MCM Description - 1

• Domain of application:

All Instrument applications dealing with motors
+ some Telescope Control functions

☞ `NTT Adaptors, EMMI`
☞ `VLT Adaptors, Test Camera, FORS, ISAAC, CONICA, UVES etc...`

• Characteristics:

- common layer between applications and device drivers hiding the HW specificities,
- allows parallel operations on motors,
  - ☞ `FORS is using 19 motors`
- versatile control
  - fully customizable initialization sequence
  - Position (absolute/relative)
  - Speed (to switches)
  - Tracking (in position or velocity)
- supports a variety of hardware configurations.

MCM Description - 2

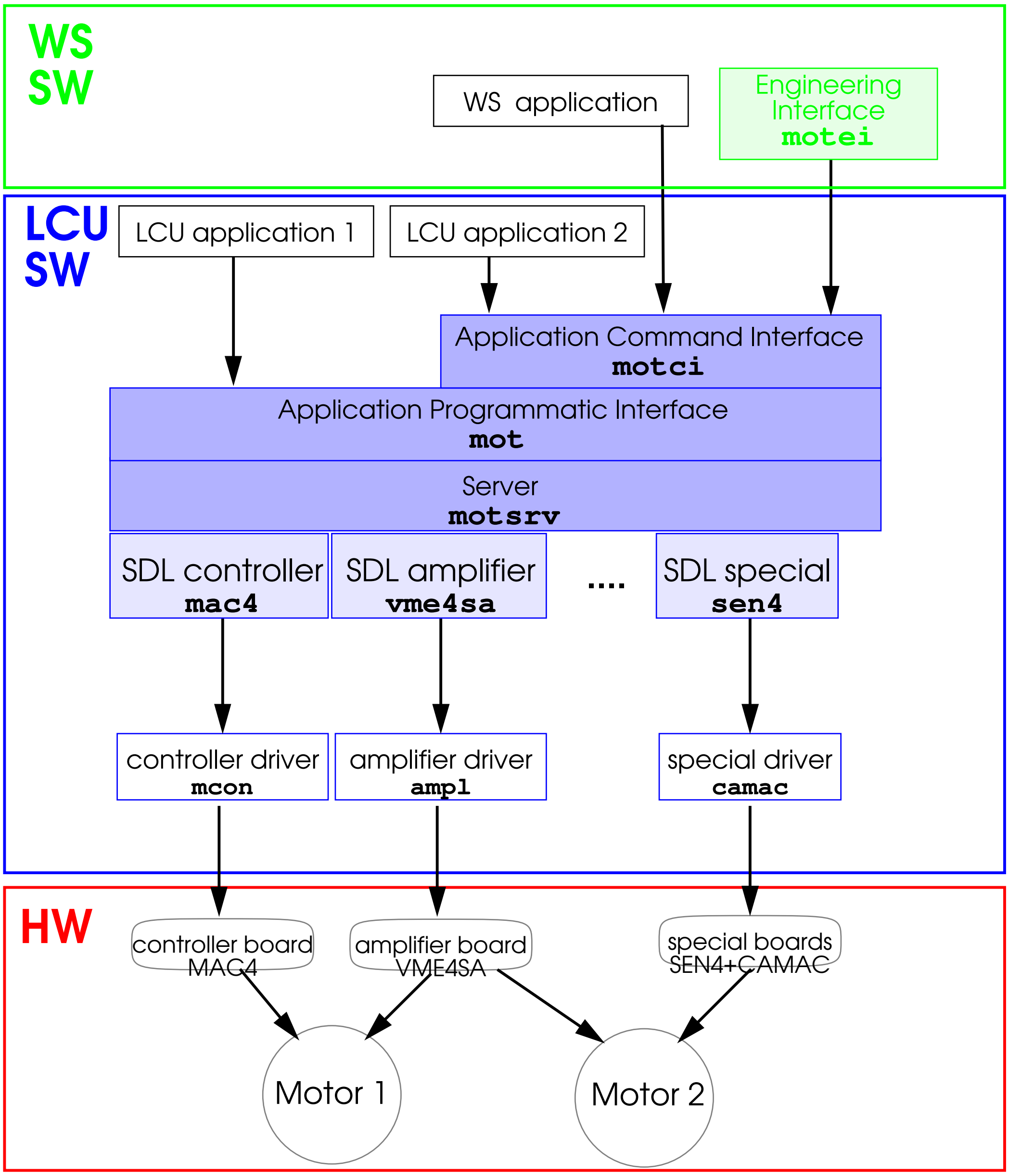
- Configurations:
  - Motor:
    - DC (TC, FORS, UVES, ...)
    - Stepper (ISAAC)
  - Amplifier:
    - Active (VLT, TC, FORS, UVES, ...)
    - Stand-Alone (VLT Coudé)
  - Encoder:
    - Incremental (UVES, NTT Adaptor, EMMI)
    - Absolute (FORS, EMMI)
    - External Incr. (VLT Adaptors)
    - External Abs.
    - No Encoder (ISAAC, EMMI)
  - Linear/Circular Axis
  - 3+2 HW Switches + Encoder Zero Pulse + SW Limits
  - 3 Signals (Interlock, Brake) via Digital I/O Board

