E U R O P E A N S O U T H E R N O B S E R V A T O R Y

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral
Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

LA SILLA OBSERVATORY

2P2 TCS Software Drivers

Doc. No.: 2P2-MAN-ESO-60100-0001

Issue: 1.1

Date: 3-April-2000

Prepared: J. Alonso

Name Date Signature

Approved: G. Andreoni

Name Date Signature

Released: G. Andreoni

Name Date Signature

CHANGE RECORD

Revision	Date	Section/Paragraph	Remarks
Issue 1.0	13/8/1998	All	First Issue
Issue 1.1	3/4/2000	Several	Feedback from FRI + revision by JAL

1. INTRODUCTION

1.1. PURPOSE AND SCOPE

This document contains the prototypes together with a description of all the functions written under vxWorks for controlling the 2p2 meter telescope. Among these functions are included the auxiliary system and environmental sensing.

1.2. REFERENCE AND APPLICABLE DOCUMENTS

1 Telescope Control Module TCM (Hardware schematics)

1.3. ACRONYMS AND ABBREVIATIONS

BCD	Binary Coded Decimal
TCMDIG	Telescope Control Module Digital
TCMANA	Telescope Control Module Analogue
LCU	Local Control Unit
CPU	Central Processing Unit

2. TCM_Initialize()

NAME

TCM_Initialize() - hardware initialisation

SYNOPSIS

STATUS TCM_Initialize(void);

DESCRIPTION

This function checks the various hardware subsystems initialising the axis motion processors together with the associated peripherals within the TCMDIG and TCMANA.

RETURNS

OK or ERROR

ERRNO

S_tcsLib_ALPHA_ROD260_MALFUNCTION
Self explanatory
S_tcsLib_DELTA_ROD260_MALFUNCTION
Self explanatory

CAVEATS

The state of the TCM is reflected by a global variable of the enumerated type **modstat.** After the TCMDIG initialisation the state should be **Analog** or **Digital** and the **TCM_Initialize()** function should return **OK.** In case of problems the state remains **TCM_not_initialize** and the pertinent error number is set returning **ERROR.**

3. Initialize()

NAME

Initialize() - axis initialisation

SYNOPSIS

STATUS Initialize(axis)
ax axis; /* 0 alpha, 1 delta */

DESCRIPTION

This function initialises the specified axis. The initialisation procedure is accomplished in three stages;

1- Performs all the error checking (TCMDIG initialisation, TCM mode, power amplifiers on state and the axis absolute position) and defines the actual position as home position, enables the reference point interrupt detection and moves the axis very far away.
2- Starts when the reference point interrupt is detected. The axis is instructed to decelerate and stop, a move is done to the exact reference point and the reference point is defined as home position.
3- The absolute position is measured and a preset is done to the absolute zero position, them a offset is added, if specified, by preseting the axis and defining the zero position again.

RETURNS

OK or ERROR

ERRNO

S_tcsLib_VALUE_OUT_OF_RANGE
The specified axis does not exist.
S_tcsLib_TCM_NOT_INITIALIZED
Self explanatory
S_tcsLib_TCM_ANALOG_MODE
Self explanatory
S_tcsLib_ALPHA_PRESET_IN_PROGRESS
Re-initialisation can only be requested in **track** and **slew** modes.
S_tcsLib_DELTA_PRESET_IN_PROGRESS
Same as alpha
S_tcsLib_INIT_NOT_ALLOWED_AT_MORE_THAN_45_DEG
One or both axis are inclined more than 45deg. from zenith position.
S_tcsLib_ALPHA_DRIVE_KEPCO_POWER_OFF
Self explanatory
S_tcsLib_DELTA_DRIVE_KEPCO_POWER_OFF
Self explanatory

S_tcsLib_ALPHA_PRELOAD_KEPCO_POWER_OFF
Self explanatory
S_tcsLib_DELTA_PRELOAD_KEPCO_POWER_OFF
Self explanatory

