's memory normally reserved for P-variables and is defined fully in reference [11].

Figure 3.12 helps to clarify the situation. Figure 3.13 and Figure 3.14 go into more detail for each axis.

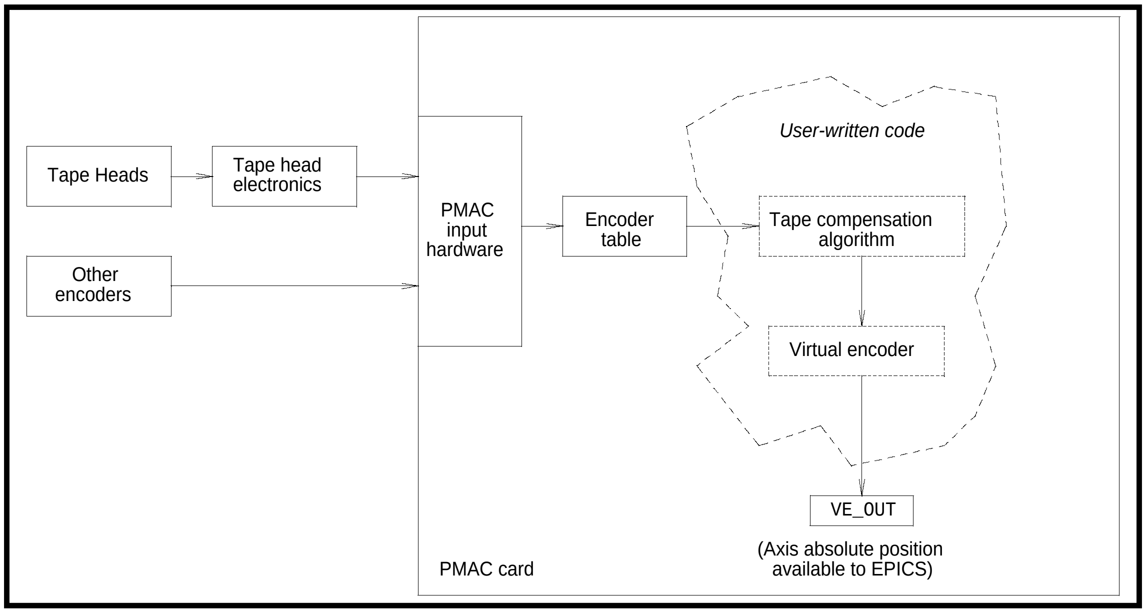


Figure 3.12 - Relative Position Of User Written DSP Code

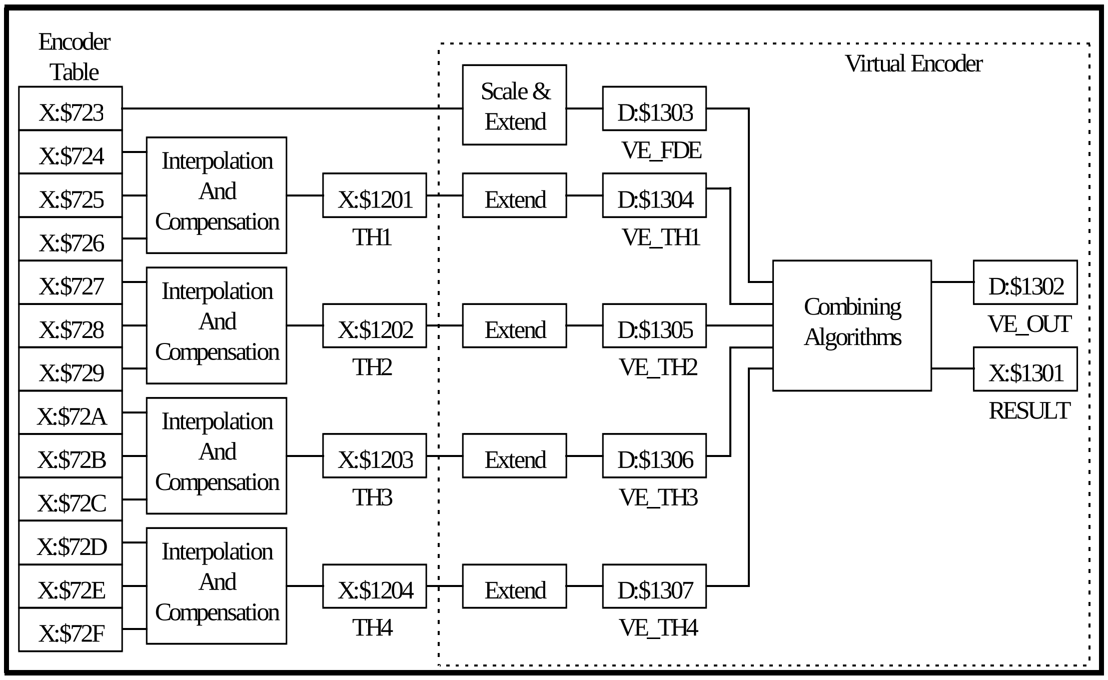


Figure 3.13 - Azimuth User Written DSP Code

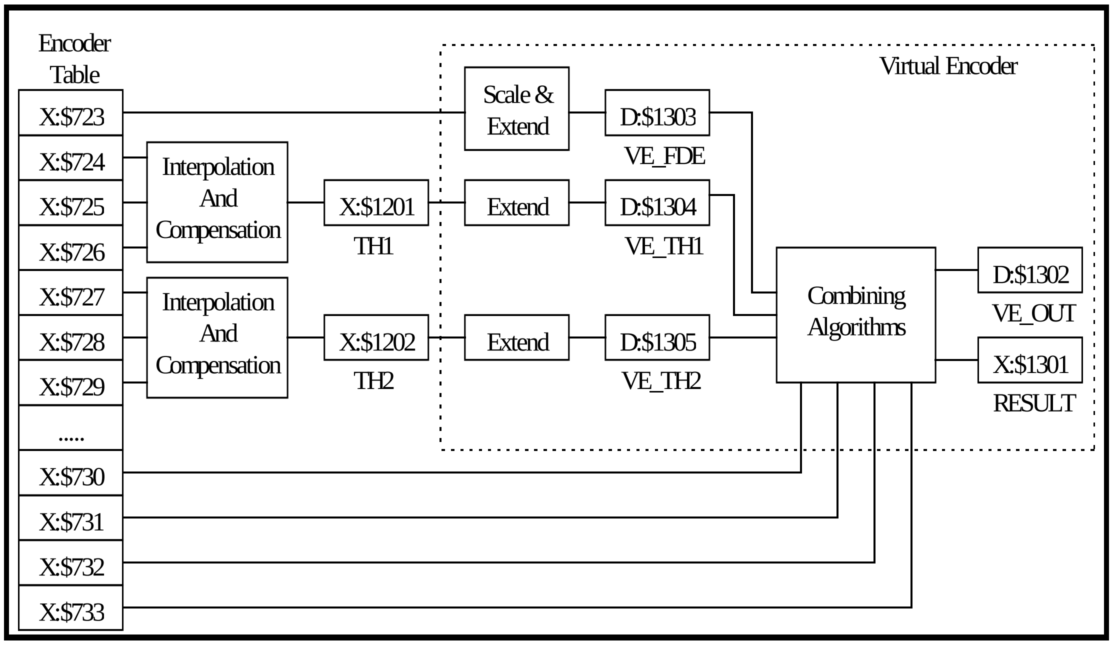


Figure 3.14 - Elevation User Written DSP Code

NOTES:

1. Outputs of Interpolation and Compensation routines (TH1 - TH4) are 24 bit integers with units of one bit equals 5 milli-arcseconds (both axes).
2. Outputs of Virtual Encoder (VE_***) are 48 bit integers with units of one bit equals 5 milli-arcseconds (both axes).
3. RESULT is the lower 24 bits of VE_OUT.

3.2.3.3.1. Initialisation

When the Mount Control System is rebooted, the telescope will have to move to locate a fiducial in order to set the Encoder subsystem to the correct absolute position. Upon reboot, the azimuth axis assumes an absolute position of zero and the elevation axis assumes an absolute position of 45°.

Note that the encoding system does not automatically initialise following a reboot - it will always wait until it receives an initialisation command from the TCS or the MEC. Further, the MCS software will not allow a TCS move command until an initialisation has been performed.

The fiducial signals enter the MCS through the PMAC card as described in Section 3.2.3.2.1. PMAC's standard software homing routine is utilised to reference the encoder system to absolute position. The MCS software (EPICS) would have to implement the following as part of the encoder initialisation procedure:

1. Read range switch (this is the topple brackets / absolute encoder for azimuth and a tilt switch for elevation). The status of the range switch indicates which way the axis must move so as not to hit a limit switch.
2. Set the velocity of the homing move in I-variable I223. This is ±17.2 for azimuth and ±20.8 for elevation. (i.e. a linear velocity of 2 mm/s). The direction is determined by the value read in (1).
3. Enable axis.
4. Issue a HOME command to PMAC.
5. Wait for a "home complete" status bit to become true.
6. Read fiducial code from M19.
7. Disable axis.
8. Add absolute fiducial position to actual motor position register (D:$67).

After initialisation, the fiducial signals shall be continuously monitored by the virtual encoder code. Every time a fiducial mark is passed, the virtual encoder output and the fiducial number will be stored by the virtual encoder and a flag set to inform the MCS software. For more details see section 3.2.3.3.3. and the relevant parts of reference [2].

3.2.3.3.2. Interpolation And Compensation

BASIC ALGORITHM

To provide a position value from each tape encoder head, we need to interpolate between pitches using the analogue sine and cosine signals. The following algorithm will do this:

- Normalise the sinusoidal signals so that they lie between -1 and 1.
- Take the arc-tangent of the sinusoidal amplitude, using both the sine and cosine values to obtain an unambiguous angle, *A*, in the range 0°-360°.
- Calculate *Position* using:

$$Position = \left[PitchCounts + \frac{A}{360} \right] \times 40\mu m$$

PitchCounts is taken from the counter that counts tape pitches.

The flowchart in Figure 3.15 shows how the interpolation and compensation algorithm can be implemented. Because the arctangent function is not available in Motorola 56000 assembly language, it must be implemented as a subroutine. A flowchart for this is shown in Figure 3.16.

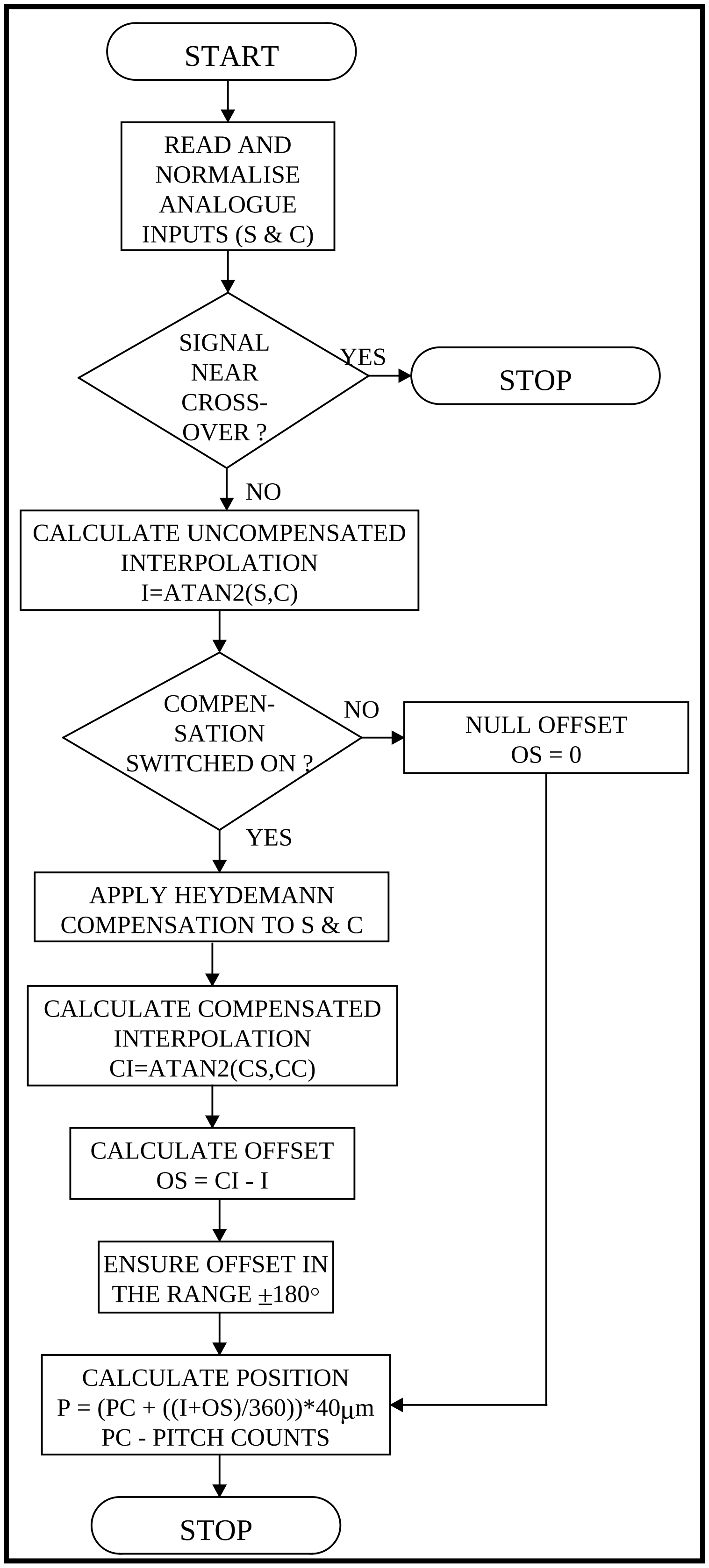

START
READ AND NORMALISE ANALOGUE INPUTS (S & C)
SIGNAL NEAR CROSS-OVER ?
YES
STOP
NO
CALCULATE UNCOMPENSATED INTERPOLATION I=ATAN2(S,C)
COMPEN-SATION SWITCHED ON ?
NO
NULL OFFSET OS = 0
YES
APPLY HEYDEMANN COMPENSATION TO S & C
CALCULATE COMPENSATED INTERPOLATION CI=ATAN2(CS,CC)
CALCULATE OFFSET OS = CI - I
ENSURE OFFSET IN THE RANGE ±180°
CALCULATE POSITION P = (PC + ((I+OS)/360))*40µm PC - PITCH COUNTS
STOP

Figure 3.15 - Flowchart For Interpolation And Compensation Algorithms

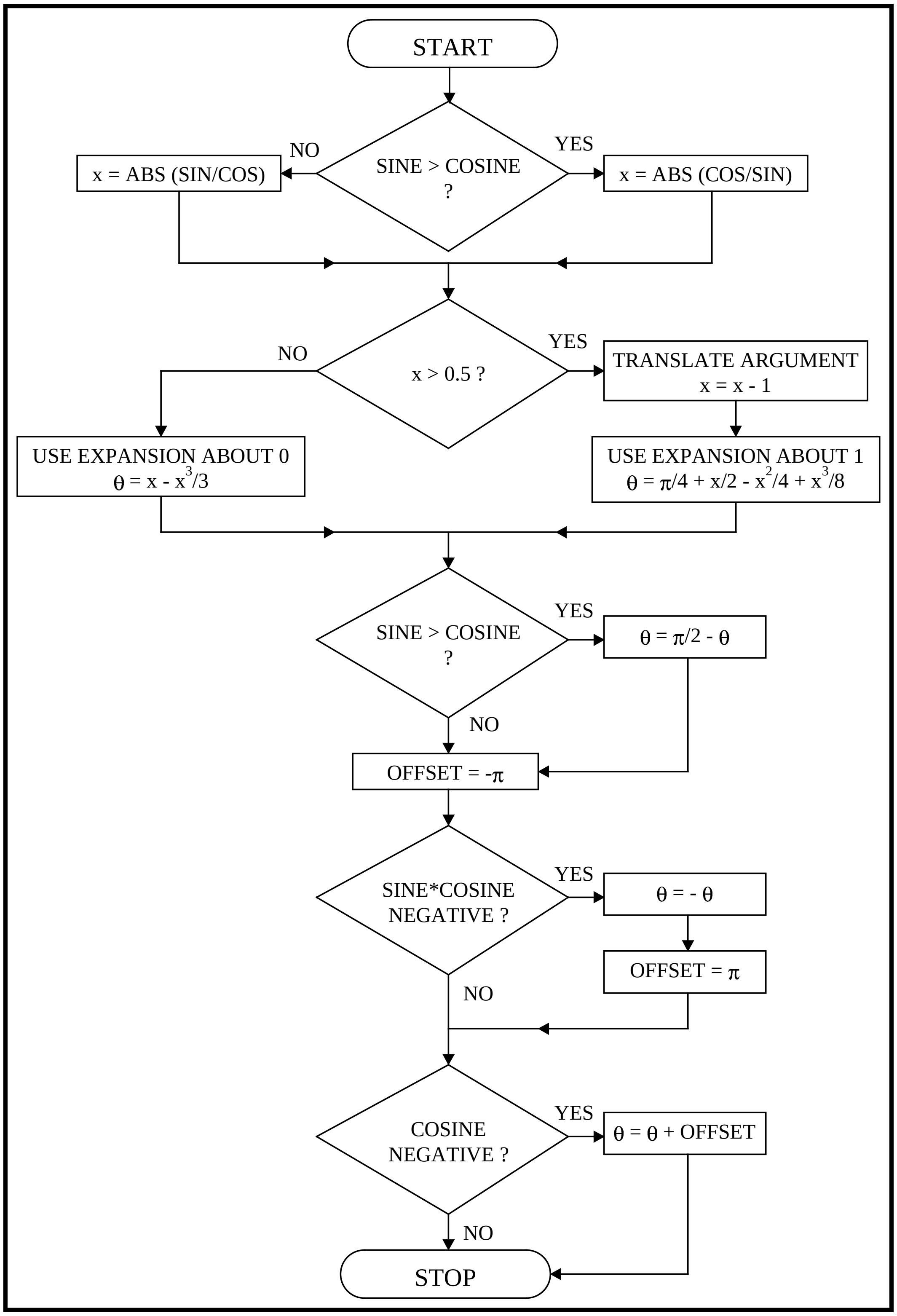


Figure 3.16 - Flowchart For Arctangent Function

OUTPUT SCALING

Calculating position (last block of flowchart in Figure 3.15) must be done carefully so that the result is scaled correctly. The required output scaling is 5 milli-arcseconds per bit, for each axis; this means a linear scaling of:

Azimuth	0.116355283 µm (for 9.6 m diameter)
Elevation	0.096962736 µm (for 8 m diameter)

The pitch count is a 24 bit number, the highest significant 19 bits is the current count of 40 μm pitches. The lower 5 bits are always zero due to the ×32 carried out by the encoder table:

$P_{18}P_{17}P_{16}P_{15}\ P_{14}P_{13}P_{12}P_{11}\ P_{10}P_9P_8P_7\ P_6P_5P_4P_3\ P_2P_1P_0 0\ 0000$

The interpolation and offset values are unsigned 23 bit numbers representing 0-40 μm, i.e. all zeros represents zero μm and all ones represent 40 μm:

$0I_{22}I_{21}I_{20}\ I_{19}I_{18}I_{17}I_{16}\ I_{15}I_{14}I_{13}I_{12}\ I11I_{10}I_9I_8\ I_7I_6I_5I_4\ I_3I_2I_1I_0$

If the pitch count value is shifted left four times and the interpolation value shifted right 14 times, then the two can be added together to leave a 24 bit integer with one bit representing 0.078125 μm (i.e. 40 / $2^{(5+4)}$). Thus the combined result is:

$P_{14}P_{13}P_{12}P_{11}\ P_{10}P_9P_8P_7\ P_6P_5P_4P_3\ P_2P_1P_0I_{22}\ I_{21}I_{20}I_{19}I_{18}\ I_{17}I_{16}I_{15}I_{14}$

All that remains is to multiply by one of the following factors to give the required output scaling:

Azimuth	× 0.078125 / 0.116 = 0.671434916
Elevation	× 0.078125 / 0.097 = 0.805721899

So, to calculate final position:

1. Add interpolation and offset to get a corrected interpolation.
2. Shift corrected interpolation 14 bits right.
3. Shift pitch count 4 bits left.
4. Add shifted pitch count to shifted corrected interpolation.
5. Multiply the result by relevant axis scaling factor.