- Platforms:

WS : Engineering Interface on CCS
`motei (panels & TCL scripts)`

LCU : Motion Control Core on LCC & Command Interpreter
`mcm`

MCM Architecture



Creating a motor branch in the database

In file $VLTDATA/ENVIRONMENTS/<env>/dbl/DATABASE.db

•static instanciation

```
#define motNUMBER_NAMED_POSITIONS 10
#define motNUMBER_NAMED_SPEEDS 5
#define motNUMBER_INITHW_STEPS 3
#define motNUMBER_CONV_METHODS 5

#define motMOTORROOT :Instr1:Motors:FOCUS
#define motMOTORTYPE motDVAMI
#define motMOTORNAME FOCUS

#include "motor.db"
```

•manual edition

```
POINT BASE_CLASS :Instr1
BEGIN
    ATTRIBUTE BASE_CLASS Motors
    BEGIN
        ATTRIBUTE motDVAMI FOCUS
        BEGIN
            ALIAS "FOCUS"
            ATTRIBUTE int32 SERVER:CONF.axis motAXIS_LINEAR
            etc...
        END
        etc...
    END
    etc...
END
```

Database structure

```
<alias>motor  :SERVER  :CONF .motor
                             .boards
                             .events
                             .signals
                             .axis
                             .encoder
                             .specialPositions
                             .offsets
                             .namedPositions()
                             .namedSpeeds()
                             .twoStepOffset
                             .indexPulseSpeed
                             .defaultSpeed
                       :INIT .actions()
                       :UNIT .defaultUnits
                             .unitConversion()
               :SDL    :<sdl>    :CFG .device
                                      .configValues()
                                 :RUN
               :STATUS
```

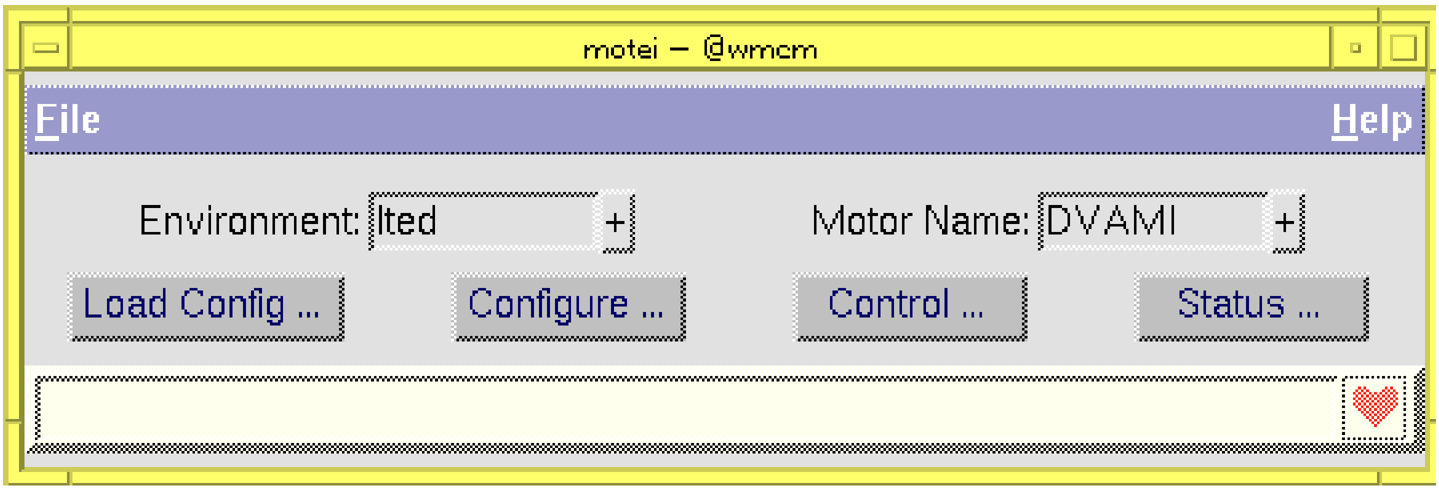
Only these **attributes** must be configured for the specific hardware configuration.