CAVEATS
The state of the particular axis is reflected by a global variable of the enumerated type **axstat.** After the axis initialisation the state should change from **init** to **slew** and the **Initialize()** function returns **OK**. In case of an error occurrence there are two possibilities:

1- The state does not change and the **Initialize()** function sets the pertinent error number in the execution context (e.g. the vxWorks shell) and returns **ERROR**.
2- The **Initialize()** function returns **OK** and the state remains **init** several seconds but suddenly the state goes back to **not_initialized**. In this case the error number could be imported to the execution context by using **errnoOfTaskGet()** from **init1_alpha_ID** and/or **init1_delta_ID**.

4. Position()

NAME
Position() - read axis position

SYNOPSIS
int Position(axis)
ax axis; /* 0 alpha, 1 delta */

DESCRIPTION
This function reads the absolute position of the specified axis. It is possible to call this function up to 1000 times per second when the low - level drivers are alone in the LCU. However the calling rate is limited by higher level programs running together in the LCU.

RETURNS
An integer value representing the actual axis position.

ERRNO
N/A

CAVEATS
The position value is only meaningful if the axis is initialised. When the axis is not initialised the returned value is always zero.

5. Preset()

NAME

Preset() - move the axis to a specified position

SYNOPSIS

STATUS Preset(axis, pos, vel)
ax axis; /* 0 alpha, 1 delta */
int pos; /* position */
int vel; /* speed */

DESCRIPTION

This function loads position and speed in the axis control processor, and then starts the motion of the specified axis. It is valid to call this function during **preset, track** and **slew** axis states up to 1000 times per second. At the end of a succesful move the status should change from **preset** to **slew**. If the move fails the status should change from **preset** to **not_initialized**.

RETURNS

OK or ERROR.

ERRNO

S_tcsLib_VALUE_OUT_OF_RANGE
The specified axis does not exist.
S_tcsLib_TCM_NOT_INITIALIZED
Self explanatory
S_tcsLib_TCM_ANALOG_MODE
Self explanatory
S_tcsLib_ALPHA_AXIS_NOT_INITIALIZED
Self explanatory
S_tcsLib_ALPHA_SLEW_SPEED_OUT_OF_RANGE
Specified **vel** is greater than **servo1.max_slew_speed**
S_tcsLib_DELTA_AXIS_NOT_INITIALIZED
Self explanatory
S_tcsLib_DELTA_SLEW_SPEED_OUT_OF_RANGE
Specified **vel** is greater than **servo2.max_slew_speed**
S_tcsLib_ALPHA_DRIVE_KEPCO_POWER_OFF
Self explanatory
S_tcsLib_DELTA_DRIVE_KEPCO_POWER_OFF
Self explanatory
S_tcsLib_ALPHA_PRELOAD_KEPCO_POWER_OFF
Self explanatory
S_tcsLib_DELTA_PRELOAD_KEPCO_POWER_OFF
Self explanatory

CAVEATS

In the error occurrence case there are two possibilities:

1- The state of the axis does not change and the **Preset()** function sets the pertinent error number in the context of the calling task (e.g. the vxWorks shell) and returns **ERROR**.
2- **Preset()** returns **OK** and the axis state remains **preset** several seconds but suddenly the state change to **not_initialized**. In this case the error number must be imported to the execution context using **errnoOfTaskGet()** from **pos_err_al_ID** for alpha and **pos_err_de_ID** for delta.

6. Mode()

NAME

Mode() – track/slew modes selection

SYNOPSIS

```
STATUS Mode(axis, mode)
    ax axis; /* 0 alpha, 1 delta */
    mod mode; /* 0 SLEW, 1 TRACK */
```

DESCRIPTION

This function selects the track and slew operation modes for a particular axis.

RETURNS

OK or ERROR.