APPLYING COMPENSATION PARAMETERS

If *ch1* and *ch2* denote the uncompensated quadrature signals, then the compensated signals, *ch1c* and *ch2c*, are given by the following equations (see references [13] & [14]):

$$ch1c = ch1 - p$$
$$ch2c = \frac{ch1c \times Sa + G(ch2 - q)}{Ca}$$

Where *p*, *q*, *a* and *G* are the Heydemann parameters and *Sa* and *Ca* represent sin(*a*) and cos(*a*) respectively.

The *ch1*, *ch2*, *ch1c* and *ch2c* values are scaled as signed 21 bit integer (sign extended to 24 bits) representing ± 5V input to the ACC28. This can also be thought of as 24 bit fraction notation with the binary point after the fourth most significant bit and a range of ± 8. In this case, the ± 5V input range is represented by a number in the range ± 1. If the Heydemann parameters *p*, *q*, *G* and $^1/_{Ca}$ are stored in this same format and *Sa* is stored in standard 56000 fractional notation then the compensation can be applied with the following algorithm:

1. Move *ch1* into accumulator.
2. Subtract *p* from accumulator.
3. Move accumulator into *ch1c*.
4. Move *ch2* into accumulator.
5. Subtract *q* from accumulator.
6. Multiply accumulator by *G*.
7. Shift accumulator left three times.
8. Multiply *Sa* and *ch1c* and add to accumulator.

9. Multiply accumulator by $^{1}/_{Ca}$.
10. Shift accumulator left three times.
11. Move accumulator into *ch2c*.

CALIBRATION MODE

In order to calculate the relevant Heydemann compensation parameters, a set of quadrature, i.e. sine and cosine, values need to be acquired. This is done by moving the axis at a constant velocity and logging the relevant ADC channels at the correct rate. The actual velocity and logging rate will depend upon a number of things:

- The total length of the tape should be divided into a number of equal regions, and each region should use its own local compensation parameters. This allows compensation for local patches of dirt and scratches. For each region, a lower limit on the number of samples per compensation run is arbitrarily set at 20.
- The number of regions depends upon how much data we are prepared to store on the host and the amount of time allowed to do a compensation run. Each region will require the storage of five 24 bit fixed point values. A space of 200 Kbytes will allow about 13650 regions. A reasonable limit for compensation time is one hour per axis.
- The samples must be equally spaced across the 0-360° sub-pitch range.

Calculations and experiments (see reference [9]) indicate that, with these constraints, compensation will be effective.

The Heydemann parameters are calculated off line by solving sets of over defined simultaneous equations using a least squares fitting technique (see references [9], [13] & [14]).

3.2.3.3.3. Virtual Encoder

The virtual encoder for each of the axes is required to generate a discrete value corresponding to the axis absolute position, as summarised in Section 3.2.2. In order to do this the virtual encoder implements a series of algorithms, taking as inputs the encoder counts from axis tape encoders (the primary encoding system), friction driven encoders, and in the case of the elevation axis, linear translation sensors. The tape encoder values are not used directly by the virtual encoder; they have interpolation and compensation applied to them beforehand; see Section 3.2.3.3.2.

In normal operation, the virtual encoder is continuously generating a 48-bit output word corresponding to absolute axis position. The virtual encoder will also have a diagnostics mode, the details of which are to be determined. The most likely scenario would involve EPICS commanding the virtual encoder to write diagnostic data to an array of memory locations in the PMAC address space, which could then be monitored. Data written in this way may include:

- PMAC status bits
- flags used by the virtual encoder or compensation routine
- virtual encoder or compensation routine internal variables

SET-UP WORD

The `SETUP` word is a 24-bit word in PMAC memory space at `X:$1300`. EPICS can cause the virtual encoder to enter certain modes and determine its status by reading and writing to this location.

Echo Mode			Echo Algorithm			Fiducial Passed	Init. Done
Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16

Undefined				n			m
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8

m		Algorithm			Mode		
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

How the virtual encoder interprets the various bit patterns is shown below.

Mode Bits

Bit 2	Bit 1	Bit 0	Virtual Encoder Mode
0	0	0	*Normal* operation
0	0	1	*Diagnostic* mode

Table 3.4 - Virtual Encoder Mode Bits

Algorithm Bits

Bit 5	Bit 4	Bit 3	Virtual Encoder Algorithm
0	0	0	FDE only
0	0	1	Tape N only
0	1	0	Tape M & N only
0	1	1	Azimuth only: Tape heads 1 - 4 (All heads)
0	1	1	Elevation only: Tape heads 1 - 2, plus translation sensors 1 - 4.
1	0	0	Elevation only: Tape head N, plus associated translation sensors (1, 2 or 3, 4).

Table 3.5 - Virtual Encoder Algorithm Bits

Head m, n

These are binary values representing the number of the tape head(s) to be used in the virtual encoder algorithm. For azimuth, m and n may be 1, 2, 3 or 4. For elevation, they may be 1 or 2. Figure 3.17 shows which tape heads correspond to which head numbers for the azimuth axes. In the elevation case, tape head 1 is on the left side of the elevation mounting and tape head 2 is on the right, when viewed from a position facing the access stairway. (It may help to refer to Figure 3.34, on page 60.)

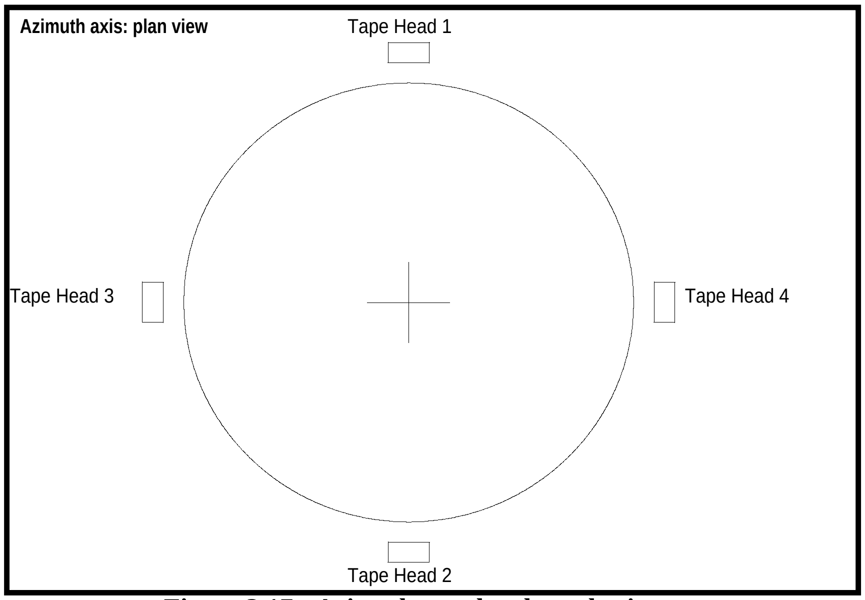


Figure 3.17 - Azimuth tape head numbering

Echo Mode

These bits mirror the virtual encoder mode back to EPICS. This ensures that EPICS receives confirmation of the current mode from the virtual encoder.

Echo Algorithm

These bits mirror the virtual encoder algorithm back to EPICS. This ensures that EPICS receives confirmation of the current algorithm being used to calculate position.

Fiducial Passed

The `Fiducial Passed` bit indicates to EPICS that a fiducial has been passed. It is set by the virtual encoder and can be read and cleared by EPICS.

VE_OUT WORD

This is the 48-bit output of the virtual encoder, and is present in PMAC memory at `D:$1302`. One bit of this word represents 5 milli-arcseconds on axis.

VIRTUAL ENCODER OPERATION FLOWCHART

The operation of the virtual encoder is summarised in the flowchart in Figure 3.18. This code is executed by PMAC exactly once per servo cycle and is not interruptable by other code, PLC programs etc., running on the card.

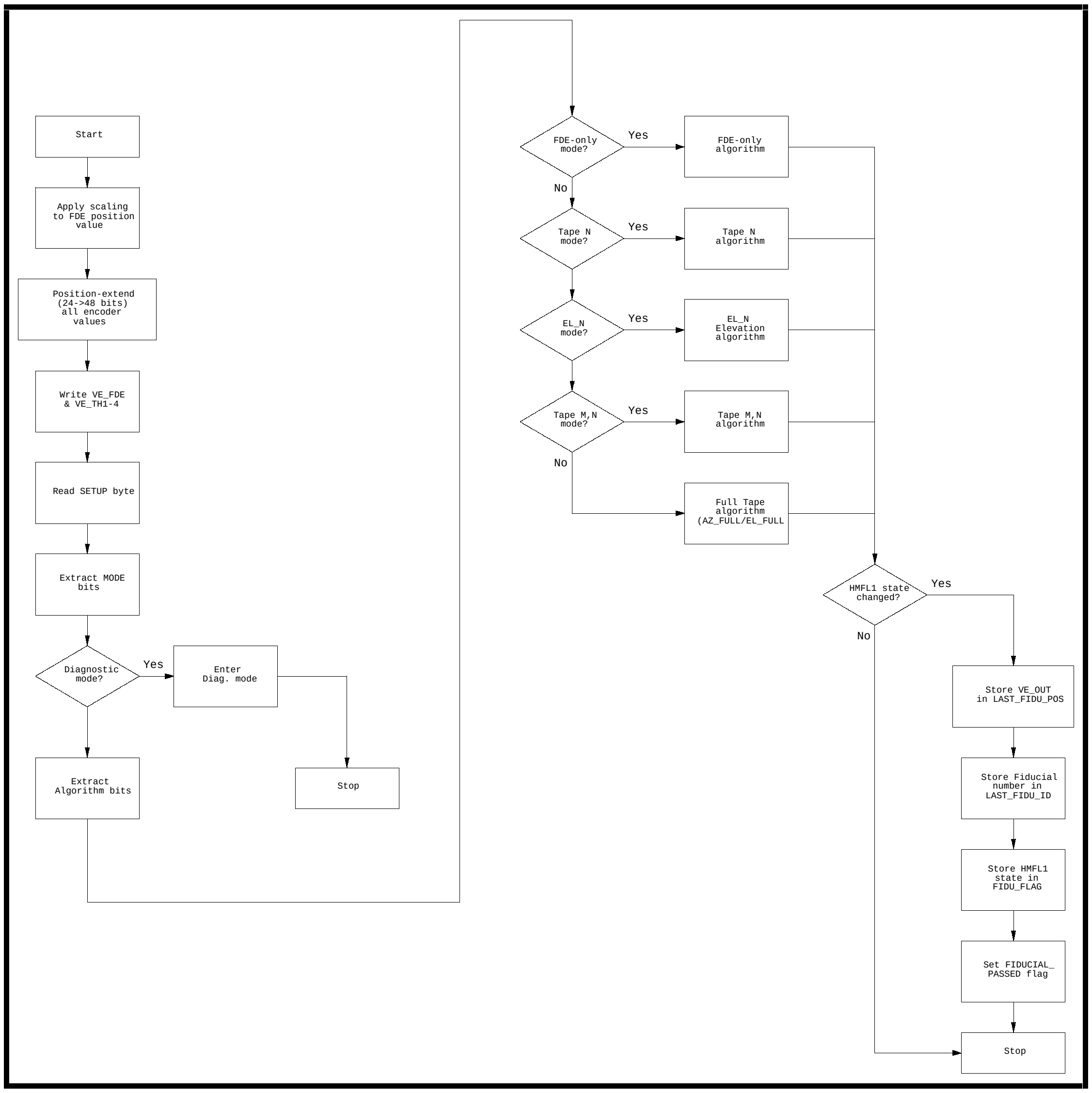


Figure 3.18 - Main virtual encoder flowchart

VIRTUAL ENCODER ALGORITHMS

The flowcharts shown in Figure 3.19, Figure 3.20 and Figure 3.21 are for algorithms that take the various encoder values and calculate the absolute position to be placed in `VE_OUT`. They are selected via the `MODE` bits in `SETUP`. Note that `AZ_FULL` and `EL_FULL` are for the azimuth and elevation axes respectively; also that the `TH` values referred to are the compensated encoder values generated by the interpolation and compensation routine.

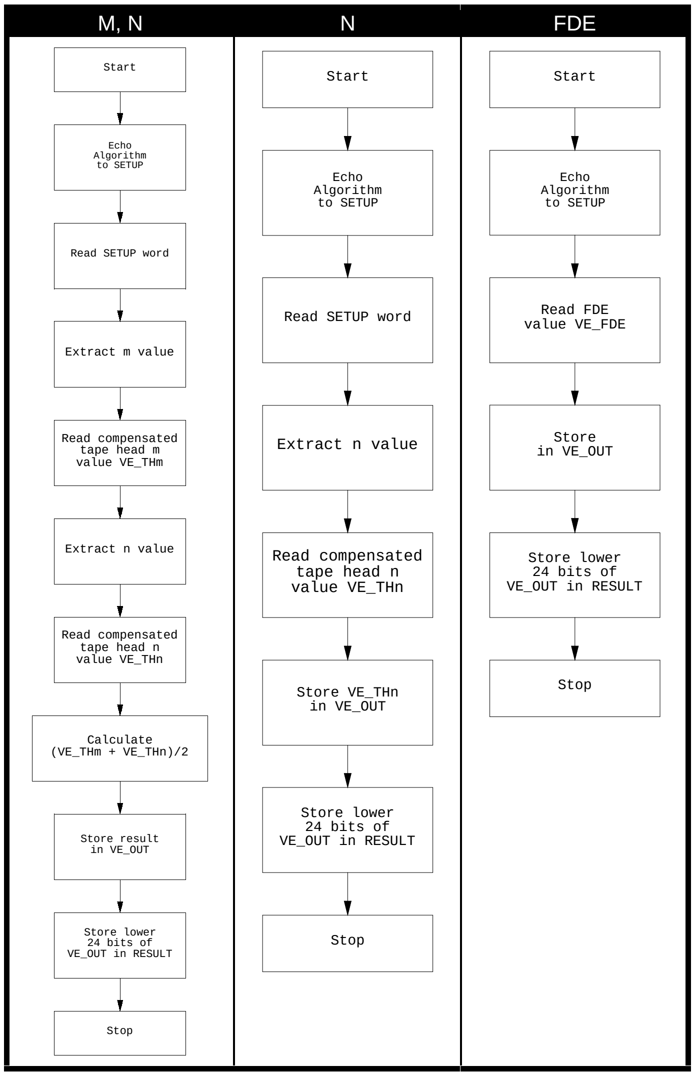

M, N
N
FDE
Start
Echo Algorithm to SETUP
Read SETUP word
Extract m value
Read compensated tape head m value VE_THm
Extract n value
Read compensated tape head n value VE_THn
Calculate (VE_THm + VE_THn)/2
Store result in VE_OUT
Store lower 24 bits of VE_OUT in RESULT
Stop
Start
Echo Algorithm to SETUP
Read SETUP word
Extract n value
Read compensated tape head n value VE_THn
Store VE_THn in VE_OUT
Store lower 24 bits of VE_OUT in RESULT
Stop
Start
Echo Algorithm to SETUP
Read FDE value VE_FDE
Store in VE_OUT
Store lower 24 bits of VE_OUT in RESULT
Stop

Figure 3.19 - Combining Algorithms Common To Both Axes

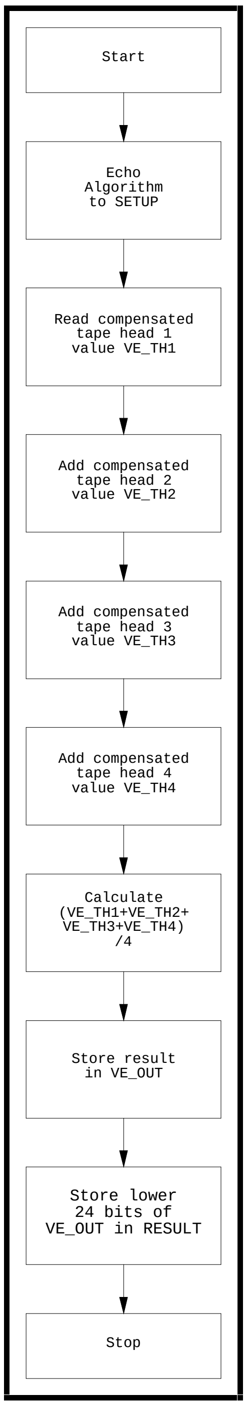

Start
Echo Algorithm to SETUP
Read compensated tape head 1 value VE_TH1
Add compensated tape head 2 value VE_TH2
Add compensated tape head 3 value VE_TH3
Add compensated tape head 4 value VE_TH4
Calculate (VE_TH1+VE_TH2+ VE_TH3+VE_TH4) /4
Store result in VE_OUT
Store lower 24 bits of VE_OUT in RESULT
Stop

Figure 3.20 - AZ_FULL Average Four Heads (Azimuth Axis Only)

