

VST Project

Telescope Control Software - Main Axes System VSTAXIS Module Description

Doc.no. : VST-SPE-ESO-25000-1099

Date: September 23, 2004
Issue: 1.0

*DRAFT VERSION. NOT VALID FOR REVIEW
FOR TECHNICAL DISCUSSION ONLY*

	Name	Date	Signature
<i>Prepared by</i>	P. Schipani, M. Brescia		
<i>Approved by</i>	D.Mancini		
<i>Released by</i>	D.Mancini		

Change Record

Issue/Rev.	Date	Section/Page affected	Reason/Remarks
1.0	23/09/04	All	First Issue

Index

1	Introduction	4
2	Purpose	4
3	Scope	4
4	Reference Documents	4
5	Module changes	5
5.1	axisInternal.h	6
5.2	axisInitData.c	8
5.3	axisServoLoop.c	9
5.4	Makefile.....	10
6	Abbreviation and Acronyms	11
7	Pendings.....	11

1 Introduction

The *vstaxis* software module described in this document is mostly derived from the *axis* ver.2.29.1.1 VLT module. So far VST is using the *vstaxis* module. ESO staff will decide whether to merge the modifications done by the VST staff in a new version of the original *axis* module, so having only one module for VLT and VST, or not.

2 Purpose

This purpose of this document is to describe the differences between the original *axis* module implementation and the *vstaxis* one.

3 Scope

This document covers only the differences between the original *axis* module implementation and the *vstaxis* one. It does not give any detail on the operation of the original *axis* module. It is assumed the reader has a good knowledge of the VLT programming standards and of the original *axis* module.

4 Reference Documents

id.	Document code	Type	Source	Date	Issue
1	VLT-SPE-ESO-17130-1210	VLT Software Axis Control Module acm/axis Design Description	ESO	21.09.2001	2
2	D355711 - 00-A-02	ERO7081C Specifications	Heidenhain	13.02.2001	

5 Module changes

The VST axes are controlled with ST (Software Tacho) option during the tracking phase. The HT (Hardware Tacho) option is also used, but just for the initialization of the encoders (INIT command). The switch between the different tacho operational modes is totally implemented in the *vstlaz* module, the equivalent of VLT *altaz*.

The only change in *vstaxis* with respect to the original *axis* module is in the feedback speed computation.

In *axis* with the ST option the velocity feedback is computed as follows:

$$V_f(t) = \frac{p(t) - p(t - \Delta t)}{\Delta t} \quad (1)$$

where Δt = sampling_time. Once defined:

N_L = number of encoder lines

K_I = encoder electronics interpolation factor

N_H = number of encoder reading heads

In VST case the following values apply (for both altitude and azimuth axes):

$$N_L = 72000$$

$$K_I = 4096$$

$$N_H = 4$$

Therefore the position resolution ΔP is:

$$\Delta P = \frac{360 * 3600}{N_L K_I N_H} = 0.0011 \text{ arcsec} \quad (2)$$

The speed resolution with ST, according to the (1) would be:

$$\Delta v = \frac{\Delta p}{\Delta t}$$

The following table shows the speed resolutions vs two possible choices for the sampling time (so far VST is using $\Delta t = 0.002$):

Speed Resolution with ST Δv [arcsec/s]	Sampling Time Δt [s]
0.55	0.002
0.22	0.005

The resolution could be poor (especially with sampling time=0.002) and this could produce problems at low speeds. This is the reason why some modifications have been done to the original *axis* module. A moving average has been implemented in the speed calculation in order to increase the speed resolution. So far the empirical tests have shown an improvement of the speed loop quality.