ERRNO

S_tcsLib_VALUE_OUT_OF_RANGE
    The specified axis does not exist.
S_tcsLib_TCM_NOT_INITIALIZED
    Self explanatory
S_tcsLib_TCM_ANALOG_MODE
    Self explanatory
S_tcsLib_ALPHA_AXIS_NOT_INITIALIZED
    Self explanatory
S_tcsLib_ALPHA_PRESET_IN_PROGRESS
    During preset the mode can not be changed to track.
S_tcsLib_DELTA_AXIS_NOT_INITIALIZED
    Self explanatory
S_tcsLib_DELTA_PRESET_IN_PROGRESS
    During preset the mode can not be changed to track.

CAVEATS

A preset can be aborted by **Mode(axis,SLEW)**, after executing, the particular axis will decelerate and stop.

7. Track()

NAME

Track() - changes tracking velocity

SYNOPSIS

STATUS Track(axis, vel)
ax axis; /* 0 alpha, 1 delta */
int vel; /* velocity */

DESCRIPTION

This function allows change the tracking velocity of the specified axis. It is particularly useful for applying offsets or minute changes in position.

RETURNS

OK or ERROR.

ERRNO

S_tcsLib_VALUE_OUT_OF_RANGE
The specified axis does not exist.
S_tcsLib_TCM_NOT_INITIALIZED
Self explanatory
S_tcsLib_TCM_ANALOG_MODE
Self explanatory
S_tcsLib_ALPHA_AXIS_NOT_INITIALIZED
Self explanatory
S_tcsLib_ALPHA_TRACK_SPEED_OUT_OF_RANGE
The absolute value of **vel** is greater than **servo1.max_track_speed**
S_tcsLib_ALPHA_TRACK_MODE_NOT_SELECTED
Self explanatory
S_tcsLib_DELTA_AXIS_NOT_INITIALIZED
Self explanatory
S_tcsLib_DELTA_TRACK_SPEED_OUT_OF_RANGE
The absolute value of **vel** is greater than **servo2.max_track_speed**
S_tcsLib_DELTA_TRACK_MODE_NOT_SELECTED
Self explanatory

CAVEATS

Please refer to the appendix for the allowed maximum tracking velocity.

8. Handset()

NAME

Handset() - reads the digital handset

SYNOPSIS

int Handset(void);

DESCRIPTION

reads the digital handset port.

RETURNS
An integer with the following bit distribution;
0 = delta -
1 = delta +
2 = alpha –
3 = alpha +
4 = guide
5 = offset
6 = set
7 – 31 = don't care

ERRNO
N/A

CAVEATS
N/A

9. Focus_preset()

NAME
Focus_preset() – Moves M2 mirror to a specified position

SYNOPSIS
STATUS Focus_preset(pos)
int pos;

DESCRIPTION
This function presets the M2 mirror to any position within the allowed range. Please refer to the appendix for the actual range and limit numbers.

RETURNS
OK or ERROR.

ERRNO
S_tcsLib_FOCUS_PRESET_ACTIVE
Before instructing a new motion the focus mechanism must be on position.
S_tcsLib_VALUE_OUT_OF_RANGE
The number specified is outside the software limits.
S_tcsLib_FOCUS_UPPER_LIMIT
The mechanism is on the hardware limit.
S_tcsLib_FOCUS_LOWER_LIMIT
The mechanism is on the hardware limit.

CAVEATS
The software limits are set slightly before the hardware limits, this means that the upper and lower hardware limits should never be reached under normal operation.

The sole function of the hardware limits is protection against catastrophic malfunctions. The upper and lower limit values are specified in the appendix of this document.

10. Focus_position()

NAME

Focus_position() – returns the focus position

SYNOPSIS

STATUS Focus_position(void);

DESCRIPTION

This function returns the M2 mirror position.