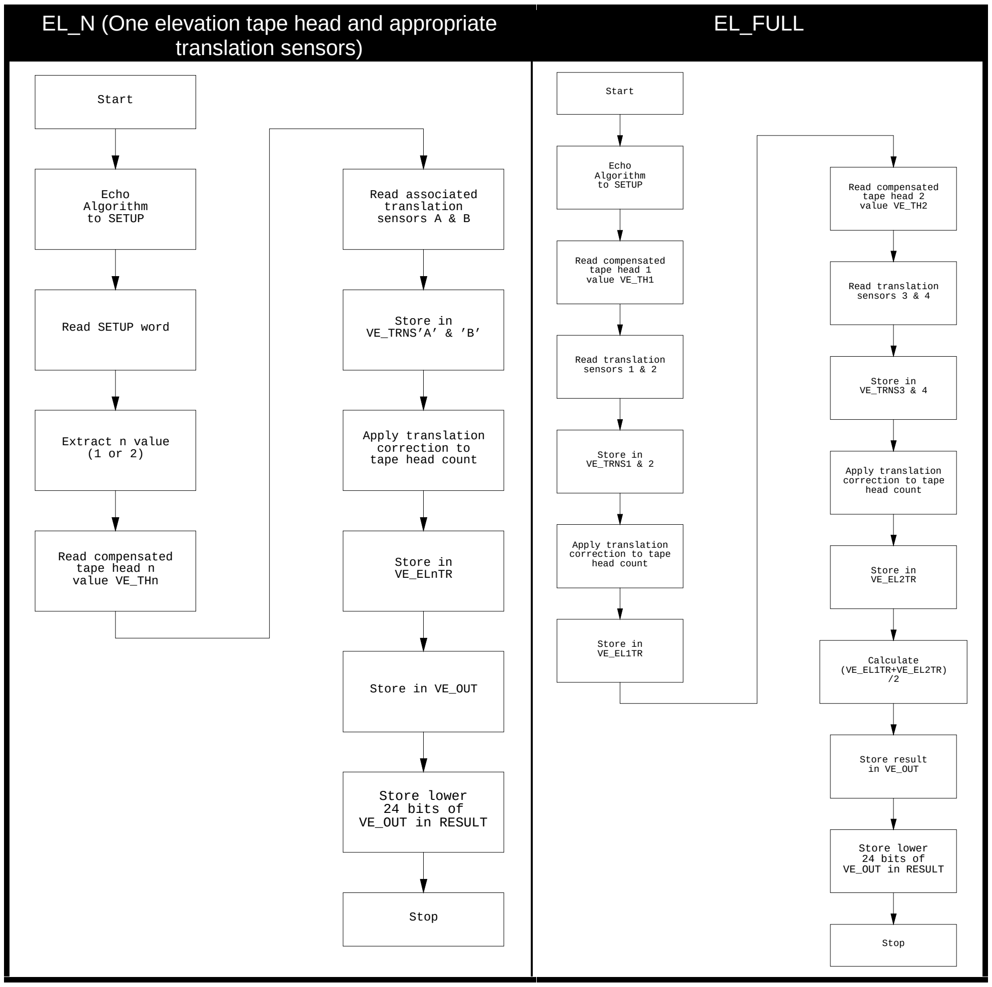

EL_N (One elevation tape head and appropriate translation sensors)
Start
Echo Algorithm to SETUP
Read SETUP word
Extract n value (1 or 2)
Read compensated tape head n value VE_THn
Read associated translation sensors A & B
Store in VE_TRNS'A' & 'B'
Apply translation correction to tape head count
Store in VE_ELnTR
Store in VE_OUT
Store lower 24 bits of VE_OUT in RESULT
Stop
EL_FULL
Start
Echo Algorithm to SETUP
Read compensated tape head 1 value VE_TH1
Read translation sensors 1 & 2
Store in VE_TRNS1 & 2
Apply translation correction to tape head count
Store in VE_EL1TR
Read compensated tape head 2 value VE_TH2
Read translation sensors 3 & 4
Store in VE_TRNS3 & 4
Apply translation correction to tape head count
Store in VE_EL2TR
Calculate (VE_EL1TR+VE_EL2TR) /2
Store result in VE_OUT
Store lower 24 bits of VE_OUT in RESULT
Stop

Figure 3.21 - Elevation Combining Algorithms

3.3. INTERLOCK INTERFACE SUBSYSTEM

3.3.1. Introduction

The Interlock System, or GIS, is part of another work package within the Controls Group. It has ultimate control over the power to the brakes and drive power amplifiers. The MCS must be capable of generating interlock requests in order to instruct the GIS to enable/disable the telescope drives.

3.3.2. Purpose

Each input and output from the Interlock System will be comprised of two TTL signals, one being the complement of the other. These shall be interpreted based on the logic table shown in Table 3.6.

$x \backslash \overline{x}$	0	1
0	Set	Reset
1	Set	Set

Table 3.6 - Interlock System Communication Logic Table

Signals shall be organised so that the Set state is the fail-safe state.

The Interlock Interface subsystem shall receive drive condition signals from the GIS for each of the main drives. The drive condition signal is an Interlock Demand from the GIS.

The Interlock Interface subsystem shall provide drive enable signals to the GIS for the main drives. The drive enable signal is an Interlock Event to the GIS.

There is no longer any requirement to interface drive enable and drive condition signals concerning the service wrap drives and the counterweight drives. These drives are considered non-safety critical and so do not require to be ultimately controlled by the GIS. The limit switches of these drives will be handled by the MCS within the relevant subsystem.

3.3.3. Function

Because the Interlock System and the MCS of the Gemini Telescopes are the subjects of different work packages, the interaction between the two systems must be well defined. This section attempts such a definition and is divided into two parts. The first part deals with the electrical interface between the two systems. The second part describes the desired (by the MCS) action of the Interlock System in the event of limit condition occurring in one of the axes controlled by the MCS.

3.3.3.1. Electrical Interface

Each MCS axis (azimuth and elevation) has an interface to the Interlock System. This consists of two dual TTL signals:

- DRIVE ENABLE (MCS ➔ Interlock System: an Interlock Event)
  The Interlock System treats this as a 'lock out' signal just as it would a signal from a limit switch or locking pin micro-switch. This gives the MCS the capability of controlling the power to the brakes and power amplifiers.
- DRIVE CONDITION (Interlock System ➔ MCS: an Interlock Demand)
  This signal indicates to the MCS whether the associated drive is interlocked out. The MCS will have no knowledge of what is causing the interlock, the reason should be available via the Channel Access interface to the Interlock System.

NOTE: These signals shall interface to the GIS via the standard Gemini TTL I/O board, the XYCOM XVME-240 Digital Input/Output Module.

3.3.3.2. GIS Functionality

It is not the responsibility of the MCS work package to define the operation of the GIS. However, in the absence of any other formal definition, the operation of the GIS, from the point of view of the MCS, is described in this section. Figure 3.22 shows the inputs and outputs of to and from the GIS. For each axis, the GIS implements a logical AND function. When all the inputs are in the safe condition then the axis drives are enabled by the following sequence:

- Power to FST-2 amplifiers is switched on.
- FST-2 '/Enable' lines are pulled low.
- Power to brakes is switched on (This removes the brakes).
- Assert the drive condition signal.

If one or more of the inputs change to the unsafe or interlocked condition, then the axis drives are disabled by the following sequence.

- Dis-assert drive condition signal.
- Power to brakes is switched off (This applies the brakes).
- FST-2 '/Enable' lines are pulled high.
- Power to FST-2 amplifiers is switched off.

Note that it is not absolutely necessary to switch the power to the FST-2 amplifiers on and off since the power stages are adequately isolated by pulling the '/Enable' lines high.

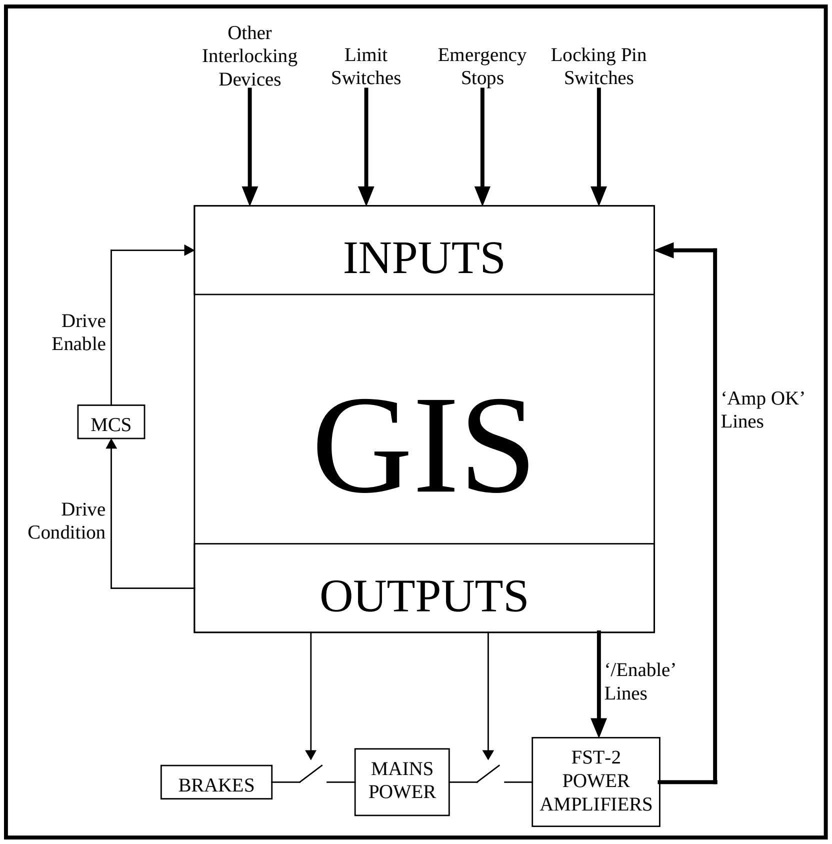


Figure 3.22 - GIS Interconnection of a Single Axis

<u>ENABLING AN AXIS FROM THE MCS</u>

While an axis is STOPped the velocity demand from the MCS is zero and its drive enable signal is not asserted. This means that the axis drive is disabled as described above. This in turn means that the associated drive condition signal is also not asserted. Imagine now that the MCS wants to move the axis, it must:

- Assert the drive enable signal.
- Wait for a pre-determined length of time. This will depend upon how long it takes the GIS to enable the axis drive as described above.
- Check the drive condition signal. If this is now asserted it means that the brakes have been removed and the power amplifiers are enabled, so the required velocity demand can be applied. If the drive condition signal is still not asserted it means that there is something interlocking the axis and so the MCS must report an error.

If the axis is moving when the drive condition signal is dis-asserted then it means that the brakes have been applied and power has been removed from the power amplifiers. The MCS must immediately reduce the demand signal to the power amplifiers to zero, dis-assert the drive enable signal and report an error.

3.3.3.3. Limit Definitions

What happens at limit positions must be considered carefully in order to avoid the possibility of attaining an irreversible interlock condition. This condition could, for want of a better expression, be called 'Interlock dead lock'.

Special cases exist for the azimuth axis, so we divide the problem into two areas:

1. Functionality Common to both Axes
2. Azimuth Axis Special Functionality.

These are considered in the sections below.

In the following sections, the rôle of the GIS from the point of view of the MCS is defined. It is not the responsibility of the MCS to implement this functionality.

3.3.3.3.1. Main Axis Common Functionality

The two main axes, azimuth and elevation, are very similar in operation but the azimuth axis has a greater functionality. Functions common to both axes are dealt with in this section.

We define three limits, positioned as shown in Figure 3.23.

1. MCS Limit
2. Interlock System Limit
3. Damper or Physical Limit

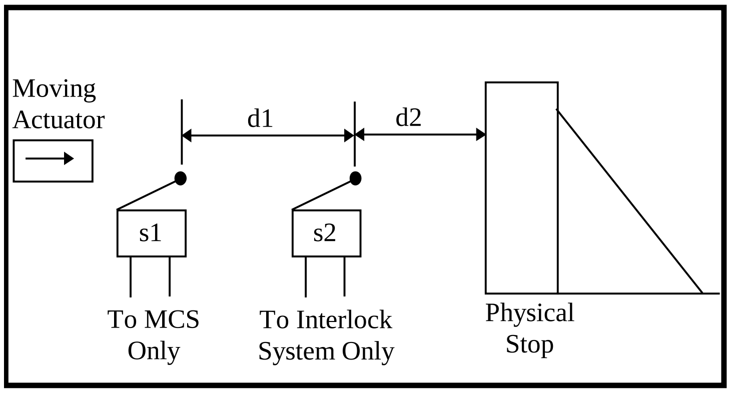


Figure 3.23 - Position Of Limit Switches

<u>MCS LIMIT</u>

The MCS limit is triggered by s1. When this happens the MCS will:

- Ramp down the demand to the drives at maximum deceleration.
- When the axis reaches zero velocity, the drive enable signal is dis-asserted.
- An appropriate error is reported to whichever client issued the original movement command.

While the axis is still in this limit, the MCS will allow movement only in the direction out of the limit. If a movement command that contradicts this rule is received then the axis is not moved and an error is reported.

The desire is to fix distance d1 such that even if the axis is moving with full velocity, the MCS can decelerate to zero velocity before reaching s2. It can be shown (using the data given below) that d1 for azimuth is 20° and d1 for elevation is 5.625°. These values are obviously too large so an area of reduced maximum velocity is required around each limit. This is achieved by adding the MCS pre-limit switch as shown in Figure 3.24.

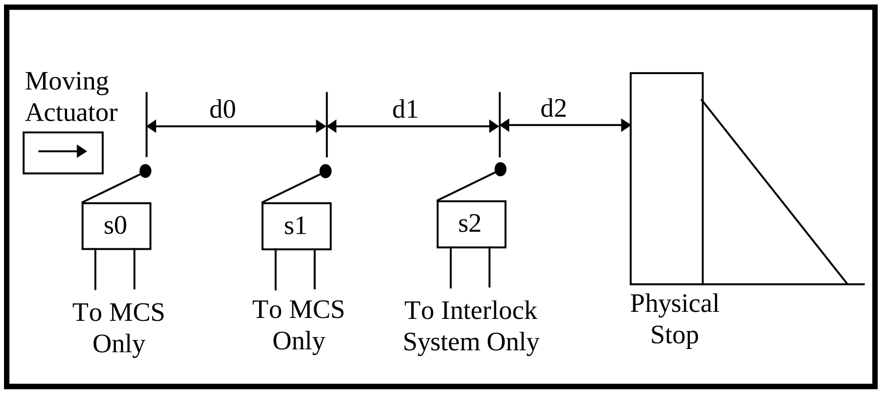


Figure 3.24 - Position Of MCS Pre-Limit Switch s0

The MCS pre-limit is triggered by s0. While in this limit the MCS will continue to operate normally except that the velocity limit is reduced to some value Vsafe. If the axis velocity exceeds Vsafe upon entering the limit then the MCS will ramp down the demand to the drives at maximum deceleration until the velocity has reached Vsafe.

The value of Vsafe and d0 will depend upon the desired value chosen for d1.

An alternative way to reduce the value of d1 is to implement a dynamic velocity limit around the end of the ranges. At any time the velocity limit for an axis is given by:

$$Limit = Vm$$

or

$$Limit = \sqrt{2Am \times S}$$

Whichever is the smallest. Vm is the axis velocity limit, Am is the axis acceleration limit and S is the distance from the nearest limit. This method will effectively reduce the value of d1 to zero since s0 and s1 will no longer exist.

The decision on to which method is to be used will be dependant upon which is the easier to implement in software.

NOTE ON LIMIT SWITCHES
Since s0 and s1 operate soft limits within the MCS they need not be physical micro-switches even though they are depicted as such in the figures above. The limits can be triggered by the MCS encoder value.

INTERLOCK SYSTEM LIMIT

Under normal operation s2 will never be activated. However, in the event of an MCS failure or when the telescope is under the control of the Interlock system, s2 will trigger the Interlock System Limit. When this happens the Interlock system will:

- Disable the amplifiers (using '/Enable' lines and maybe removing power).
- Apply the brakes.

This will, of course, dis-assert the relevant drive condition signal and there is no way that the MCS can drive the axis out of an Interlock System limit. The axis must be retrieved under the control of the Interlock System.

This time the desire is to fix distance d2 such that even if the axis is moving with full velocity, the brakes can decelerate to zero velocity before reaching the physical stop. For the given values of braking deceleration, it can be shown that the d2 distances are quite small, so there is no need to provide an area of reduced maximum velocity around each limit.

DAMPER OR PHYSICAL LIMIT

The telescope cannot be moved without an active Interlock System so the theory is that the Damper Limit should never be reached. However, in the event of Interlock System or brake failure, these dampers have been designed to stop the telescope at maximum velocity.

ACTUAL VALUES

Table 3.7 contains the data required for the calculation of limit positions.

Item	Azimuth	Elevation
Max. Velocity, Vm	2 °/sec	0.75 °/sec
Max. Acceleration, Am	0.1 °/sec^2	0.05 °/sec^2
Min. Braking Deceleration, Ab	2 °/sec^2	0.8 °/sec^2
Damper Positions, DP	275° & -275°	14.5° & 90°
Observing Limits, OL	270° & -270°	15° & 89.5°

Table 3.7 - Main Axis Limits

The distance, d2, is calculated first using the following equation:

$$d2 = \frac{Vm^2}{2 \times Ab}$$

Distance, d1, is then given by:

$$d1 = \left| DP - OL \right| - d2$$

Vsafe can then be calculated from:

$$Vsafe = \sqrt{2 \times Am \times d1}$$

Then d0 is given by:

$$d0 = \frac{2Vm(Vsafe - Vm) + (Vsafe - Vm)^2}{-2\,Am}$$

All this gives rise to Table 3.8.

AXIS	d2	d1	Vsafe	d0
azimuth	1°	4°	0.9°/s	16°
elevation	0.35°	0.15°	0.12°/s	5.5°

Table 3.8 - Limit Position Summary

Figure 3.25 & Figure 3.26 show the various limit positions for the azimuth and elevation axes.