Engineering Interface

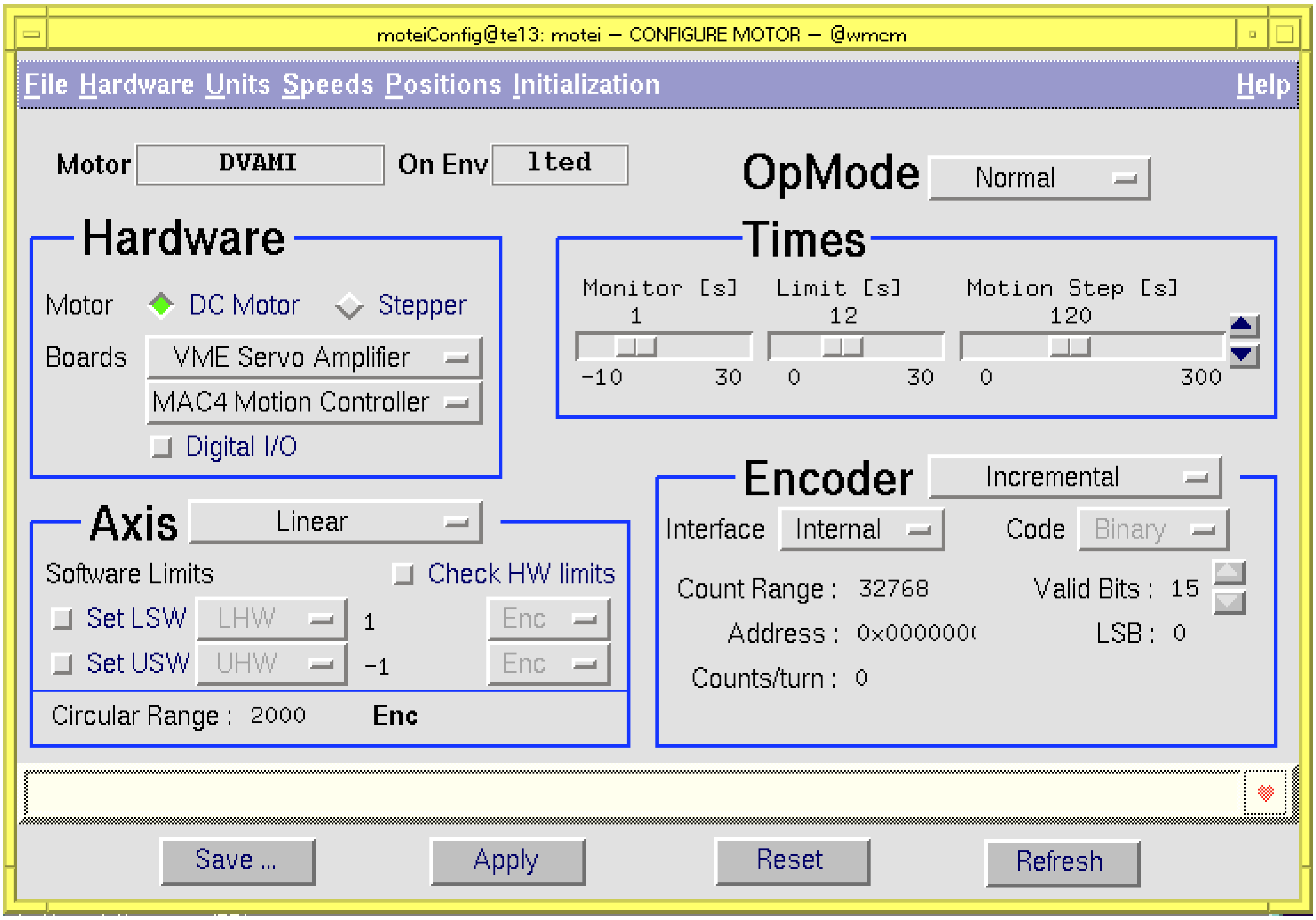
- Full configuration tool for any kind of motor
- Load/Save DB configuration
- Map all ACI commands
- On-Line configuration/motions (LCU)
  Off-Line configuration only (WS)