RETURNS

OK or ERROR

ERRNO

S_tcsLib_FOCUS_UPPER_LIMIT
On the hardware limit
S_tcsLib_FOCUS_LOWER_LIMIT
On the hardware limit

CAVEATS

The software limits are set slightly before the hardware limits, this means that the upper and lower hardware limits should never be reached under normal operation. The sole function of the hardware limits is protection against catastrophic malfunctions. The upper and lower limit values are specified in the appendix of this document.

11. Mirror()

NAME

Mirror() – open and close the main mirror cover

SYNOPSIS

mirstat Mirror(action)
miract action; /* 0 mclose, 1 mopen, 2 mstatus */

DESCRIPTION

This function opens, close and gets the M1 main mirror cover status.

RETURNS

An enumerated variable of the type **mirstat**

Typedef enum {moving, closed, opened, ok, error, local} **mirstat**

ERRNO
S_tcsLib_MIRROR_COVER_MECHANICAL_PROBLEM
This message indicates that the main mirror cover mechanism got stuck and/or a limit switch failed.

CAVEATS
N/A

12. Rotator_position()

NAME
Rotator_position() – returns the rotator position

SYNOPSIS
STATUS Rotator_position(position)
char *position[];

DESCRIPTION
This function delivers the rotator position in a character string. A pointer to a string of at least 10 characters must be passed.

RETURNS
OK

ERRNO
N/A

CAVEATS
The rotator's position is delivered in degrees with a precision and warranted accuracy of a tenth of a degree.

13. Sidereal_time_sync()

NAME
Sidereal_time_sync() – returns the sidereal time from the Cerme clock

SYNOPSIS
int Sidereal_time_sync(void)

DESCRIPTION
This function waits for synchronisation with the next sidereal second and reads the Cerme clock.

RETURNS
An integer containing the seconds, minutes and hours in BCD format. The seconds comes on the least significant byte and so on.

ERRNO
N/A

CAVEATS
Because this function is used for synchronisation purposes it should be called bearing in mind that uses 100% of the CPU for a maximum time of 1 second.

14. Hours() Minutes() Seconds()

NAME

Hours()
Minutes()
Seconds() – functions for reading hours, minutes and seconds

SYNOPSIS

```
int Hours(void);
int Minutes(void);
int Seconds(void);
```

DESCRIPTION
Three non-blocking functions for reading sidereal hours, minutes and seconds from the Cerme clock.

RETURNS
An integer with the, hours minutes or seconds. BCD coded in the least significant byte.

15. Domepos()

NAME

Domepos() – returns the dome position

SYNOPSIS

```
int Domepos(void);
```

DESCRIPTION
This function reads and returns the dome azimuth. The dome micro controller is interrogated by the function via the RS-232 port 4 of the LCU's MV167 CPU.

RETURNS
An integer with the dome azimuth coded from 0 to 999 for the 0 to 360 degrees positions.

ERRNO
S_tcsLib_DOME_u_CONTROLLER_NOT_RESPONDING
dome u controller malfunction or power fail

CAVEATS
N/A

16. Domeset()

NAME
Domeset() – presets the dome to the specified position

SYNOPSIS
int Domeset(pos)
int pos;

DESCRIPTION
This function presets the dome to the specified azimuth of 0 to 999 which corresponds to 0 to 360 degrees. The command is sent to and the answer received from the dome micro controller via the RS-232 port 4 of the LCU's MV167 CPU.

RETURNS
OK or ERROR

ERRNO
S_tcsLib_VALUE_OUT_OF_RANGE
The specified value is not in the range 0 to 999

CAVEATS
N/A

17. Domestat()

NAME
Domestat() – returns the dome status

SYNOPSIS
dome_status Domestat(void);

DESCRIPTION
This function gets and returns the dome status. The command is sent to and the answer received from the dome micro controller via the RS-232 port 4 of the LCU's MV167 CPU.