5.1 axisInternal.h

The array *posSpeedOld* has been added to the struct *axisServo*:

```
typedef struct
{
  vltINT32          semaphoresLost;          /* global counter for
                                              * lost triggers */
  vltDOUBLE         sumOfPosError;          /* integral of position error */
  vltINT32          toggle;                 /* MAC SYNC PATCH 11.1.97*/
  /* status updated in DB too*/
  vltDOUBLE         remTrkTime;             /* rem tracking time ??
                                              * not supported yet */

  acmAXIS_SERVO_STATE servoState;
  vltBYTES32        servoStateName;
  vltUINT32          encReadCounter;        /* enc reads since last init */
  vltDOUBLE          pos;                   /* current encoder reading */
  vltDOUBLE          posLast;               /* last encoder reading */
  vltDOUBLE          posTime;               /* time of current encoder reading */
  vltDOUBLE          posTimeLast;          /* time of last encoder reading */
  vltLOGICAL         posSpeedValid;        /* valid flag for actual speed */
  vltDOUBLE          posSpeed;              /* speed calculated from last
                                              encoder readings [rad/s] */
  vltDOUBLE          posSpeedOld[50]; /* added for VST: 230604 */
  vltDOUBLE          posSetpoint;           /* in rad */
  vltDOUBLE          posError;              /* position error */
  vltDOUBLE          posErrorRMS2;         /* position error RMS squared */
  vltDOUBLE          posErrorRMS2_coeff;   /* related filter coeff */
  vltDOUBLE          posErrorRMS2_coeff_inv;
  vltUINT32          posErrorRMS2_auxcount;
  vltDOUBLE          velError;

  acmHW_CYCLIC_VELERR velErr;              /* velocity error, updated in
                                              cyclic function if available*/

  vltDOUBLE          speedRefExt;           /* in rad/sec */
  vltLOGICAL         speedRefExtEnable;

  vltDOUBLE          speedSetpoint;         /* in rad/sec */
  vltDOUBLE          speedSetpointUser;    /* rad/sec * convFactor */
  vltDOUBLE          speedSetpointUserWritten; /* like written */

  vltDOUBLE          speedActual;          /* in rad/sec */
  vltDOUBLE          speedActualUser;     /* in rad/sec * convFactor */
  vltDOUBLE          speedActualUserWritten; /* like written */

  vltDOUBLE          velocityFeedForward; /* in rad/sec */

  vltDOUBLE          torquePidOutput;      /* torque output from PID controller */

  vltDOUBLE          torqueRef;            /* torque ref output after filters */
  vltDOUBLE          torqueRefUser;        /* Volt */
  vltDOUBLE          torqueRefUserWritten; /* Volt after clipping */

  vltDOUBLE          dynamicDampValue;     /* SWVC dynamic damping factor */

  vltDOUBLE          deadBandCtr;          /* rad/sec * convFactor */
  vltDOUBLE          acceleration;         /* in rad/(sec*sec) */
  vltLOGICAL         interlockActive;
  vltLOGICAL         vicinityUp;
  vltLOGICAL         vicinityLow;

  /* functions */
  /* !!! tphan
  axisSERVO_LOOP_ACTION_FUNCTIONS actions;
  axisTRIGGER_GENERATING_FUNCTIONS generateTrigger;
  */
  /* setpoints */
  axisPOS_REF        posRef;               /* valid for state
                                              * ~Pos
                                              * ~PosExtrapol*/
  axisPRESET_REF     presetRef;           /* valid for state
                                              * ~PosPreset and

```

```
axisVEL_REF          velRef;          * ~PosFixed*/
                                     /* valid for state
                                     * ~SpeedNoEnc and
                                     * ~SpeedWithEnc
                                     */
vltDOUBLE            pathCorrection; /* correction for path
                                     optimization */
vltDOUBLE            lastRef;         /* last tracking reference */
vltDOUBLE            lastRawRef;     /* last tracking reference */
/*acmACTION*/
vltINT32             actAcmAction; /* is not necessarily the
                                     * action defined in
                                     * posRef.act.ref.action,
                                     * as it is different at
                                     * startup. */

/*acmSTATE*/
vltINT32             inTargetRadiusCounter;
vltINT32             actAcmState;
vltLOGICAL           tracking; /* derived from actAcmState*/
/* trigger */
axisTRIGGER          trigger;
/* timer */
axisTIMER            timerState;
axisTIMER            timerBackoff;

vltLOGICAL           openloop;

}axisSERVO;
```

5.2 axisInitData.c

The initialization of the *posSpeedOld* array has been added to the *axisInitDataServoData* function.

```
void axisInitDataServoData(axisSERVO *data)
{
    int i; /* added for VST */

    data->posSpeedValid = ccsFALSE;

    /*
     * dynamic data: the dynamic data is initialized together with the DB
     * in axisInitStatusDb()
     */
    data->semaphoresLost = 0;
    data->sumOfPosError = 0.0;
    data->toggle = 0;

    /* setpoints */
    data->posRef.last.refValid = ccsFALSE;
    data->posRef.act.refValid = ccsFALSE;
    data->posRef.next.refValid = ccsFALSE;
    /*
     * is initialized together with the DB
     * in axisInitStatusDb()
     */
    data->posRef.actSlope = 0.0;
    data->posRef.actStep = 0.0;
    /*
     */
    data->posRef.lastSetpoint = 0.0;
    data->posRef.agc = 0.0;
    data->posRef.agcActive = ccsFALSE;

    data->presetRef.pos = 0.0;
    data->presetRef.timeout = 0.0;
    data->presetRef.stopFunction = NULL;

    data->velRef.speed = 0.0;
    data->velRef.timeout = 0.0;
    data->velRef.stopFunction = NULL;
    data->velRef.updateVelRefFunction = NULL;

    data->pathCorrection = 0.0;
    data->lastRef = 0.0;
    data->lastRawRef = 0.0;

    data->actAcmAction = acmACTION_NONE;
    data->actAcmState = acmSTATE_UNDEFINED;
    data->tracking = ccsFALSE;

    /* added for VST moving average: 230604 */
    for(i=0; i<50; i++)
        data->posSpeedOld[i] = 0.0;

    /* trigger */
    memset(&axisGlobals->servoData.trigger.flags,
           0x00, sizeof(axisTRIGGER_FLAG));

    /* the flags are the valid flag too
     data-trigger.data
     */
    return;
}
```