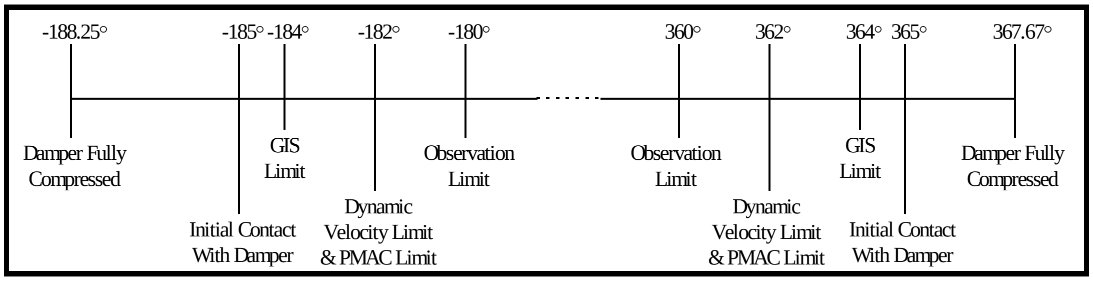


Figure 3.25 - Scale Drawing Showing The Position Of The Various Azimuth Limits

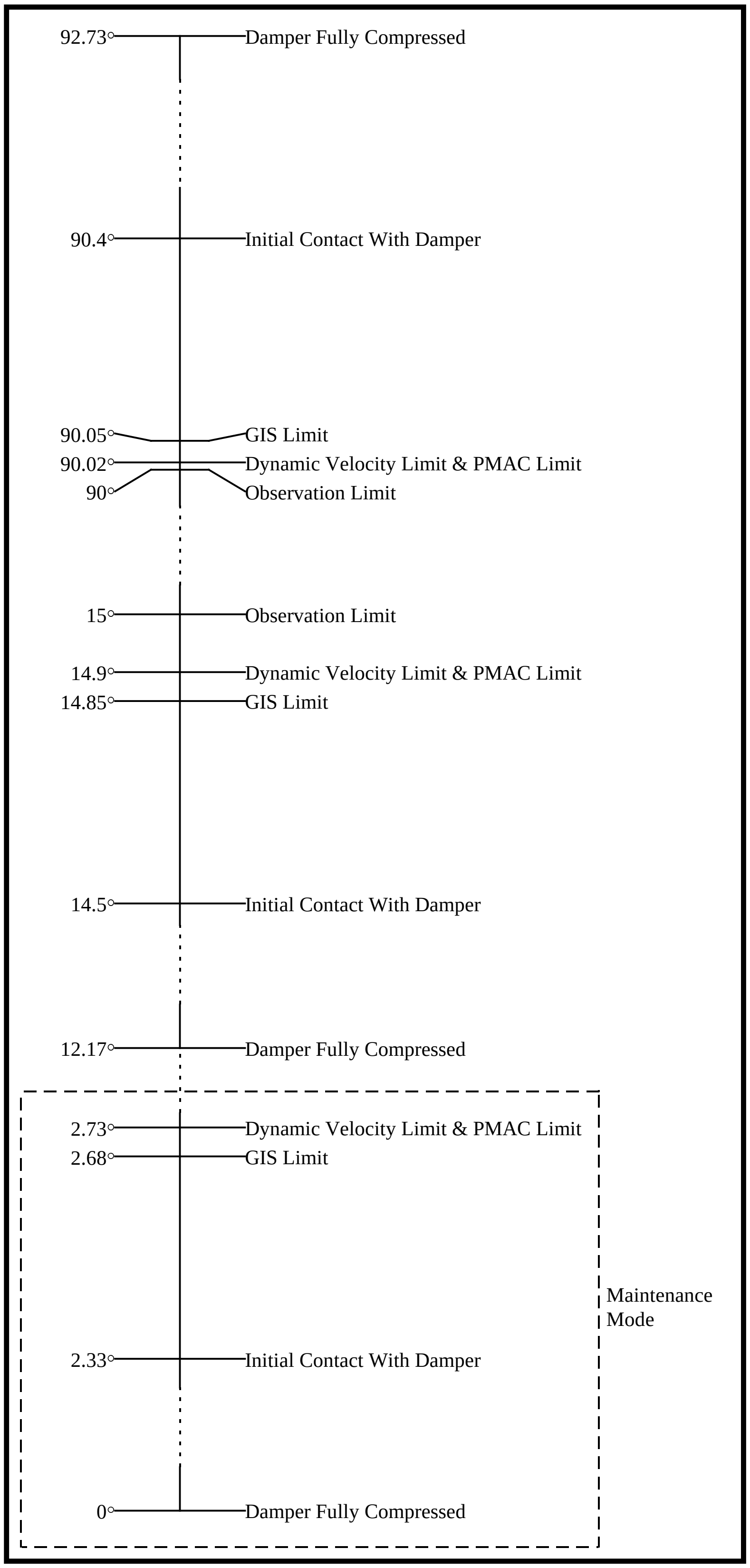


Figure 3.26 - Scale Drawing Showing The Position Of The Various Elevation Limits

3.3.3.3.2. Azimuth Axis Special Functionality

There are two reasons why the interlocking of the azimuth axis may depart from the common functionality as described in Section 3.3.3.3.1:

1. Someone tampers with the Azimuth Topple Brackets.

2. Movement in the 0-15° access position.

TAMPERING WITH THE AZIMUTH TOPPLE BRACKETS

It is the azimuth topple brackets that actuate the Interlock System limit micro-switches *and* the damper limits. If the brackets are illegally moved into the incorrect position then the azimuth axis is left without any hardware protection (electrical or mechanical).

To partially solve this problem, it was originally planned to have additional micro-switches mounted on the inside of the cable wrap so that they are actuated by the wrap bridge rather than the topple brackets. This is, unfortunately, not possible due to the mechanical design of the wrap.

The current solution is to compare the topple bracket position with the coarse absolute encoder that is to be mounted on the azimuth axis, if these two don't agree then an interlocked condition occurs. This is a function of the GIS since the MCS does not read this absolute encoder. This is not entirely desirable since the absolute encoder output cannot be made satisfactorily fail-safe and there would still be no physical stop if the topple brackets were tampered with.

MOVEMENT IN THE 0-15° ACCESS POSITION

For top end ring changes and mirror cleaning it is required to lower the OSS into the 0-15° range (normal elevation limit is 15°). While in this range, azimuth movement must be limited to a small range or damage to the OSS or surrounding structure is possible.

To allow the elevation axis to move below the 15° damper limit, the striker bracket is designed to rotate to provide a new Interlock System limit and damper limit at 0°. There are micro-switches on the rotating bracket that are read by the MCS and the GIS so both can determine its position.

While in the 0-15° mode the MCS will:

- Use modified values for the MCS limit and the MCS pre-limit on the elevation axis in order to allow computer-controlled movement below 15°.
- Use modified values for the MCS limit and the MCS pre-limit on the azimuth axis in order to allow computer-controlled movement only between the safe ranges. Note that this safe range may map to two different ranges within the total 540° of azimuth movement, depending upon its position relative to the cable wrap.
- Use a lower value for maximum velocity for both azimuth and elevation axes.

In order that the Interlock System provides adequate protection while in the 0-15° access position there should be an additional pair of limit switches at a suitable distance outside the azimuth safe range MCS limits. These switches should be activated by fixed striker brackets rather than topple brackets. While in the 0-15° mode the Interlock System will use the output from these micro-switches for a modified Interlock System limit.

The MCS limit, the MCS pre-limit and the Interlock System limit, though triggered at different positions, will have the same effect as described above.

3.4. PROTECTION HARDWARE

3.4.1. Introduction

A number of subsystems require active protection in order to guarantee safety. One of the most important of these is the over-velocity protection system for the telescope.

The system must actively check that the velocities of the different axes are within safe limits and take appropriate action if they are not. The logic of this is to be implemented by the Gemini Interlock System (GIS). It is a requirement of the Mount Control System (MCS) to provide velocity signals to the GIS.

3.4.2. Purpose

The MCS shall provide signals to the interlock system indicating that the velocity of each axis is either above or below some safe limit. These signals shall be derived from good quality tacho-generators.

There is no longer a requirement of the MCS to provide binary signals indicating whether, or not, an axis is near to one of its limits. These requirements are obsolete since there is no longer a need for a zone of reduced maximum velocity around the limits - see Section 3.3.3.3.1.

3.4.3. Function

The transducer used to measure velocity is a tacho-generator. There will be one of these per axis and they will be mounted on one of the encoder brackets. The tachos are supplied by Bowmar Instruments Ltd and are type Servotek PN7453B-1 20.8V. These units output 20.8 V per 1000 RPM or, with a 30 mm diameter wheel mounted on the shaft, about 1V per °/sec of the axis. This means that the output at maximum velocity will be about 2 V.

Located close to the tacho-generators will be a small electronic circuit which will perform a comparison between the voltage from the tachos and a reference voltage that represents the safe velocity limit for each axis. The output shall be in the standard interlock complimentary TTL format. See Section 3.3.2.

Figure 3.27 shows a functional block diagram of the connection between the tacho-generators and the GIS. Vr should be set so that an interlock condition is indicated to the GIS when the axis velocity is just outside the designed maximum. Nominal values for Vr are 3V (3°/s) for azimuth and 1V (1°/s) for elevation. The precise value of Vr will be adjustable, its setting will depend upon the exact diameter of the drive wheel.

A circuit diagram showing the implementation of this interface is show in reference [20]. A ±15V power supply will be required. Ideally, this should be supplied by the GIS, however this may not be possible so the supply for the MCS velocity combining circuit and tacho averaging circuit can be used.

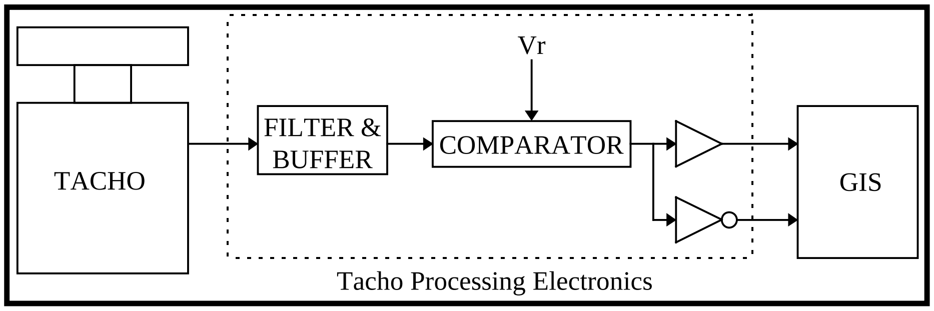


Figure 3.27 - Tacho-Generator to GIS Connection

3.5. COUNTERWEIGHTS SUBSYSTEM

3.5.1. Introduction

Although the telescope is designed to be well balanced in any of the upper-end and instrument cluster configurations it is anticipated that small trim weights will be required to balance the telescope. These weights are remotely controlled and remotely monitored.

The system is comprised of two axial and two cross-axial balance devices mounted to the outside face of the centre section assembly (‘Axial’ is defined here as along the axis of the OSS and ‘cross-axial’ is perpendicular to the OSS axis and to the elevation rotation axis). The counterweight units are located on the same side of the OSS as the elevation disks.

For a fuller description of the OSS counter balance assembly, see reference [3].

3.5.2. Purpose

The counterweights subsystem shall receive a Move command, containing the desired position for each of the balance devices, from the MCS software.

The counterweights subsystem shall receive a position signal at a suitable servo rate, for each of the balance devices, from the counterweight hardware.

The counterweight subsystem shall provide a demand signal, proportional to the required torque or velocity, to each of the drive amplifiers at a suitable servo rate.

Since the failure of a counterweight drive is not a safety issue, there is no requirement to interface with the GIS. Instead, the counterweights subsystem shall read the hardware limit switches and act on them accordingly.

An interface shall be provided so that the following quantities can be monitored by the MCS software at some fixed rate, up to a maximum of 50 Hz.

The demand signal as sent to the power amplifiers.

Actual Position.

3.5.3. Function

The counterweight drives shall be implemented using a PMAC card. One PMAC will cope with both counterweight drives and service-wrap drives. The detailed set-up of this card is described in reference [12].

In order to move a counterweight, the software will have to issue the following commands.

1. Close Loop (This enables the relevant PA).
2. Issue Move Command.
3. Wait For Move Completion
4. Open Loop (This will disable the relevant PA).

While disabled, no power is supplied to the counterweight hardware and the weights will remain in their last commanded position. This is assured by a fail-safe electrically-operated brake (integral to the drive motor).

If a limit switch is triggered while the corresponding counterweight is moving then the counterweights subsystem shall stop the drive and report an error to the EPICS software. While the axis is still in this limit, movement will be allowed only in the direction out of the limit. If a command that contradicts this rule is received then the axis is not moved and an error is reported.

The actual hardware design of the counterweight system has been carried out by the TBEG. Details are included here for reference only. Figure 3.28 shows a block diagram of the proposed system, Table 3.9 details the actual components to be used and Figure 3.29 shows the layout of the hardware.

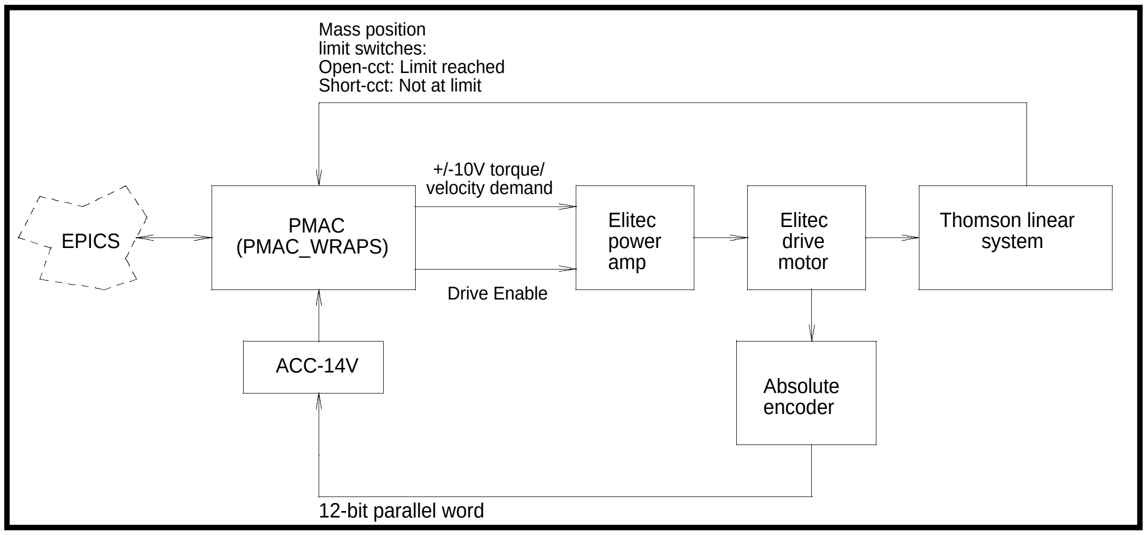


Figure 3.28 - The counterweight drive system

Quantity	Description	Code	Supplier
1	Brushless motor with integral fail-safe brake	MBA105-2005	Elitec Ltd.
1	Absolute encoder	SHM9 series	Elitec Ltd.
1	Linear slide system	2EB-24-FTBJL64	Thomson IBL Ltd.
1	Power amplifier	HPD5	Elitec Ltd.
2	Limit switch	Unspecified	Unspecified

Table 3.9 - Counterweight Hardware Components

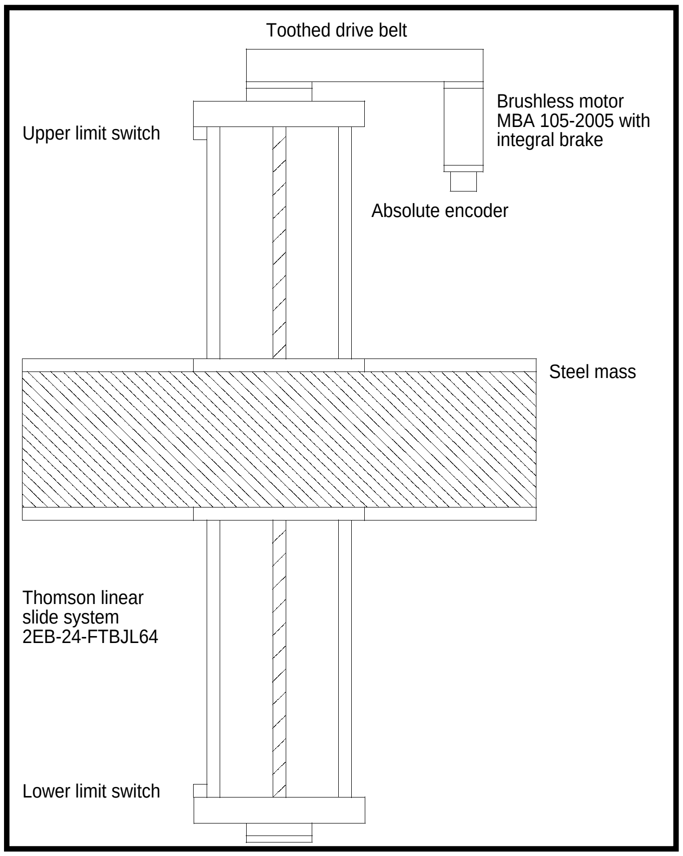


Figure 3.29 - Counterweight Hardware

A dynamic model of a single counterweight has been created using Simulink in Matlab. This can be downloaded to the hardware simulator and run in real time to help develop a suitable controller on the counterweight PMAC card. The Simulink model is shown in Figure 3.30; the input is a voltage corresponding to a torque demand, the output is load mass position.