RETURNS
A **dome_status**
typedef enum {DLOCAL, DREMOTE, DMOVING} **dome_status;**

ERRNO
S_tcsLib_DOME_u_CONTROLLER_NOT_RESPONDING
Dome micro controller not responding or power fail
S_tcsLib_UNKNOWN_STATUS_RECEIVED
Self explanatory

CAVEATS
N/A

18. Domestop()

NAME
Domestop() – stops the dome motion

SYNOPSIS
int Domestop(void);

DESCRIPTION
This function stops the dome motion. The command is sent to the dome micro controller via the RS-232 port 4 of the LCU's MV167 CPU.

RETURNS
OK

CAVEATS
N/A

19. Slitopen()

NAME
Slitopen()

SYNOPSYS
int Slitopen(void);
DESCRIPTION
This function opens the dome slit. The command is sent to the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS
OK

ERRNO
N/A

CAVEATS
N/A

20. Slitclose()

NAME
Slitclose() – closes the dome slit

SYNOPSIS
int Slitclose(void);

DESCRIPTION
This function closes the dome slit. The command is sent to the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS
OK

ERRNO
N/A

CAVEATS
N/A

21. Slitstop()

NAME
Slitstop() – stops the slit motion

SYNOPSIS
int Slitstop(void);

DESCRIPTION
This function stops the dome slit motion. The command is sent to the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.
RETURNS
OK

ERRNO
N/A

CAVEATS
N/A

22. Slitstat()

NAME

Slitstat() – returns the slit status

SYNOPSIS

slit_status Slitstat(void);

DESCRIPTION

This function reads and returns the slit status. The command is sent to and the answer is received from the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS

A **slit_status**
typedef enum {SOPEN, SLCLOSED, SLMOVING, SLSTOP} **slit_status;**

ERRNO

S_tcsLib_SLIT_WSCR_u_CONTROLLER_NOT_RESPONDING
Slit micro controller malfunction or power fail
S_tcsLib_UNKNOWN_STATUS_RECEIVED
Self explanatory

CAVEATS

N/A

23. Upwspos()

NAME

Upwspos() – returns the upper wind screen position

SYNOPSIS

int Upwspos(void);

DESCRIPTION

This function reads and returns the upper wind screen position, the valid range of positions is from 0 to 999. The command is sent to and the answer received from the slit micro controller via RS-232 port 3 of the LCU's MV167 CPU.

RETURNS

An integer with the upper wind screen position.

ERRNO

S_tcsLib_SLIT_WSCR_u_CONTROLLER_NOT_RESPONDING
Dome micro controller malfunction or power fail

CAVEATS
N/A

24. Upwsset()

NAME

Upwsset() – sends the upper wind screen to the specified position

SYNOPSIS

int Upwsset(pos)
int pos;

DECRIPTION
This function sends the upper windscreen to the specified position. The valid range of positions is from 0 to 999. The command is sent to the slit micro controller via RS-232 port 3 of the LCU's MV167 CPU.

RETURNS
OK or ERROR

ERRNO
S_tcsLib_VALUE_OUT_OF_RANGE
The specified value is not in the range 0 to 999

CAVEATS
N/A

25. Upwsstop()

NAME

Upwsstop() – stops the upper windscreen motion

SYNOPSIS

int Upwsstop(void);

DESCRIPTION
This function stops the upper windscreen motion. The command is sent to the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS
OK

ERRNO
N/A

CAVEATS
N/A

26. Upwsstat()

NAME

Upwsstat() – reads the upper windscreen status

SYNOPSIS

upper_wscr_status Upwsstat(void);

DESCRIPTION

This function gets and returns the upper windscreen status. The command is sent to and the answer received from the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS

A **upper_wscr_status**
typedef enum {UWSMOVING, UWSSTOP} **upper_wscr_status**

ERRNO

S_tcsLib_SLIT_WSCR_u_CONTROLLER_NOT_RESPONDING
Dome micro controller malfunction or power fail
S_tcsLib_UNKNOWN_STATUS_RECEIVED
Self explanatory