Starting *motei* from the Unix shell:

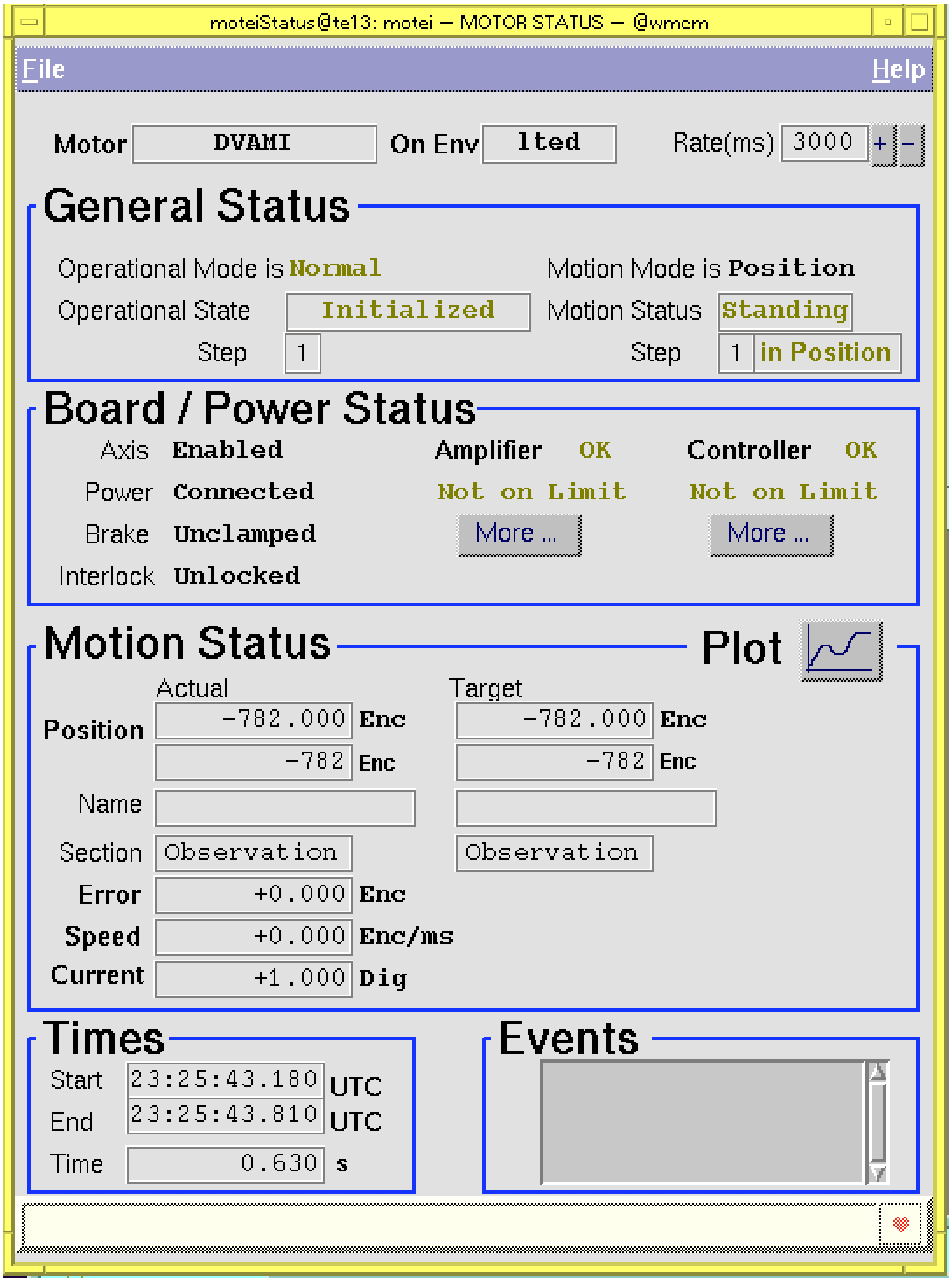
```
prmd@te13:~/MCM/DEMO [60]-> motei [[lted] DVAMI]
```