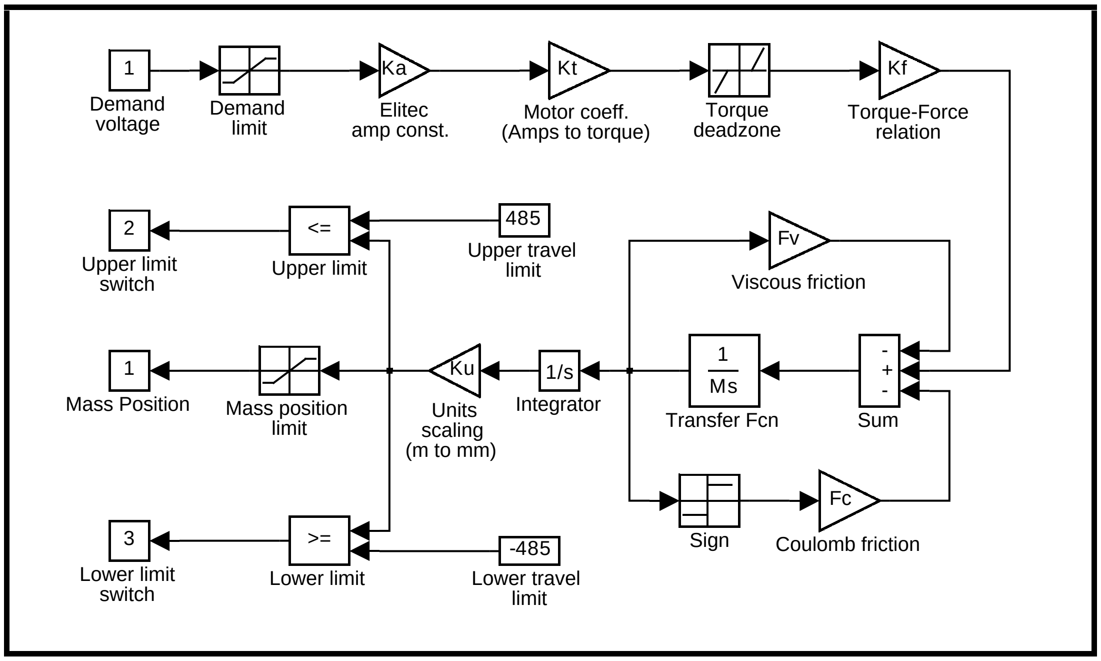


Figure 3.30 - Dynamic Model Of The Counterweight System

Parameter	Value	Units	Description
Ka	0.5	A/V	Amplifier gain constant
Kt	2.42	Nm/A	Motor constant
Tu	4	Nm	Torque deadzone (‡)
Td	-0.5	Nm	Torque deadzone (‡)
Kf	1	N/Nm	Torque-force constant of linear system (‡)
M	3777 (Axial) 1472 (Cross-axial)	N	Load force (3777N – axial, 1472N – cross-axial)
Fv	0.06	Nm/rad/s	Viscous friction of load system (‡)
Fc	1	Nm	Coulomb friction of load system (‡)
Ku	1000	–	Scaling for mass position (m to mm)
(‡) indicates an estimated parameter.			

Table 3.10 - Counterweight Model Parameters

3.6. SERVICE WRAP-UPS SUBSYSTEM

3.6.1. Introduction

In order to provide power, data services, networking, cryogenics, and other services to the Cassegrain focus it is necessary to provide a means of passing the services through the azimuth, elevation, and Cassegrain bearings. The Cassegrain cable wrap is a separate work package and is under the control of the Gemini Instrument Group. In order to minimise the disturbance torque introduced into the motions of the different axes it is advisable to drive the service separately from the telescope axis. It is intended to separately servo the service wrap-ups and to drive them as a follower to the axis they are connected to.

There is one azimuth cable-wrap and two elevation cable-wraps. For a description of the configuration of these, see references [24] & [25].

3.6.2. Purpose

The service wrap-ups subsystem shall receive an input, at a suitable servo rate, from a differential encoder from each drive (azimuth wrap and two elevation wraps), indicating the position relative to the associated main axis.

The service wrap-ups subsystem shall provide a demand signal, at a suitable servo rate, proportional to the required torque or velocity, to each of the drive amplifiers (azimuth wrap and two elevation wraps).

Since the failure of a service wrap is not a safety issue, there is no requirement for the MCS to interface with the GIS. Instead, the service wrap-ups subsystem shall have internal software limits and act on them accordingly. Note that the hardware should contain limit switches which should be read exclusively by the GIS to enable suitable operation during hand-paddle mode.

An interface shall be provided so that the following quantities can be monitored by the software at some fixed rate, up to a maximum of 50 Hz.

The demand signal as sent to the power amplifiers.

The differential encoder read-back.

Motor drive current.

3.6.3. Function

The service wrap drives shall be implemented using a PMAC card. One PMAC will cope with both service-wrap and counterweight drives. The detailed set-up of this card is described in reference [12].

3.6.3.1. Interface To Hardware

Each service-wrap will utilise one PMAC channel which will be permanently enabled and run, as a regulator system, with a constant position demand of zero. Motor feedback will be via an LVDT and PMAC Accessory 28. The motor current shall be monitored via one of the MCS ADC boards. Figure 3.31 shows a schematic of how the MCS fits in with the service wrap hardware.

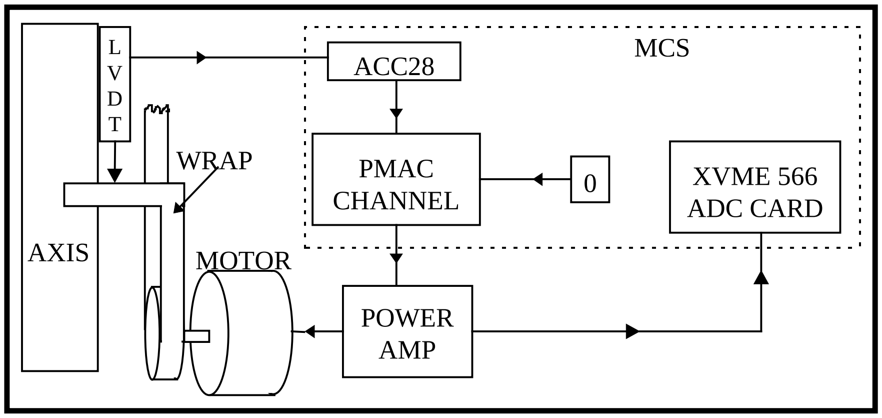


Figure 3.31 - Configuration Of A Service Wrap Drive

Note that the azimuth wrap may consist of two motors in order to provide the correct amount of torque.

The actual hardware required to implement the service-wraps is the responsibility of the TBEG. Table 3.11 details the actual components to be used, this is included for reference only. The following interfaces between this hardware and the MCS are defined:

INPUTS to hardware

- A ±10V signal for each wrap that represents the torque OR velocity required from the motor.

OUTPUTS from hardware

- A ±5V signal from each wrap indicating the absolute relative position between the wrap and the axis that it is tracking.
- A ±10V signal from each wrap indicating the current flowing in the drive motor.
- Two sets signals from each wrap drive indicating end-of-travel position. These shall be read and acted upon by the GIS, so the format of these signals is dependant upon GIS requirements

Description	Code	Supplier
Indurex brushed gear-motor with integral 100:1 gearbox and brake.	KR16M4CH/BRK/KR16-100	Kollmorgen
Servo amplifier 48VDC bus, ±10V control input, 360 W.	KXA-48	Kollmorgen
LVDT +/- 5 V DC output	Unspecified	Unspecified

Table 3.11 - Service Wrap Hardware Components

A dynamic model of the service wrap hardware has been created. This can be downloaded to the hardware simulator and run in real time to help develop a suitable controller on the service wrap PMAC card. The model is shown in Figure 3.32 and the model parameters are defined in Table 3.12. Note that not all the parameters have been defined by the IGPO so suitable estimates have been made.

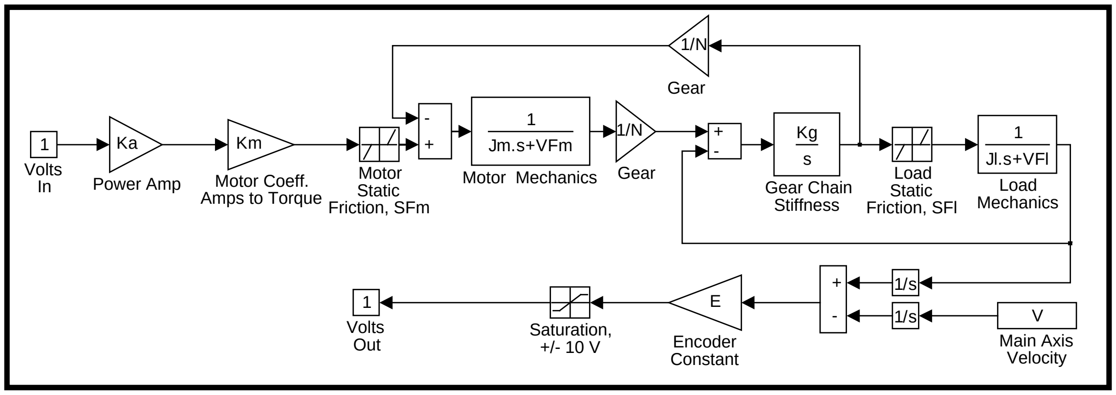


Figure 3.32 - Service Wrap Drive Hardware Simulator Model

Parameter	Value	Units	Description
N	100	-	Gear ratio
Kg	57.3	Nm / rad.	Gearbox stiffness *
Km	0.3728	Nm / A	Motor constant
Jm	6.849e-4	Kg m^2	Motor and gearbox shaft inertia
VFm	6.1e-4	Nm / rad./s	Motor viscous friction
SFm	0.078	Nm	Motor static (Coulomb) friction
SFl	7.8	Nm	Load static (Coulomb) friction *
VFl	6.1e-2	Nm / rad./s	Load viscous friction *
Jl	6.85	Kg m^2	Load inertia *
E	167	V / rad.	Encoder constant *
Ka	1.5	A / V	Amplifier gain
* Estimated Values.			

Table 3.12 - Service Wrap Model Parameters

3.6.3.2. Limits

Figure 3.33 shows, schematically, the interface between the main axis and an associated wrap. Three levels of limit can be identified:

1. MCS Wrap Limit - Triggered from the LVDT reading within the MCS.
2. Interlock System Wrap Limit - Triggered by micro-switches, read by the Interlock System.
3. Drag Plates - Physical plates that provide a mechanical link between the axis and wrap, so that the wrap can be dragged around.

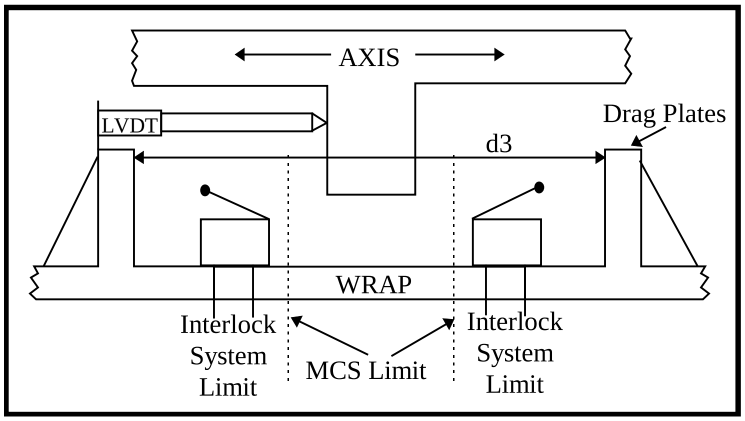


Figure 3.33 - Position Of Cable Wrap Limits

The distance, d3, between the two drag plates is limited by the span of the LVDT and the amount of 'give' in the cables.

The MCS shall disable both the service-wrap drive *and* the associated main drive upon reaching a MCS wrap limit. The effect of hitting an Interlock System limit will be defined by the GIS. Table 3.13 shows the various limit conditions with required actions, suggested recovery methods and possible causes. The safety of the outcome and the effects of the fault persisting are also shown.

Condition	Action	Recover	Possible Cause	Outcome	If Cause Persists
MCS wrap limit reached	MCS will disable wrap drive AND main drive	Successfully re-initialise the wrap servo before main servo is enabled.	Loss of wrap drive or cable jam	Safe	Initialisation fails with Fatal Error limit.
			Wrap drive runaway	Safe	Initialisation fails with Fatal Error limit. Possibility Of hitting GIS limit
GIS wrap limit reached while in MCS mode.	Defined by the GIS. Recommend that main axis is disabled.	Defined by the GIS. Recommend that the GIS wrap limit is cleared before the main axis is allowed to re-enable.	Loss of wrap drive or cable jam and MCS not reading software limit correctly.	Safe	GIS wrap limit will not clear. The fault must be fixed before the main axis can be re-enabled.
			Wrap drive runaway and MCS not reading software limit correctly.	Wrap servo still enabled, motor will stall against drag plate which has become a hard stop since main axis is disabled.	GIS wrap limit will not clear. The fault must be fixed before the main axis can be re-enabled. Damage to motor and amplifiers is possible unless suitable current limits are enforced within the amplifier.
GIS wrap limit reached while not in MCS mode and GIS is not moving the axis.	Defined by the GIS. Recommend nothing since main axis should already be disabled.	Defined by the GIS. Recommend that the GIS wrap limit is cleared before the main axis is allowed to re-enable	Repeated re-initialisation while wrap drive runaway fault exists	Wrap servo still enabled, motor will stall against drag plate which has become a hard stop since main axis is disabled.	GIS wrap limit will not clear. The fault must be fixed before the main axis can be re-enabled. Damage to motor and amplifiers is possible unless suitable current limits are enforced within the amplifier.
GIS wrap limit reached while GIS is moving the axis.	Defined by the GIS. Recommend nothing since dragging of the wrap must be permitted in this mode.	No recovery necessary	MCS Powered down or Wrap servo not initialised.	Wrap is dragged by main axis movement.	Wrap will continue to be dragged by main axis.
			Cable jam	Cables will be stretched by movement of main axis.	Main axis drive will slip or stall, or cables will be ripped out.

Table 3.13 - Service Wraps Interlock Condition Table

3.7. MONITORING SUBSYSTEM

3.7.1. Introduction

It is the responsibility of the MCS to provide a generic means of monitoring the many physical quantities that have been identified. This shall be implemented with a field bus system based upon CANbus as used by the PCS. There will be a number of bus stations at strategic points around the telescope and enclosure. Signals can be brought to these bus stations to be interfaced to the Monitoring system.

The following sections show the sort of input that is to be monitored.

3.7.1.1. Monitoring and Metrology

In addition to the standard encoders used to determine the current position of the telescope, networks of various different types of sensors may be distributed along the mount structure. Examples are:

- Temperature sensors.
- Strain gauges
- Load cells.
- Accelerometers.
- Tilt meters
- Displacement sensors. Note that the translation sensors used to provide corrections to the main elevation encoder are read directly by PMAC, not through the monitoring subsystem.

3.7.1.2. Electrical Systems Interface

In order to meet the 2% down time requirement it is necessary to monitor the status of critical electrical systems. The intent is to prevent problems by a system of periodic monitoring and preventative maintenance. In this system critical electrical systems could be monitored for voltage level, high frequency content, and (for AC systems) conformance with frequency stability specifications. In order to be effective such a system must establish a standard means of interfacing to any electrical system. This standard could be used by fabricators in order to make their systems compatible.

3.7.2. Purpose

The Monitoring subsystem shall be capable of accepting analogue signals in the range ± 10 V.

The Monitoring subsystem shall be capable of accepting binary signals in TTL format.

The Monitoring subsystem shall be capable of accepting digital words of a generic format and handshaking method.

It is the responsibility of the individual sensors to produce outputs in one of the standard formats.

The ADCs contained in the Monitoring subsystem shall have a resolution at least 12 bits.

An interface shall be provided so that EPICS software can monitor all the measured quantities at a maximum rate of 1 Hz.

3.7.3. Function

The Monitoring subsystem is based on a system of node boxes (‘nodes’) that can be placed around the telescope structure, linked to each other and the MCS by a CANbus network. Each node can read sensors and transducers connected to it, convert them into a discrete form and interface them to the MCS crate via CANbus. At the MCS, a VIPC610 CANbus-to-VME interface card enables the received data to be passed to EPICS.

Initially the subsystem will consist of six node boxes per telescope, five of which will be placed on the structure as shown in Figure 3.34. More nodes may be added to the system at a later date, if required.

Note that no nodes have been placed above the level of the mirror to assist with the requirement of keeping heat sources away from the optical path. Any sensors on the top end of the OSS will have to interface to the nearest node sited below mirror level.

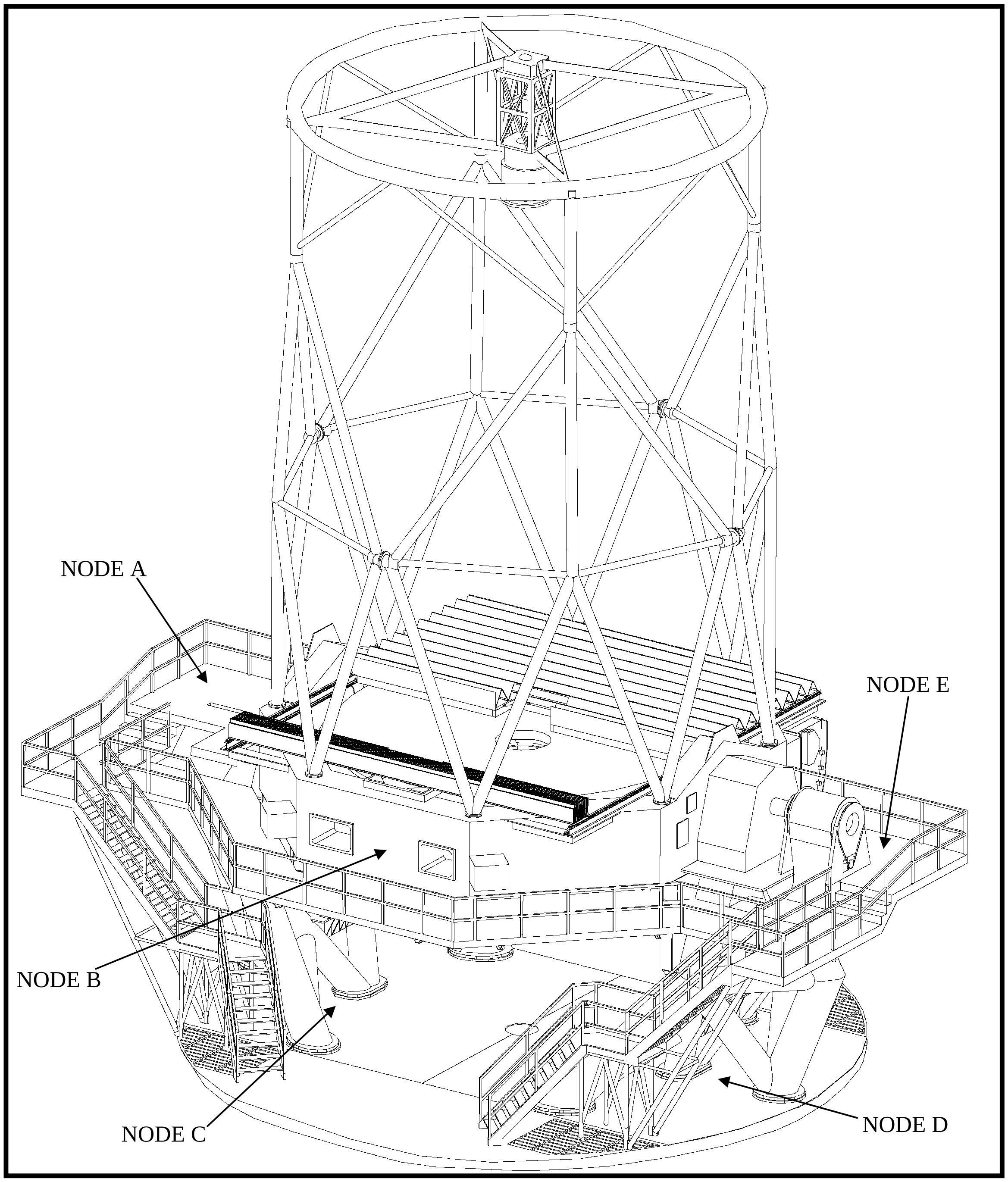


Figure 3.34 - Location Of Monitoring Subsystem Nodes

3.7.3.1. Node Box Specification

Each node box will have the following specification.