5.3 axisServoLoop.c

The moving average calculation has been added to the *axisServoLoop* function. The modifications are delimited by the usage of the VST compilation flag.

```

ccsCOMPL_STAT axisServoLoop
(
  ccsERROR *error
)
{
  int trigg;
  ccsCOMPL_STAT returnValue = SUCCESS;
  double timeDiffMin = axisGlobals->setup.servoLoopDelay / 1000000.0 * 0.5;
  double timeDiffMax = axisGlobals->setup.servoLoopDelay / 1000000.0 * 1.5;
  timsMODE timeMode; /* time system operational mode */
  int i; /* added for VST */
  int MAX = 20; /* added for VST */

.....

  /* position difference [rad] */
  posDiff =
    axisGlobals->servoData.pos -
    axisGlobals->servoData.posLast;

#ifdef VST
/*****
  added for VST moving average 230604 */

  for(i=MAX-1; i>0; i--)
    axisGlobals->servoData.posSpeedOld[i] = axisGlobals->servoData.posSpeedOld[i-1];
/*****/
#endif

  /* calculate actual speed [rad/s] */
  if (axisServoLoop_UnitTimeDiff)
    axisGlobals->servoData.posSpeed =
      posDiff / (axisGlobals->setup.servoLoopDelay / 1000000.0);
  else
    axisGlobals->servoData.posSpeed = posDiff / timeDiff1;

#ifdef VST
/*****
  added for VST moving average 230604 */

  axisGlobals->servoData.posSpeedOld[0] = axisGlobals->servoData.posSpeed;
  axisGlobals->servoData.posSpeed = 0;
  for(i=0; i <MAX; i++)
    axisGlobals->servoData.posSpeed += axisGlobals->servoData.posSpeedOld[i];
  axisGlobals->servoData.posSpeed /= MAX;
/*****/
#endif

.....

```

5.4 Makefile

The *VST* compilation flag has been added. The underlying logic is that at compilation time it can be decided to install or not the VST modifications, with the options shown in the following table:

make options	Installed code
make clean all install	Orginal axis module
make clean all install VST=1	VST modified version

```

#####
# E.S.O. - VLT project
#
# "@(#) $Id: Makefile,v 1.0 2004/09/07 11:03:57 vltscm Exp $"
#
# Makefile of axis module (VxWorks application)
#
# who    when    what
#-----
# bre/schi 01/07/04  compilation flag added for VST
# mchiesa  07/02/98  removed axisPOSLOOP.dbm
# mchiesa  18/03/96  created
#
#####
# This Makefile follows VLT Standards (see Makefile(5) for more).
#####
# REMARKS
# None
#-----

# 010704: in VST case the module must be compiled as follows
#   make clean all install VST=1
ifndef VST
  VST=0
endif

ifeq $(VST),1)
  USER_CFLAGS = -DVST=${VST}
endif

.....

```

6 Abbreviation and Acronyms

ADC	Adapter/Cassegrain
AD/ROT	Adapter/Rotator
AG	Auto-Guider
ALT	Altitude
AO	Active Optics
AZ	Azimuth
CCS	Central Common Software
CDT	Command Definition Table
CI	Command Interpreter
CIT	Command Interface Table
DB	Database
ESO	European Southern Observatory
GUI	Graphical User Interface
IA	Image Analysis
LCC	LCU Common Software
LSF	LCU Server Framework
LCU	Local Control Unit
OAC	Capodimonte Astronomical Observatory
TCS	Telescope Control Software
TIF	Telescope Interface
TWG	Technology Working Group
VLT	Very Large Telescope
VST	Very Large Telescope Survey Telescope
WS	Workstation

7 Pendings

The WHOLE document describes a software not officially released and in modification phase. Therefore many modifications are going to be done on this document in the next months till the completion of the project.



Project: VST
Doc.VST-SPE-OAC-25000-1099
DRAFT VERSION. NOT VALID FOR REVIEW

Rev.: 1.0
Date: September 23, 2004
Pag. 12 of 12

__oOo__