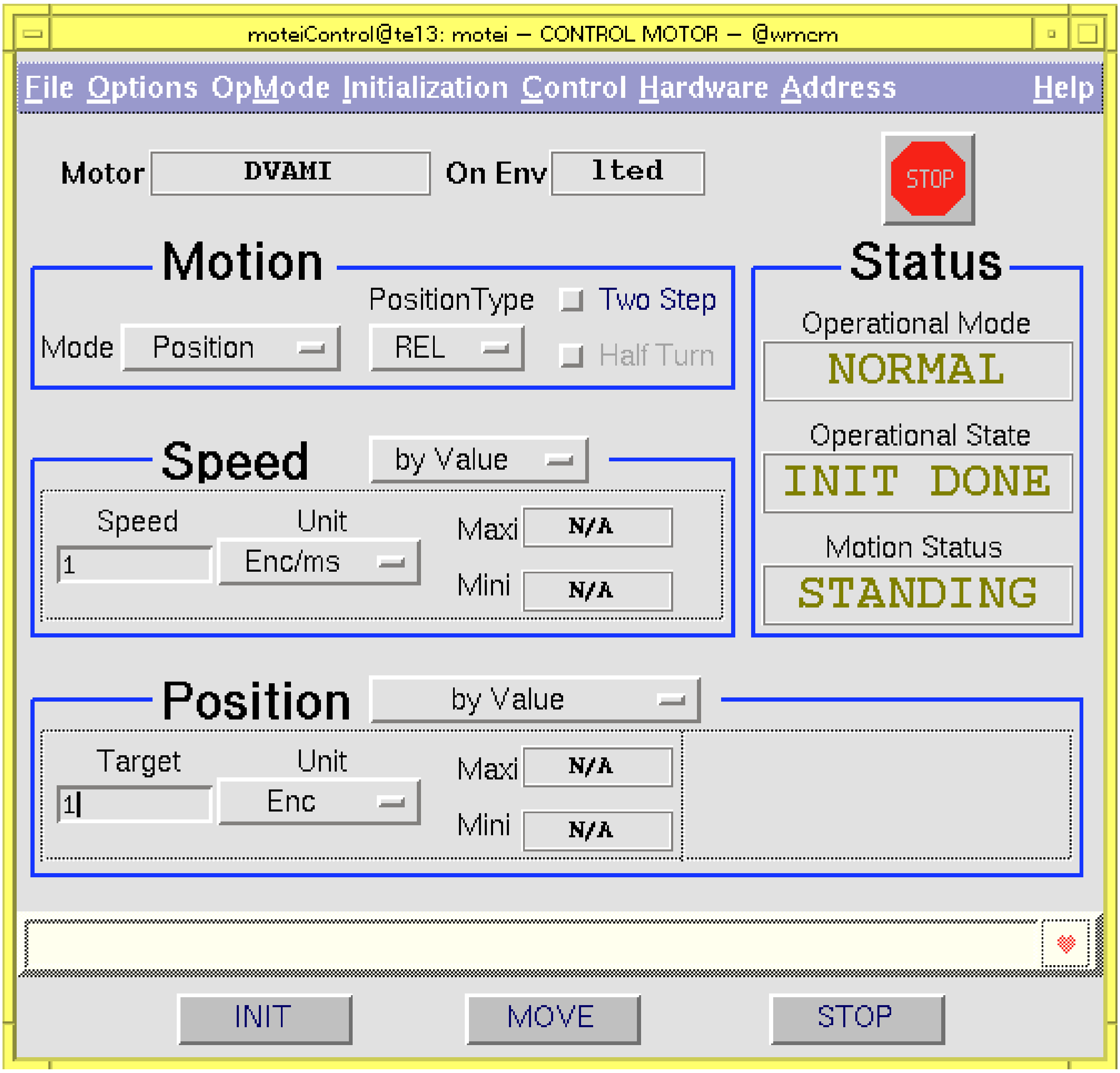
Configuring a motor:



Showing the status of a motor...



Controlling the motor: initialization and motions



Configuring a motor

- Generating a configuration file

  ➜ Using motei on- or off-line
  or
  ➜ Invoking motSaveDb from the WS

- Loading a configuration file

  ➜ Using motei on- or off-line
  or
  ➜ Calling the API function motInitDb
  or
  ➜ Invoking the ACI command INITDB

- Tuning the configuration

  ➜ Using motei on-line
  or
  ➜ Editing the database branch definition file

Using MCM via API/ACI

• Installation, Initialization

• motInstall()	INSTALL
• motAttach()	ATTACH
• motInitDb()	INITDB
• motInitSw()	INITSW
• motInitHw(), motWaitInit()	INITHW, WAITINI

• Motions, active Control

• motSetMotionMode()	MOTMODE
• motMove()	DEFSPD, MOVEABS, MOVEREL, MOVELHW, MOVEUHW, MOVESPD MOVEREF, MOVEIND, MOVEHOM
• motStopMotor()	STOPMOT
• motWaitMove()	WAITMOV

• Status, passive Control

• motStatus()	STATUS, GENSTAT, HWSTAT

• Termination

• motDetach()	DETACH
• motDeinstall()	DEINST

Example:

```
#include "mot.h"

ccsCOMPL_STAT myExample(char *motorName,ccsERROR *error)
     {
     motHANDLE        handle[2] = {0,0}; /* handles for motor access */
     motSTATUS        motorStatus;       /* motor status data structure */
     motMOTIONMODE    motionMode;        /* motion mode */
     motMOTION        motion;            /* motion definition */
     ccsCOMPL_STAT    status=SUCCESS;

     /* install & attach motor in EXCLUSIVE access mode */
     if (motInstall(motorName,error) == FAILURE)
          { /* process the error */ }

     if (motAttach(motorName,motEXCLUSIVE,&handle[0],error) == FAILURE)
          { /* process the error */ }

     /* initialize the motor DB, SW & HW */
     if (motInitDb(motorName,NULL,error) == FAILURE)
          { /* process the error */ }

     if (motInitSw(handle,TRUE,error) == FAILURE)
          { /* process the error */ }

     if (motInitHw(handle,TRUE,error) == FAILURE)
          { /* process the error */ }

     if (motWaitInit(handle,30000,&motStatus,error) == FAILURE)
          { /* process the error */ }
```

Example (cont'd)

```
/* initiate motion */
motionMode = motPOSITION_MODE;
if (motSetMotionMode(handle,&motionMode,error);

motion.posType = motABSOLUTE;
motion.pos.how = motBY_INDEX;
motion.pos.index = 0;
motion.speed.how = motBY_NAME;
strcpy(motion.speed.name,"coarse");
motion.lastStep = TRUE;

if (motMove(handle,&motion,error) == FAILURE)
     { /* process the error */ }

if (motWaitMove(handle,30000,&motStatus,error) == FAILURE)
     { /* process the error */ }

/* get the motor status */
if (motGetStatus(handle,&motStatus,error) == FAILURE)
     { /* process the error */ }

/* detach & deinstall the motor */
if (motDetach(handle,error) == FAILURE)
     { /* process the error */ }

if (motDeinstall(motorName,error) == FAILURE)
     { /* process the error */ }
...
return(status);
}
```

Plotting facility