3.7.3.1.1. Node Inputs

- Each node shall have 16 multiplexed, differential analogue inputs capable of accepting a voltage signal in the range ±10Vp-p.
- Each node shall have 16 opto-isolated binary inputs.
- Each node shall have two opto-isolated 8-bit binary word inputs.
- The node shall receive and act on commands sent from the MCS via CANbus

Note that it is the responsibility of the sensor to produce an output in a compatible format.

3.7.3.1.2. Node Outputs

The node shall make data available over the CANbus representing the values of the individual inputs, as required by the Monitoring subsystem. The node shall not act as a power supply for external sensors or equipment.

3.7.3.1.3. Node Power Supply

Power for the Monitoring and Metrology node will be derived from an internal linear DC power supply capable of supplying ±12V. It is estimated that each node should consume less than 5W in normal operation; hence a supply rated at least 12.5W should be used to account for the required de-rating. The node connects to an external 120VAC single-phase supply. In addition, an external ±12V DC power supply may be used to supply the node via a 3-pin connector. This is provided against a situation where a node may be required to be mounted where mains power is unavailable. It is not intended that this should be the primary method of powering the nodes.

3.7.3.2. Functional Description

The Monitoring node design is based on that of the PCS Axial Support node; the Monitoring nodes having considerably less functionality. The main differences are in the type of peripheral circuitry – the nodes described here have analogue and binary inputs only, and no outputs. The software required is a subset of that already developed for the Axial nodes.

Full circuit diagrams of the Monitoring and Metrology node are available in reference [21].

A block diagram of the Monitoring node is shown in Figure 3.35.

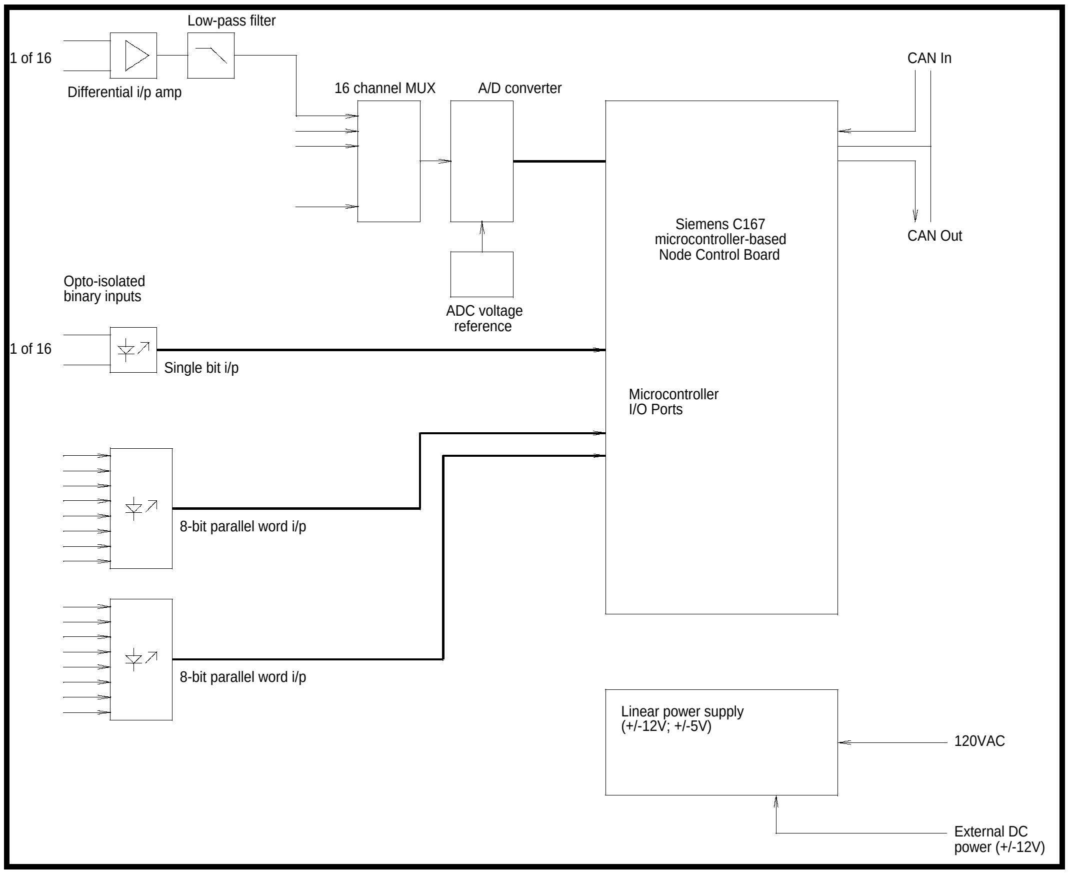


Figure 3.35 - Elements of the monitoring node

The monitoring node box is designed to interface to both analogue and digital sensors. Its core is a *node control board* or NCB, based around the Siemens C167 micro-controller. This unit is the same as used in the PCS system, and handles all the node functionality; reading ports, converting data, CAN interfacing etc. The rest of the node circuitry is devoted to interfacing external signals to this unit, multiplexing data, sampling signals etc.

In normal operation, the NCB will receive commands destined for its particular node box from the MCS over CANbus. It will interpret them, read various combinations of its inputs, process the sampled data, and transmit it back to the MCS.

3.7.3.3. Node Hardware

Nearly all the circuitry in the Monitoring and Metrology node is surface-mount to keep size to a minimum. There are two main PCBs – the NCB, containing the node micro-controller, memory and CAN interface, and the motherboard, which holds the interfacing and ancillary circuitry.

3.7.3.3.1. Node Control Board

The NCB was originally designed for the PCS by the RGO and Coefficient Design Ltd. Only a brief description of directly relevant features is given here – for further details see reference [23].

MICRO-CONTROLLER

The NCB is based around a Siemens SAB-C167CR-LM micro-controller (referred to here are the ‘C167’). Key features include integral serial ports, and extensive port-based I/O and integrated CAN support.

MEMORY

The C167 has 2Kbytes of on-chip RAM, plus 2Kbytes of internal ‘extension’ XRAM. Additionally, the NCB holds 32Kbytes of RAM and 128Kbytes of Flash memory. This will more than meet the requirements of the node software.

I/O PORTS

The node uses the micro-controller I/O ports 3, 5 and 7 to read the 32 binary inputs once they have been opto-isolated.

CAN INTERFACE

The CAN interface is electrically isolated from the rest of the node circuitry.

3.7.3.3.2. Analogue Interface

Each node has 16 analogue input channels, consisting of a differential voltage amplifier followed by a second-order anti-aliasing filter. The circuit diagram of an input channel is given in the supplementary documentation.

THE ANTI-ALIASING FILTER

The anti-aliasing filter is an operational amplifier circuit that implements a second-order Butterworth response. The circuit is shown in Figure 3.36.

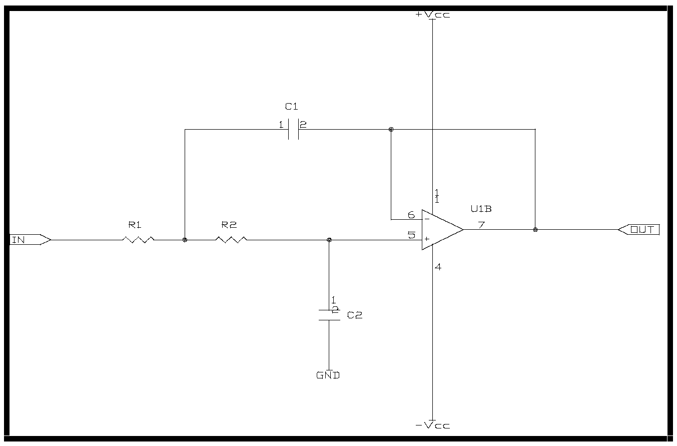


Figure 3.36 - Second-order, low-pass filter

A second-order, low-pass filter with a cut-off frequency F_C of 10Hz can be realised with R1=R2=470KΩ, C1=47nF and C2=22nF (nearest preferred values).

The channel outputs are multiplexed and passed to the 14-bit ADC. A 16-bit word representing the sampled value is then available.

3.7.3.3.3. Binary Interfaces

The node reads binary inputs in two ways; as 8-bit parallel data words and as 16 individual binary bits. A node has two 8-bit parallel inputs and 16 individual bit inputs; all are opto-isolated before being read into the C167 via the I/O ports. The opto-isolators are best driven by a current source capable of supplying at least 5mA.

3.7.3.3.4. Ancillary Circuitry

ADC VOLTAGE REFERENCE

A stable voltage reference for the ADC is provided by a REF-02CP. It will be possible to adjust the reference voltage by suitably choosing two fixed high-stability resistors.

POWER SUPPLY CIRCUITRY

The node is powered from 120VAC single-phase mains by a single linear power supply module that supplies ±12V. It is also possible to power the node externally from a ±12V DC supply via a three pin connector; a diode array ensures the internal PSU is not damaged when this is connected.

Linear regulators on the motherboard supply ±5V where required.

3.7.3.4. Node Software & CANbus Issues

The software for the Monitoring and Metrology nodes is simply a subset of that already developed and reviewed for the PCS Axial Node box. Commands required are essentially those needed to control reading inputs, transferring data and ‘housekeeping’ tasks. For details on the PCS software, see reference [15].

3.7.3.4.1. System Sampling Specification

The following is a workable specification defining how the node is to sample its inputs. The node will, for each analogue input channel:

- band-limit the analogue input signals to 10Hz
- sample the resulting signal at 25Hz
- process the sampled data by decimation/averaging of the sample sequences
- store this data internally
- output this data over CANbus either automatically or when requested by the MCS

Note that the node will sample all the analogue input channels at 25Hz and decimate them all to the same sampling interval. It will not be possible to automatically sample individual channels at different frequencies.

3.7.3.4.2. Analogue Channel Sampling

Each of the sixteen analogue inputs is passed through a second-order anti-aliasing analogue filter that attenuates signal frequencies above $F_S/2$ by 40dB/decade, where F_S is the sampling frequency. The A/D converter then samples the resulting signal at least F_S – in this case, F_S is 25Hz. This process produces a stream of samples at intervals of $1/F_S$ seconds. To allow maximum flexibility, these samples can be *decimated* (i.e. the sample sequence may be filtered and re-sampled at a lower sampling frequency than originally used).

Averaging of the sampled data will also be supported: in this case the node will average a specified number of samples and store the result. This method could be used when the analogue input is known to be either changing extremely slowly or is constant, and is a simple type of sample decimation.

3.7.3.4.3. Sample Decimation

The C167 will be capable of decimating the data that has been sampled at the fixed frequency of 25Hz.

Decimation is a procedure that takes data originally sampled at a (high) frequency and re-samples it at a lower frequency. In essence it involves re-filtering the sampled data and then re-sampling. It is not the same as taking every *n*th sample – that procedure can introduce aliasing errors.

The decimation scheme will be continuous as samples are taken. 25 Hz samples will be passed to through a digital filter (implemented in the C167) and re-sampled. Decimated samples will then be available for transmission over CANbus.

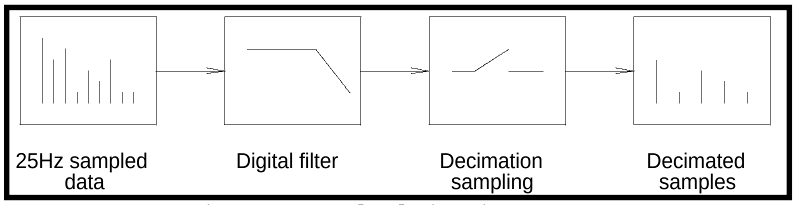


Figure 3.37 - The decimation process

3.7.3.4.4. Binary Input Sampling

The binary input channels are intended for recording the status of switches, parallel digital encoder outputs etc. As such, the main requirement is that they should be able to reject moderate levels of switch noise, contact bounce and so on. This is easily handled in software.

When commanded to read a binary input, the node will repeatedly read the associated port pin(s) a fixed number of times and determine the input state by a majority decision. This is effectively an 'averaging filter' operation.

3.7.3.4.5. Node Commands

Table 3.14 shows a list of the main functions that should be controllable via CANbus messages. Most of these would be minor modifications of existing PCS Axial Node commands.

Function	Description
Read analogue input(s)	Control the reading of analogue input channels. Includes setting sampling frequency, which channels to read, etc.
Read binary inputs(s)	Control the reading of binary input channels. Includes setting the read interval, which inputs to read, etc.
Read binary word input(s)	Control the reading of binary word input channels. Includes setting the read interval.
Reset	Reset the node to a known state.
Engineering functions	Various engineering-level status and test functions.

Table 3.14 - Node Commands

3.7.3.4.6. CANbus Message/Data Format

CANbus communications shall follow the protocols established by the PCS. See reference [16] for more details.

In summary:

- The 11-bit CANbus message identifier will be split into 6 bits to specify the node address and 5 bits to specify individual messages. This means that 63 node addresses are available, and 32 message types per address.
- The 32 messages will be divided into *Operational* and *Engineering* groups, with *Operational* messages having the highest priority. Sampled data will be transferred in the data-bytes associated with these messages.

OPERATIONAL GROUP MESSAGES

Message	Description	Data Byte							
No.		1	2	3	4	5	6	7	8
0	Reserved								
1	Reserved								
2	Node Status	Reset Flag	Checksum Flag	App. Error Flags	App. Error Flags				
3	Bus Status	Tx Error Count	Rx Error Count						

Table 3.15 - Control Messages

Message No.	Description	Data Byte 1	2	3	4	5	6	7	8
4	Binary inputs read	Binary word i/p #1	Binary word i/p #2	Binary inputs #1 to #16					
5	Group 1 analogue i/ps read	Analogue i/p #1: 14-bit value		Analogue i/p #2: 14-bit value		Analogue i/p #3: 14-bit value		Analogue i/p #4: 14-bit value	
6	Group 2 analogue i/ps read	Analogue i/p #5: 14-bit value		Analogue i/p #6: 14-bit value		Analogue i/p #7: 14-bit value		Analogue i/p #8: 14-bit value	
7	Group 3 analogue i/ps read	Analogue i/p #9: 14-bit value		Analogue i/p #10: 14-bit value		Analogue i/p #11: 14-bit value		Analogue i/p #12: 14-bit value	
8	Group 4 analogue i/ps read	Analogue i/p #13: 14-bit value		Analogue i/p #14: 14-bit value		Analogue i/p #15: 14-bit value		Analogue i/p #16: 14-bit value	

Table 3.16 - Input/Response Messages (From Node to MCS)

Message No.	Description	Data Byte 1	2	3	4	5	6	7	8
9	Set-up analogue i/p read	Configuration byte	Analogue channels to read		Sample interval (units of 1/25 sec.)				
10	Set-up binary i/p read	Configuration byte	Binary channels to read		Read interval (units of 1/25 sec.)				

Table 3.17 - Output/Command Messages (From MCS to Node)

The configuration byte has the format:

Bit 7 (MSB)	6	5	4	3	2	1	0 (LSB)
					Read Binary Word #2	Read Binary Word #1	Poll/Auto-send

Table 3.18 - Configuration Byte

LSB set means the node will 'Auto-send' i.e. send the sampled values across CAN at intervals specified by the sampling interval. LSB cleared ('Poll') means the node will only send the values of the channel samples when polled by the MCS using Remote Transmit Request.

Bits 1 and 2 are only relevant to the binary word input channels. When set, the node will read the appropriate binary input word.

Channel Select bytes

These 16 bits specify which channels are to be read. A set bit indicates a channel is to be read (either automatically or when polled). LSB corresponds to analogue channel 1 and binary input 1; MSB corresponds to analogue channel 16 and binary input 16.

Sample Interval byte

Specifies the sampling interval in units of 1/25 second. Valid values are from 1 (sampling interval is 0.04 sec) to 255 (sampling interval is 10.2 sec). Samples required less frequently can be polled for.

3.7.3.4.7. CANbus Capacity

It is possible to estimate the data-transmission capability of the CANbus system, as shown below.

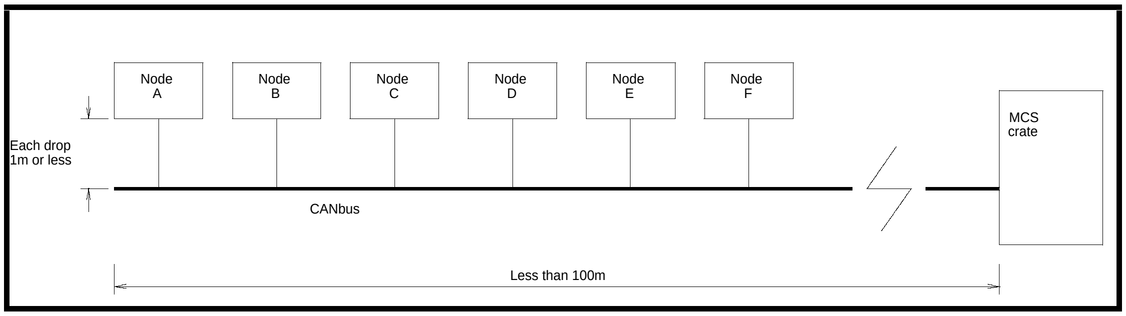


Figure 3.38 - The Monitoring CANbus system

For this configuration, a maximum bit-rate of 500kbit/second is possible. Doubling the bus length to 200m would result in a halving of the bit-rate to 250kbit/sec; this is an option if nodes have to be widely separated. Work carried out by the PCS group suggests that for greatest reliability and low sensitivity to improper bus termination, bus rates of either 250kbit/s or 125kbit/s be used. However, dropping to lower rates will prevent data transmission at the fastest sampling rates. Fortunately changing bus rate does not impact on the node hardware design.

The following cases illustrate typical data transmission situations across CANbus.