CAVEATS

N/A

27. Lowspos()

NAME

Lowspos() – reads the lower wind screen position

SYNOPSIS

int Lowspos(void);

DESCRIPTION

This function reads and returns the lower wind screen position, the valid range of positions is from 0 to 999. The command is sent and the answer received from the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS

An integer with the position

ERRNO

S_tcsLib_SLIT_WSCR_u_CONTROLLER_NOT_RESPONDING

Dome micro controller malfunction or power fail

CAVEATS
N/A

28. Lowsset()

NAME
Lowsset() – sends the lower wind screen to the specified position

SYNOPSIS
int Lowsset(pos)
int pos;

DESCRIPTION
This function sends the lower windscreen to the specified position. The valid range of positions is from 0 to 999. The command is sent to the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS
OK or ERROR

ERRNO
S_tcsLib_VALUE_OUT_OF_RANGE
The specified value is not in the range 0 to 999

CAVEATS
N/A

29. Lowsstop()

NAME
Lowsstop()

SYNOPSIS
int Lowsstop(void);

DESCRIPTION
This function stops the lower windscreen motion. The command is sent to the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS
OK

ERRNO
N/A

CAVEATS
N/A

30. Lowsstat()

NAME

Lowsstat() – reads the upper windscreen status

SYNOPSIS

lower_wscr_status Lowsstat(void);

DESCRIPTION
This function reads and returns the lower windscreen status. The command is sent to and the answer received from the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS
A **lower_wscr_status**
typedef enum {LWSMOVING, LWSSTOP}**lower_wscr_status;**

ERRNO
S_tcsLib_SLIT_WSCR_u_CONTROLLER_NOT_RESPONDING
Dome micro controller malfunction or power fail
S_tcsLib_UNKNOWN_STATUS_RECEIVED
Self explanatory

CAVEATS
N/A

31. Slwsstat()

NAME

Slwsstat() – reads the slit and windscreen status

SYNOPSIS

slit_wscr_status Slwsstat(void);

DESCRIPTION
This function reads and returns the slit and windscreen system status. The command is sent to and the answer received from the slit micro controller via the RS-232 port 3 of the LCU's MV167 CPU.

RETURNS
A **slit_wscr_status**
typedef enum {SLWSLOCAL, SLWSREMOTE} **slit_wscr_status;**

ERRNO
S_tcsLib_SLIT_WSCR_u_CONTROLLER_NOT_RESPONDING
Dome micro controller malfunction or power fail
S_tcsLib_UNKNOWN_STATUS_RECEIVED
Self explanatory

CAVEATS
N/A

32. Read_environment()

NAME
Read_environment() – reads the 16 environmental sensors

SYNOPSIS
STATUS Read_environment(temp)
temp_humi *temp;

DESCRIPTION
This function delivers the readout of 16 analogue channels associated to the temperature and humidity sensors distributed all over the telescope and building. A pointer to a structure of the type **temp_humi** must be passed to the function.

typedef struct{
char T_sensor1[10];
char T_sensor2[10];
char T_sensor3[10];
char T_sensor4[10];
char T_sensor5[10];
char T_sensor6[10];
char T_sensor7[10];
char T_sensor8[10];
char T_sensor9[10];
char T_sensor10[10];
char T_sensor11[10];
char T_sensor12[10];
char T_sensor13[10];
char T_sensor14[10];
char T_sensor15[10];
char T_sensor16[10];
} **temp_humi;**

RETURNS
OK

ERRNO
N/A

CAVEATS

The physical position of the sensors is as follows:

1- Air temperature outside south of the dome in shadow.
2- Air temperature outside north of the dome, road heat sensor.
3- Air temperature at low level in dome near floor.
4- Surface temperature of main mirror.
5- Air temperature close to the main mirror surface.
6- Surface temperature mid height of the lower Serrurier.
7- Surface temperature mid height of the upper Serrurier.
8- Surface temperature of the secondary mirror.
9- Air temperature close to the top ring.
10- Plate temperature sensor looking into the sky at the top ring.
11- Air temperature inside the dome at mid height.
12- Air temperature for the instrument.
13- Surface temperature of the delta hydrostatic pad.
14- Air temperature inside the adapter.
15- Outside relative humidity close to temperature sensor number 1.
16- Inside relative humidity close to temperature sensor number 11.