CASE 1

Six nodes, on a 500kbit/sec bus. Data required to be transmitted is considered to be made up of:

Source	Quantity	O/P Rate	Readings/sec.
16 analogue input channels per node	16*6=96	1Hz	96
Two 8-bit binary word inputs per node	2*6=12	1Hz	12
One 16-bit binary word per node	1*6=6	1Hz	6
		Total Readings/sec. on bus	**114**

Table 3.19 - Data Rates, 1 Hz Sampling

To calculate the number of bits that these readings represent we assume the worst case: that each message (i.e. each reading) has the maximum length a CAN message is allowed, 107 bits. Then, the no. of bits to be transmitted is:

$$b_{TR} = 114 \times 107 = 12198$$

Assuming a transmission rate *T* across the bus of 500kbit/sec. allows the bus loading *L* to be calculated as:

$$L = \frac{b_{TR}}{T} \times 100 = \frac{12198}{500 \times 10^3} \times 100$$

$$L = 2.4\,\%$$

If the MCS has to poll each node for its reading, this figure rises by 71%, giving:

$$L = 4.1\,\%$$

These bus loads are extremely low. This message rate could be handled easily even over a very much slower (longer) bus.

CASE 2

Six nodes, on a 500kbit/sec bus. Data required to be transmitted is considered to be made up of:

Source	Quantity	O/P Rate	Readings/sec.
16 analogue input channels per node	16*6=96	25Hz	2400
Two 8-bit binary word inputs per node	2*6=12	25Hz	300
One 16-bit binary word per node	1*6=6	25Hz	150
		Total Readings/sec. on bus	**2850**

Table 3.20 - Data Rates, 25 Hz Sampling

With the same assumptions as in Case 1:

$$b_{TR} = 2850 \times 107 = 304950$$

Assuming a transmission rate *T* across the bus of 500kbit/sec. allows the bus loading *L* to be calculated as:

$$L = \frac{b_{TR}}{T} \times 100 = \frac{304950}{500 \times 10^3} \times 100$$

$$L = 61\,\%$$

This is within limits for CANbus, which can theoretically operate at close to 100% bus load. If polling for each message is added, the bus load rises to:

$$L = 105\,\%$$

This clearly exceeds bus capacity, and therefore it is not possible to run six nodes on a 500kbit/sec. bus at a constant 25Hz message rate. Five nodes could be run at this rate however.

3.7.3.5. Enclosure & Connections

3.7.3.5.1. Housing

Each node will be housed in an ABS box of a similar type to that used by the Axial Support nodes. It will have a conductive coating applied internally to reduce EMI.

3.7.3.5.2. Physical Connections

The following physical connections are made to the node.

- CANbus (screened, twisted pair)
- 120VAC single-phase mains power
- ±12V external DC in (if required)
- Sensor and transducer connections (via two 37-way D-type connectors)

The sensors and transducers are not connected directly to the node box itself, but are routed via a breakout box as shown in Figure 3.39. This enables a node box to be easily removed without having to break a large number of connections.

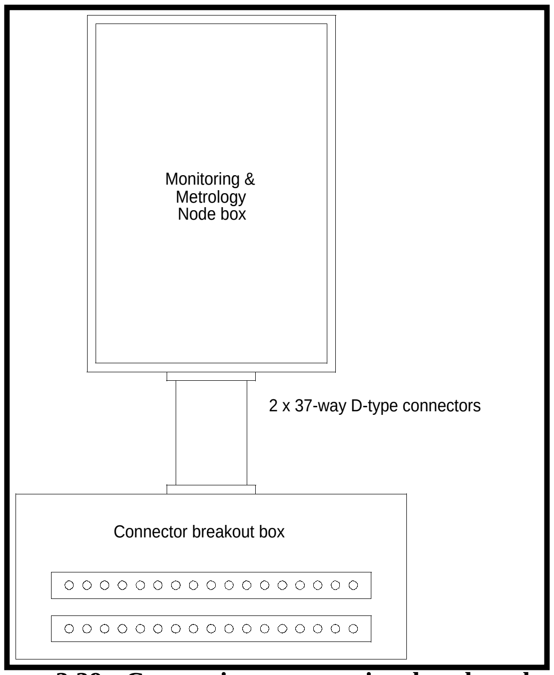


Figure 3.39 - Connecting sensors via a breakout box

The breakout box will accept two 37-way D-type connectors and make analogue and digital sensor connections available via PCB-mounted 45°-entry screw terminals.

4. INTERFACE DESCRIPTION

4.1. INTERNAL INTERFACES

The internal wiring of the VME enclosure is detailed in reference [7].

4.2. EXTERNAL INTERFACES

4.2.1. Hardware Interfaces

All of the external hardware interfaces are made through a series of 'D' type and BNC connectors mounted on the back panel of the VME enclosure. These are specified in detail in reference [7].

4.2.2. Software Interfaces

The following tables describe the EPICS process variables that will form the interface between the hardware and the software of the MCS.

Function	Description	Range	Frequency	EPICS Records	VME Board
Azimuth Drive Enable	Interlocks Azimuth Axis	0 - 1	≤ 50 Hz	2 Binary Outs	TTL
Azimuth Drive Condition	Status of Azimuth Axis	0 - 1	≤ 50 Hz	2 Binary Ins	TTL
Elevation Drive Enable	Interlocks Elevation Axis	0 - 1	≤ 50 Hz	2 Binary Outs	TTL
Elevation Drive Condition	Status of Elevation Axis	0 - 1	≤ 50 Hz	2 Binary Ins	TTL

Table 4.1 - The MCS Software Interface to the GIS

Function	Description	Range	Frequency	EPICS Records	VME Board
Az. Demand Positions	Demanded Azimuth positions	-182° : +362°	20 - 200 Hz	20 Analogue Outs	PMAC 1
Az. Demand Velocities	Demanded Azimuth velocities	± 2.0 °/sec	20 - 200 Hz	20 Analogue Outs	PMAC 1
Az. Current Position	Current Azimuth position	-188.25° : +367.67°	≤ 50 Hz	Analogue In	PMAC 1
Az. Current Velocity	Current Azimuth velocity	± 2.0 °/sec	≤ 50 Hz	Analogue In	PMAC 1
Az. Position Error	Azimuth position error	-549.67° : 550.25°	≤ 50 Hz	Analogue In	PMAC 1
Az. Motor Status	Motor servo status bits	24 bits	≤ 50 Hz	Status	PMAC 1
Az. CS Status	Co-ordinate System status bits	24 bits	≤ 50 Hz	Status	PMAC 1
Az. Motor Currents	Azimuth motor currents	0-50 Amps	≤ 50 Hz	8 Analogue Ins	XYCOM-566 (1)
Az. Tachos	Azimuth tachogenerator o/p	± 100°/sec	≤ 50 Hz	8 Analogue Ins	XYCOM-566 (2)
El. Demand Positions	Demanded Elevation positions	2.73° : +90.02º	20 - 200 Hz	20 Analogue Outs	PMAC 2
El. Demand Velocities	Demanded Elevation velocities	± 0.75°/sec	20 - 200 Hz	20 Analogue Outs	PMAC 2
El. Current Position	Current Elevation position	0° : +92.73°	≤ 50 Hz	Analogue In	PMAC 2
El. Current Velocity	Current Elevation velocity	± 0.75°/sec	≤ 50 Hz	Analogue In	PMAC 2
El. Position Error	Elevation position error	-90° : 90.02°	≤ 50 Hz	Analogue In	PMAC 2
El. Motor Status	Motor servo status bits	24 bits	≤ 50 Hz	Status	PMAC 2
El. CS Status	Co-ordinate System status bits	24 bits	≤ 50 Hz	Status	PMAC 2
El. Motor Currents	Elevation motor currents	0-50 Amps	≤ 50 Hz	4 Analogue Ins	XYCOM-566 (1)
El. Tachos	Elevation tachogenerator o/p	± 100°/sec	≤ 50 Hz	4 Analogue Ins	XYCOM-566 (2)

Table 4.2 - The MCS Software Interface to the Mount Main Servos

Function	Description	Range	Frequency	EPICS Records	VME Board
Az. VE Set-Up	Azimuth VE Set-Up word	24 bits	≤ 50 Hz	MultiBit Out	PMAC 1
Az. VE Output	Azimuth VE output	48 bits	≤ 50 Hz	MultiBit In	PMAC 1
Az. Tape Encoders	Raw tape head outputs	48 bits	≤ 50 Hz	4 MultiBit Ins	PMAC 1
Az. FDE	FDE incremental encoder readout	48 bits	≤ 50 Hz	MultiBit In	PMAC 1
Az. Last Fiducial	The last azimuth fiducial passed	1-72	≤ 50 Hz	MultiBit In	PMAC 1
Az. Inner Topple Bracket	Position of inner topple bracket	0 - 1	≤ 50 Hz	Binary In	PMAC 1
Az. Outer Topple Bracket	Position of outer topple bracket	0 - 1	≤ 50 Hz	Binary In	PMAC 1
El. VE Set-Up	Elevation VE Set-Up word	24 bits	≤ 50 Hz	MultiBit Out	PMAC 2
El. VE Output	Elevation VE output	48 bits	≤ 50 Hz	MultiBit In	PMAC 2
El. Tape Encoders	Raw tape head outputs	48 bits	≤ 50 Hz	4 MultiBit Ins	PMAC 2
El. FDE	FDE incremental encoder readout	48 bits	≤ 50 Hz	MultiBit In	PMAC 2
El. Last Fiducial	The last elevation fiducial passed	1-22	≤ 50 Hz	MultiBit In	PMAC 2
El. Tilt Switch	Position of elevation tilt switch	0 - 1	≤ 50 Hz	Binary In	PMAC 2
El. Translation	Output of elevation LVDT	0 -10mm	≤ 50 Hz	Analogue In	PMAC 2
El. Striker Bracket	Position of elevation striker bracket	0 - 1	≤ 50 Hz	Binary In	PMAC 2

Table 4.3 - The MCS Software Interface to the Encoder Hardware

Function	Description	Range	Frequency	EPICS Records	VME Board
Axial Demand Positions	Axial CW Demand	0 - 970 mm	≤ 50 Hz	2 String Outs	PMAC 3
Cross-Axial Demand Pos.	Cross-Axial Demand	0 - 970 mm	≤ 50 Hz	2 String Outs	PMAC 3
CW Motor Status	Motor servo status bits	24 bits	≤ 50 Hz	4 Status	PMAC 3
CW CS Status	Co-ordinate System status bits	24 bits	≤ 50 Hz	4 Status	PMAC 3
Axial CW Positions	Current positions of Axial CWs	0 - 970 mm	≤ 50 Hz	2 Analogue Ins	PMAC 3
Cross-Axial CW Positions	Current positions of Cross-Axial CWs	0 - 970 mm	≤ 50 Hz	2 Analogue Ins	PMAC 3

Table 4.4 - The MCS Software Interface to the counterweight hardware

Function	Description	Range	Frequency	EPICS Records	VME Board
SW Motor Status	Motor servo status bits	24 bits	≤ 50 Hz	3 Status	PMAC 3
SW CS Status	Co-ordinate System status bits	24 bits	≤ 50 Hz	3 Status	PMAC 3
SW Positions	Differential positions wrt associated axis	TBD mm	≤ 50 Hz	3 Analogue Ins	PMAC 3
SW Drive Currents	Service-wrap motor drive currents	0-50 Amps	≤ 50 Hz	4 Analogue Ins	XYCOM-566 (1)

Table 4.5 - The MCS Software Interface to the service wrap hardware

Function	Description	Range	Frequency	EPICS Records	VME Board
Analogue Monitors	Sensors connected via the Monitoring subsystem's analogue inputs	±10 V	1 Hz	16 Analogue Ins per Node	CAN VIPC-610
Binary Monitors	Sensors connected via the Monitoring subsystem's binary inputs	0 - 1	1 Hz	16 Binary Ins per Node	CAN VIPC-610
Word Monitors	Sensors connected via the Monitoring subsystem's digital word inputs	8 bits	1 Hz	2 MultiBit Ins per Node	CAN VIPC-610

Table 4.6 - The MCS Software Interface to the monitoring hardware

Function	Description	Range	Frequency	EPICS Records	VME Board
Time Health	Holds the health of the time system	0 - 2	1 Hz	SIR	Bancomm 635/7
Time Simulation	Is time real or simulated?	0 - 1	Variable	Binary Out	Bancomm 635/7
Access to Time Library	Subroutine Calls to timeLib	N/A	Variable	genSub	Bancomm 635/7
Time Log	Logged messages about time system	N/A	Variable	DHS History	Bancomm 635/7