Worth to mention that by giving the command "**denv**" at the vxWorks shell a listing of the 16 sensor values is displayed on screen.

33. N2_mode_manual()

NAME
N2_mode_manual() – sets the N2 gas flushing to manual mode

SYNOPSIS
STATUS N2_mode_manual(void);

DESCRIPTION
This function forces the N2 gas flushing for the mosaic window to manual allowing the opening and closing of the valve by using **N2_open()** or **N2_close()**. At boot time a task is created to check the relative humidity at the dome, for humidity values H >=60% the valve is opened and for H<=50% the valve is closed automatically by the task. The default state for the mentioned task at creation time is "automatic", when the task is forced to manual by using **N2_mode_manual()** a warning message is issued to the standard out every 10 seconds.

RETURNS
OK

ERRNO
N/A

CAVEATS
N/A

34. N2_mode_automatic()

NAME
N2_mode_automatic() – sets the N2 gas flushing to automatic mode

SYNOPSIS
STATUS N2_mode_automatic(void);

DESCRIPTION
This function returns the N2 flushing system to default state "automatic".

RETURNS
OK

ERRNO
N/A

CAVEATS
N/A

35. N2_open()

NAME
N2_open() – opens the N2 gas valve.

SYNOPSIS
STATUS N2_open(void);

DESCRIPTION
This function manually opens the N2 gas valve for flushing the mosaic window.

RETURNS
OK or ERROR

ERRNO
The message "Please set N2mode to manual" is sent to the standard out and ERROR is returned if the mode is "automatic" when the **N2_open()** is issued.

CAVEATS
N/A

36. N2_close()

NAME
N2_close() – closes the N2 gas valve.

SYNOPSIS
STATUS N2_close(void);

DESCRIPTION
The same as **N2_open** but for closing the valve

RETURNS
OK or ERROR

ERRNO
The same as N2_open

CAVEATS
N/A

37. N2_status()

NAME
N2_status(void) – displays the state of the N2 flushing system

SYNOPSIS
STATUS N2_status(void);

DESCRIPTION
This function displays the state of the N2 flushing system. Four states are possible:
1- "N2 valve is closed"
2- "N2 supply set to manual"
3- "N2 valve is open"
4- "N2 supply set to automatic"
The pertinent message is sent to the standard out.

RETURNS
OK

ERRNO
N/A

CAVEATS

N/A

38. APPENDIX

38.1. Main Drive Constants

38.1.1.Alpha

Resolution = 0.0036 arcsec
Backlash = 28 X 0.0036 arcsec
Maximum Velocity = +/-3600 arcsec/sec (TCMDIG setting = +/- 21500000)
Acceleration = +/- 400 arcsec/sec*sec (fixed)
Tracking Velocity = 15.04106777 arcsec/sec (TCMDIG setting = 91288)
Maximum Tracking Velocity = +/-75.2 arcsec/sec (TCMDIG setting = +/-456440)

38.1.2.Delta

Resolution = 0.0036 arcsec
Backlash = 65 X 0.0036 arcsec
Maximum Velocity = +/-3600 arcsec/sec (TCMDIG setting = +/- 21500000)
Acceleration = +/- 400 arcsec/sec*sec (fixed)
Maximum Tracking Velocity = +/-75.2 arcsec/sec (TCMDIG setting = +/-456440)

38.2. Focus Drive Constants

Focus Range = 31.73mm
Upper Limit = 29140
Lower Limit = 3758
Resolution = 1.25 microns
Approximate focus position for the old adapter+CCD = 20099