Table 4.7 - The MCS Software Interface to the Gemini Time System

5. APPENDICES

5.1. APPENDIX A - PMAC User Written DSP Code

5.1.1. Heidenhain Interpolation And Compensation Code

Listed below is a partial code implementation of the Motorola 56000 DSP code to perform the interpolation and compensation required for the Heidenhain tape encoding system. Note that currently the code only services one head. When compiled, this code occupies 144 locations of P memory (432 bytes).

```
;*********************************************************************************;
;JOHN WILKES                       HEIDCOMP.ASM                    25 OCTOBER 1996  ;
;                 HEIDENHAIN TAPE ENCODER INTERPOLATION AND COMPENSATION            ;
;                                                                                   ;
; This file contains the 56000 assembler code to implement interpolation and        ;
; compensation of the signals from Heidenhain read heads. The code is designed to   ;
; run on a PMAC card as a user written servo routine. Because of this, the code     ;
; below protects all registers except A, B, X and Y and only one level of the stack ;
; is used. The routine is terminated with a jump to location P:$23. The data memory ;
; used is the free memory between L:$7F0..$7FF and X and Y memory within the P-     ;
; variable memory space. A memory map that shows this memory usage in detail, along ;
; with flowcharts explaining the code, can be found in separate documentation.      ;
;                                                                                   ;
; There are three values derived from each Heidenhain head. There are two quadrature;
; (Sine and Cosine) values that can be used to interpolate or determine absolute    ;
; position within a pitch. There is also a pitch count value that indicates the     ;
; incremental number of tape pitches. All three of these values are available as    ;
; outputs from PMAC’s encoder table. This routine takes these three values for each ;
; head and produces a single 24 bit integer which is the incremental position of the;
; tape head. The resolution of this output is 5 milliarcseconds, which means the    ;
; resolution is slightly different for each axis (azimuth / elevation) To be precise;
; the resolution is 0.116355283 microns for azimuth and 0.096962736 microns for     ;
; elevation (based upon nominal 9.6 m and 8 m diameters for Az and El respectively).;
; Note that these values for resolution will allow roll over of the 24 bit word but ;
; this will be dealt with by the virtual encoder code.                              ;
;                                                                                   ;
; Documented spaces have been included in the code below to allow the virtual       ;
; encoder code to be added to this file.                                            ;
;                                                                                   ;
;*********************************************************************************;

;*********************************************************************************;
;               SYMBOL DEFINITIONS FOR INTERPOLATION AND COMPENSATION               ;
;*********************************************************************************;
ONE             EQU     $7FFFFF
STORE_R0        EQU     $770    ; Storage address for R0
STORE_R1        EQU     $771    ; Storage address for R1
STORE_R2        EQU     $772    ; Storage address for R2
STORE_R3        EQU     $773    ; Storage address for R3

BLANKING        EQU     4800    ; Analogue range around crossover point that is blanked
                                ; to avoid the one pitch error.
RES             EQU     0.671434916    ; 0.078125/R, where R is the required resolution
                                       ; of the result in microns.
                                       ; RES = 0.671434916 for Azimuth based on 9.6 m
                                       ; RES = 0.805721899 for Elevation based on 8 m
ENC_TABLE       EQU     $724    ; Encoder table pointer - For all head signals
TH_OUT          EQU     $1201   ; Pointer to output table
COMP_PARA       EQU     $1211   ; Pointer to compensation parameters
SAVE_A          EQU     $77E    ; Temporary storage for A register

ATAN_ARG        EQU     $77F    ; ARCTANGENT FUNCTION ARGUMENTS

                ORG     X:$1280               ; COEFFICIENTS OF ARCTANGENT SERIES
COEFF_0         DC      0,ONE,0,-1.0/3
COEFF_1         DC      @ATN(1),0.5,-0.25,0.125

; NOTE: SYMBOL DEFINITIONS FOR VIRTUAL ENCODER COULD BE INCLUDED HERE
```

```
;*****************************************************************************;
;                    CODE FOR INTERPOLATION AND COMPENSATION                  ;
;                                                                             ;
; Currently, this code works for one head only. However, by inclusion of some ;
; simple loop code as indicated, it can be easily expanded to 2 or 4 heads. Note ;
; the loop can not be implemented using the DO command since PMAC's restrictions on ;
; stack usage will not allow it.                                              ;
;                                                                             ;
;*****************************************************************************;
               ORG     P:$B800                      ; Locate Code
               MOVE    R0,X:STORE_R0                ; Save contents of R0
               MOVE    R1,X:STORE_R1                ; Save contents of R1
               MOVE    R2,X:STORE_R2                ; Save contents of R2
               MOVE    R3,X:STORE_R3                ; Save contents of R3

               ; HEIDENHAIN INTERPOLATION AND COMPENSATION
               MOVE    #ENC_TABLE,R1                ; R1 pointer to encoder table
               MOVE    #TH_OUT,R2                   ; R2 pointer to output table
               MOVE    #COMP_PARA,R3                ; R3 pointer to compensation
                                                    ; parameters

               ; ? Some kind of loop for all heads

               MOVE    X:(R1)+,X1                   ; Get sine analogue channel
               MOVE    X:(R1)+,X0                   ; Get cosine analogue channel
               MOVE    #0,B                         ; Clear offset (B)

               JSET    #23,X0,INTERP                ; Interpolate if cosine is -ve
               MOVE    X1,A                         ; And sine is NOT in
               MOVE    #BLANKING,Y0                 ; the blanking region
               CMPM    Y0,A          #0,A           ;
               JLT     CALC_RESULT                  ;

INTERP         MOVE    X,L:ATAN_ARG                 ; Save arguments and
               JSR     ATAN2                        ; perform Arctangent

               JCLR    #0,Y:(R2),CALC_RESULT        ; Skip compensation if turned off

COMP           JSR     APPLY_COMP                   ; Apply compensation to signals

               MOVE    A,L:SAVE_A                   ; Save uncomped interp. value
               MOVE    X,L:ATAN_ARG                 ; Save compensated arguments and
               JSR     ATAN2                        ; perform Arctangent

               MOVE    A,B                          ; Compensated interp. value to B
               MOVE    L:SAVE_A,A                   ; Restore Uncomped interp. value
               SUB     A,B           #0.5,Y0        ; Calculate offset

               CMPM    Y0,B                         ; Check to see whether offset
               JLT     CALC_RESULT                  ; overflows
               MOVE    #-1.0,Y0                     ; If so add 1 if offset is +ve
               JCLR    #23,B,CORRECT_OFFSET         ; and minus one if offset is
               MOVE    #ONE,Y0                      ; -ve
CORRECT_OFFSET ADD     Y0,B                         ;

CALC_RESULT    ADD     B,A           X:(R1)+,B      ; Add offset to interpolation
               REP     #14                          ; Shift result 14 bits to the
                ASR    A                            ; right to scale correctly

               REP     #4                           ; Shift pitch count 4 bits to
                ASL    B                            ; the left to scale correctly

               ADD     B,A           #RES,X0        ; Combine pitch count and interp.

               MOVE    A,X1                         ; Multiply by relevant factor
               MPY     X0,X1,A                      ; in order to leave result
                                                    ; in 5 milliarcsec resolution

               MOVE    A,X:(R2)+                    ; Store Result

               ; End Loop

               ; VIRTUAL ENCODER CODE COULD BE INCLUDED HERE

               MOVE    X:STORE_R0,R0                ; Restore contents of R0
               MOVE    X:STORE_R1,R1                ; Restore contents of R1
               MOVE    X:STORE_R2,R2                ; Restore contents of R2
               MOVE    X:STORE_R3,R3                ; Restore contents of R3
               JMP     <$23                         ; Return to PMAC Code
```

```
;*******************************************************************************;
;                                END OF MAIN CODE                               ;
;*******************************************************************************;

;*******************************************************************************;
;                  SUBROUTINE TO PERFORM ARCTANGENT FUNCTION                    ;
;                                                                               ;
; This subroutine finds the unambiguous angle between 0-360 degrees the sine and  ;
; cosine of which are given as arguments. The Arguments should be stored at the   ;
; memory location pointed to by ATAN_ARG. The Sine argument in X:ATAN_ARG and the ;
; Cosine argument in Y:ATAN_ARG. Upon completion, the result is stored in the A   ;
; accumulator in standard 56000 fractional notation. The result is a positive     ;
; fraction, zero represents 0 degrees and 1 represents 360 degrees. Upon          ;
; completion the B accumulator is cleared.                                        ;
;                                                                               ;
; The routine uses one of two series expansions of the arctangent function. For   ;
; Sine/Cosine less than one (result < 45 degrees) the expansion about 0 is used.  ;
; For Sine/Cosine greater than one (result > 45 degrees) the expansion about one  ;
; is used. Use of the two expansions, allows greater accuracy from                ;
; quite low order series. Third order series are used. Even so, the accuracy is   ;
; not brilliant and the error can be almost 0.5 degrees in places - This is       ;
; quite sufficient for the present application. A Higher order series could be    ;
; implemented by adding the extra code - a loop could easily be formed to allow   ;
; this.                                                                           ;
;                                                                               ;
; The A, B, X, Y and R0 registers will be modified by this routine. No stack      ;
; levels are used.                                                                ;
;                                                                               ;
;*******************************************************************************;

ATAN2          MOVE    Y:ATAN_ARG,B                      ; Get Absolute data into the
               ABS     B               X:ATAN_ARG,A      ; accumulators. Sine->A and
               ABS     A               B,X0              ; Cosine->B

               CMP     B,A             #0.5,Y0           ; Sine - Cosine
               TGT     A,B                               ; If Sine > Cosine then
               TGT     X0,A                              ; swap arguments to keep
               MOVE    B,X0                              ; Quotient less than one
               MOVEC   SR,Y1                             ; Save SR for later.

               AND     #$FE,CCR                          ; Perform one quadrant
               REP     #$18                              ; division - See 56000
               DIV     X0,A                              ; Manual under DIV instruction
               MOVE    A0,A                              ; Transfer result into A

               MOVE    #COEFF_0,R0                       ; Set R0 to point to the coeffs
                                                         ; for exp. about 0 series
               CMP     Y0,A            #ONE,Y0           ; If x < 0.5 then jump to
               JLT     POLY                              ; series calculation
               MOVE    #COEFF_1,R0                       ; Else prepare for exp. about
               SUB     Y0,A                              ; 1 series.

POLY           MOVE    A,X0                              ; X0 is x
               MOVE    X:(R0)+,A                         ; Move constant to result
               MOVE    X:(R0)+,Y0                        ; Move linear coeff. to Y0
               MAC     X0,Y0,A         X0,X1             ; Calc. linear term then add
                                                         ; to result. Make X1 = x

               ; Note that PMAC's stack restrictions do not allow the following
               ; to be written as a DO loop
               MPY     X1,X0,B         X:(R0)+,Y0        ; Put x^2 into X1 and Get
               MOVE    B,X1                              ; the x^2 coeff. in Y0
               MAC     X1,Y0,A                           ; Add term to result

               MPY     X1,X0,B         X:(R0)+,Y0                ; Put x^3 into X1 and Get
               MOVE    B,X1                                      ; the x^3 coeff. in Y0
               MAC     X1,Y0,A         #1/(@ASN(1)*2),X0         ; Add term to result

POLY_END       MOVE    A,X1                              ; Scale Result so to +/- pi
               MPY     X0,X1,A                           ; maps to +/- 1

               MOVEC   Y1,SR                             ; Restore SR from before
               JLT     QUAD_OK                           ; If Sine > Cosine then
               NEG     A               #0.5,Y0           ; A = pi/2 - A
               ADD     Y0,A

QUAD_OK        MOVE    L:ATAN_ARG,X                      ; Get Sine and Cosine values
               MPY     X0,X1,B         #-1.0,Y0          ; Multiply to find sign
               JPL     HALF_OK                           ; If sign is negative then
               NEG     A               #ONE,Y0           ; invert result. Set up correct
                                                         ; offset in Y0
```

```
HALF_OK        JCLR    #23,X0,FULL_OK               ; Test sign of Cosine value
               ADD     Y0,A                         ; If -ve then A = A + offset

FULL_OK        ASR     A              #ONE,Y0       ; Convert range to 0-2*pi maps
               JPL     ATAN2_STOP                   ; to 0-1
               ADD     Y0,A           #0,B          ; Clear B (offset)

ATAN2_STOP     RTS

;*************************************************************************************;
;                      SUBROUTINE TO APPLY HEYDEMANN COMPENSATION                     ;
;                                                                                     ;
; This routine is used to calculate Heydemann compensated Sine and Cosine signals.    ;
; Before execution, the X register should contain the uncompensated signals, Sine in ;
; X1 and Cosine in X0, and the compensation parameters should be stored in a table    ;
; in X memory pointed to by R3. The values should be stored in standard 56000         ;
; fractional notation and in the following order - P/8, Q/8, G/8, sin(A) and         ;
; 1/(8*cos(A)). Note that all except sin(a) have a divide by 8 scaling factor to      ;
; keep the calculations easy, and allow multipliers greater than one. Note also that ;
; the reciprocal of the cos(A) parameter is used to avoid the need to do division.    ;
; Upon completion, the compensated signals are stored in the X register, Sine in X1   ;
; and Cosine in X0.                                                                   ;
;                                                                                     ;
; The operations performed are:                                                       ;
; Sine_Comp = Sine - P                                                                ;
; Cosine_Comp = (Sine_Comp*Sin(A) + G*(Cosine-Q)) / cos(A)                            ;
;                                                                                     ;
; The B, X, Y0 and R3 registers will be modified by this routine. No stack levels     ;
; are used.                                                                           ;
;                                                                                     ;
;*************************************************************************************;

APPLY_COMP     MOVE    X1,B                         ; Move Sine to B
               MOVE    X:(R3)+,Y0                   ; Get P parameter
               SUB     Y0,B           X:(R3)+,Y0    ; Calculate Sine_Comp and get
               MOVE    B,X1                         ; q parameter

               MOVE    X0,B                         ; Move cosine to B
               SUB     Y0,B           X:(R3)+,Y0    ; Calculate intermediate result
               MOVE    B,X0                         ; and get G parameter

               MPY     Y0,X0,B        X:(R3)+,Y0    ; Calculate intermediate result
               REP     #3                           ; and get Sa parameter. Then
                ASL    B                            ; correct the result by shifting

               MAC     Y0,X1,B        X:(R3)+,Y0    ; Calculate intermediate result
               MOVE    B,X0                         ; and get 1/Ca parameter

               MPY     Y0,X0,B                      ; Calculate final result for
               REP     #3                           ; Cosine_Comp and correct by
                ASL    B                            ; shifting

               MOVE    B,X0                         ; Store Cosine_Comp in X0

APPLY_COMP_STOP        RTS

                       END
```

5.1.2. Virtual Encoder Code

The following is the current partial code implementation for the Gemini virtual encoder. It is written in Motorola DSP56000 assembly language. When compiled, this code occupies 163 locations of P memory (489 bytes).

```
;-----------------------------------------------------
;VE_MAIN.ASM

;Current development code for the Gemini telescope
;Mount Control System Virtual Encoder.
;(56000 DSP running on PMAC-VME.)

;Written by C. Carter, Royal Greenwich Observatory, UK
;Last modified 7th November 1996

;-----------------------------------------------------
;Variable Definitions:
```

```
SETUP          EQU $1300      ;24-bit SETUP word
RESULT         EQU $1301      ;Lower 24 bits of the VE output
VE_OUT         EQU $1302      ;Full 48-bit output position

FDE            EQU $723       ;Encoder table FDE value
TH1            EQU $1201      ;Unextended, compensated TH1 value
TH2            EQU $1202      ;Unextended, compensated TH2 value
TH3            EQU $1203      ;Unextended, compensated TH3 value (Az.)
TH4            EQU $1204      ;Unextended, compensated TH4 value (Az.)

VE_FDE         EQU $1303      ;Position according to FDE
VE_TH1         EQU $1304      ;Extended, compensated TH1 value
VE_TH2         EQU $1305      ;          "           TH2 value
VE_TH3         EQU $1306      ;          "           TH3 value
VE_TH4         EQU $1307      ;          "           TH4 value

VE_TRNS1       EQU $1308      ;El. translation sensor 1
VE_TRNS2       EQU $1309      ;El. translation sensor 2
VE_TRNS3       EQU $130A      ;El. translation sensor 3
VE_TRNS4       EQU $130B      ;El. translation sensor 4

VE_EL1TR       EQU $130C      ;El. corrected position (head 1)
VE_EL2TR       EQU $130D      ;El. corrected position (head 2)

LAST_FIDU_POS  EQU $130E      ;VE o/p at last fiducial pass
LAST_FIDU_ID   EQU $130F      ;ID byte of last-passed fiducial
FIDU_FLAG      EQU $1310      ;Status of HMFL1 after fiducial pass
LAST_HMFL1     EQU $1311      ;Value of HMFL1 500us ago

;------------------------------------------------------
;Define PMAC variable locations

HMFL1          EQU $C000      ;Motor 1 Home Flag locn. (Bit 20)
FIDU_JOPT      EQU $FFC2      ;JOPT (J5) inputs (Y, bits 0-7)

;------------------------------------------------------
;SETUP bit masks

MODE_MASK      EQU $07        ;Mask for B0-B2 (Mode)
ALGO_MASK      EQU $38        ;Mask for B3-B5 (Algorithm)
M_MASK         EQU $1C0       ;Mask for B6-B8 (M)
N_MASK         EQU $E00       ;Mask for B9-B11   (N)

;------------------------------------------------------
;HMFL1 bit mask

HMFL1_MASK     EQU $100000
JOPT_MASK      EQU $FF

;------------------------------------------------------
;Location to store volatile PMAC register(s)

SAVE_R0        EQU $7F0       ;Save locn. for R0

;------------------------------------------------------
;Define start of VE code in PMAC Program memory.
;Program code area is from P:$B800 to P:$BBFF
;(May be moved up to allow for Compensation code)

       ORG     P:$B800

;Save register(s) before the code starts

       MOVE R0,X:SAVE_R0

;Clear the registers and accumulators used by PMAC.
;At the start of each servo-cycle, PMAC sets:
;'A' Accumulator: desired velocity (48 bits)
;'B' Accumulator: desired position (48 bits)
;'X' Register: actual position (48 bits)
;'Y1' Register: Ix08 value (Mot. X Pos. Scale Factor)

       CLR A
       CLR B

;------------------------------------------------------
;Apply scaling to FDE position value

;------------------------------------------------------
;Position-extend (24->48 bits) all encoder values
```

```
;------------------------------------------------------
;Write values to VE_FDE and VE_TH1-4

;------------------------------------------------------
;Extract Mode bits

        MOVE X:SETUP,A1
        MOVE #MODE_MASK,R0
        MOVE R0,X0
        AND X0,A

;------------------------------------------------------
;Jump to Diagnostic mode?

        MOVE #$01,B1
        CMP B,A
        JEQ DIAG_MODE  ;If Bit 1 set then enter DIAG_MODE

;If no match, default to ’normal’ mode

;Set Echo mode bits for EPICS (Normal Mode=000)

        BCLR #21,X:SETUP
        BCLR #22,X:SETUP
        BCLR #23,X:SETUP


;-----------------------------------------------------
;Extract Algorithm bits

        MOVE X:SETUP,A1
        MOVE #ALGO_MASK,R0
        MOVE R0,X0
        AND X0,A

;------------------------------------------------------
;Jump to FDE/Tape N/Tape N,M/All Heads?

        MOVE #$08,B1
        CMP B,A
        JSEQ TAPE_N    ;Run the TAPE_N code

        MOVE #$10,B1
        CMP B,A
        JSEQ TAPE_MN   ;Run the TAPE M & N code

        MOVE #$18,B1
        CMP B,A
        JSEQ ALL_HEADS ;Run the ALL HEADS code

;If no match, default to the FDE algorithm

        JMP FDE_ONLY

;------------------------------------------------------
;Check if axis has passed a fiducial

FIDU_CHECK

        MOVE X:HMFL1,A1               ;Mask all but bit 20 of HMFL1
        MOVE #HMFL1_MASK,R0   ;and store the value of the bit
        MOVE R0,X0
        AND X0,A
        MOVE A1,X:FIDU_FLAG

        MOVE X:LAST_HMFL1,B1  ;Ditto for the previous HMFL1 value
        MOVE #HMFL1_MASK,R0
        MOVE R0,X0
        AND X0,B

        CMP B,A
        JEQ FIDU_NOT_PASSED   ;If the bits are equal, a fiducial
                              ;has not been passed

FIDU_PASSED

        MOVE L:VE_OUT,A10             ;Store VE_OUT in LAST_FIDU_POS
        MOVE A10,L:LAST_FIDU_POS

        MOVE Y:FIDU_JOPT,A1
        MOVE #JOPT_MASK,R0
        MOVE R0,X0
        AND X0,A                ;Mask off all but bits 0-7 of FIDU_JOPT
        MOVE A1,X:LAST_FIDU_ID          ;Store ID of this fiducial

        BSET #20,X:SETUP        ;Set the FIDUCIAL PASSED bit in STATUS

FIDU_NOT_PASSED

        MOVE X:HMFL1,A1
        MOVE A1,X:LAST_HMFL1

;-----------------------------------------------------
;Clean up and exit. Last one out, please turn off the lights.

EXIT
        MOVE X:SAVE_R0,R0       ;Restore the R0 register

        JMP <$23                ;PMAC takes over from here

;-----------------------------------------------------
;-----------------------------------------------------
;Special Mode routines follow: so far, Diagnostic is the
;only one.

;-----------------------------------------------------
;Diagnostic mode routines

DIAG_MODE

;Set Echo mode bits for EPICS (Diag. Mode=001)
        BSET #21,X:SETUP
        BCLR #22,X:SETUP
        BCLR #23,X:SETUP

;Diagnostic routines here...
;Could include mirroring PMAC status bits, VE or
;Comp. code flags and/or variables to EPICS

        JMP EXIT

;-----------------------------------------------------
;-----------------------------------------------------
;Encoder handling subroutines follow

;-----------------------------------------------------
FDE_ONLY

        BCLR #20,X:SETUP        ;Echo Algorithm bits in SETUP (000)
        BCLR #19,X:SETUP
        BCLR #18,X:SETUP

        MOVE L:VE_FDE,A10       ;Loads VE_FDE into lower 48 bits of A
        MOVE A10,L:VE_OUT       ;Loads lower 48 bits of A into VE_OUT
        JSR UPDATE_LOW
        JMP FIDU_CHECK

;-----------------------------------------------------
TAPE_N

; Note: Elevation version includes translation sensors

        BCLR #20,X:SETUP        ;Echo Algorithm bits in SETUP (001)
        BCLR #19,X:SETUP
        BSET #18,X:SETUP

        MOVE X:SETUP,A1
        MOVE #N_MASK,R0
        MOVE R0,X0
        AND X0,A

;Check which head to read

        MOVE #$01,B1
        CMP B,A
        JSEQ READ_VE_TH1

        MOVE #$02,B1
        CMP B,A
        JSEQ READ_VE_TH2

        MOVE #$03,B1
        CMP B,A
        JSEQ READ_VE_TH3

        MOVE #$04,B1
        CMP B,A
        JSEQ READ_VE_TH4

;Routines to read appropriate tape head

READ_VE_TH1

        MOVE L:VE_TH1,A10
        JMP TAPE_N_EXIT

READ_VE_TH2

        MOVE L:VE_TH2,A10
        JMP TAPE_N_EXIT

READ_VE_TH3

        MOVE L:VE_TH3,A10
        JMP TAPE_N_EXIT

READ_VE_TH4

        MOVE L:VE_TH4,A10
        JMP TAPE_N_EXIT

;Save the head value in VE_OUT

TAPE_N_EXIT

        MOVE A10,L:VE_OUT
        JSR UPDATE_LOW
        JMP FIDU_CHECK

;-----------------------------------------------------
TAPE_MN

        BCLR #20,X:SETUP       ;Echo Algorithm bits in SETUP (010)
        BSET #19,X:SETUP
        BCLR #18,X:SETUP

;Tape_MN routine here. Differs for Azimuth & Elevation.

        JMP FIDU_CHECK

;-----------------------------------------------------
ALL_HEADS

        BCLR #20,X:SETUP       ;Echo Algorithm bits in SETUP (011)
        BSET #19,X:SETUP
        BSET #18,X:SETUP

;ALL_HEADS routine here. Differs for Azimuth & Elevation.

        JMP FIDU_CHECK

;-----------------------------------------------------
;-----------------------------------------------------
;Transfer lower 24 bits of VE_OUT (in A10) into RESULT

UPDATE_LOW

        MOVE A0,X:RESULT
        RTS
;-----------------------------------------------------
;-----------------------------------------------------
        END
```