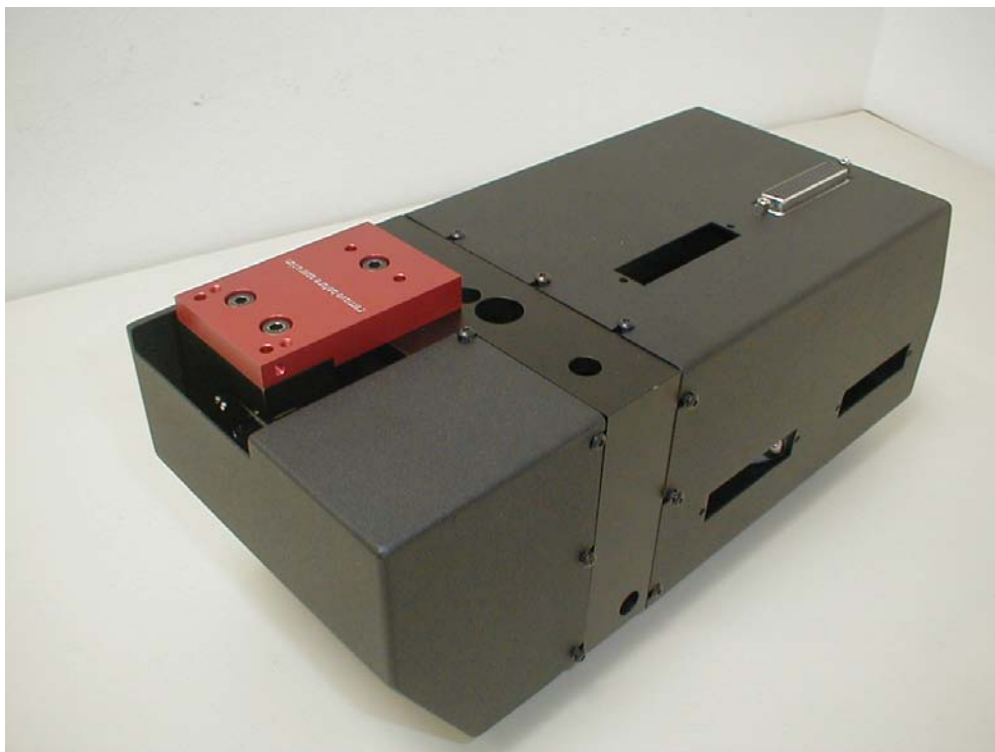


# MA100 - The Six Axis Unit

## Programmers Guide

Version 2.0



Content:

<b>1 INTRODUCTION .....</b>	<b>3</b>
1.1 General information .....	3
1.2 The file SAXU.INI .....	3
1.3 Files created by the program .....	5
<b>2 SIX AXIS UNIT INTERFACE: FUNCTION OVERVIEW .....</b>	<b>6</b>
2.1 Connection board functions .....	6
2.2 Referencing functions .....	6
2.3 Moving functions .....	6
2.4 Other functions .....	6
<b>3 CHANGING BOARD-IDS .....</b>	<b>7</b>
3.1 Systematic of Board-IDs .....	7
3.2 Mechanism to change the board identification number .....	7
<b>4 EXPORTED FUNCTIONS OF SIXAXISUNIT.DLL .....</b>	<b>8</b>
4.1 General Parameters .....	8
4.2 Description of the functions .....	8
<b>5 DESCRIPTION OF THE MAIN-PROGRAM (SIXAXISUNITMAIN.EXE) .13</b>	
5.1 Main Dialog .....	13
5.1.1 Area A – Initialisation .....	13
5.1.2 Point 1 – Kind of Movement .....	14
5.1.3 Point 2 – Restore Last Position .....	14
5.1.4 Point 3 – Working .....	15
5.1.5 Point 4 – Go to Reference Position .....	15
5.1.6 Area B – Translation/Rotation 1-dim, relative .....	15
5.1.7 Area C – Translation/Rotation 3-dim, relative .....	15
5.1.8 Area D – Absolute Movement .....	15
5.1.9 Area E – Position of Stages .....	15
5.1.10 Area F – Velocity .....	16
5.1.11 Area G – Pivot Point .....	16
5.1.12 Area H – Actual .....	16
5.2 Configure Pivot Points Dialog .....	17

# 1 Introduction

## 1.1 General information

After the installation of the hardware and the drivers for the boards and the installation of the Six Axis Unit Software (see: Saxu\_Users\_Manual). You can use either the program (**SixAxisUnitMain.exe**) to control the Six Axis Unit or develop your own main-program.

In the working directory of the Six Axis Unit Software there are the following files, which you need for developing your own program:

C843\_GCS\_DLL.dll  
C843\_Interface.dll  
C843UserStages.dat  
expSixAxisUnit.h  
Lvanlys.dll  
Mathematic\_Matlab\_v01.dll  
MC.dll  
PiStages.dat  
SixAxisUnit.dll  
SixAxisUnitMain.exe  
transdll.dll  
data (directory)  
MatLabRunTime (directory)

If You are programming in C the DLLs can be stored in the same directory like your main-program. If You are programming in LabView copy the DLL-Files to the Windows-Systemdirectory.

The DLL **SixAxisUnit.dll** contains the exported functions, which can be used for programming, the declaration of the functions are in the Header-File **expSixAxisUnit.h**.

In the data directory there are two INI-files:

kalib.ini (do not change this)  
saxu.ini (settings for the program, see section below)

In the MatLabRunTime directory there are the MatLabRunTime DLLs which the program is using. The path to this directory has to be added to your PATH variable.

## 1.2 The file SAXU.INI

The file saxu.ini contains information about the unit and the stages. In general it is not necessary to change it. Only the following parameters can be changed to your own needs:

### Section [Geometry]

max\_move defines the positive and negative limits to which the stages are allowed to move (max value is 5)

max\_vel defines the max. velocity of the stages. This value depends on the used stages. If you use the M-230K040 stages this value is 1.5 mm/sec, if you use the M-230.10 stages the value is 1.0 mm/sec. You can change this value to a smaller value, but not to a bigger one.

### Section [Manipulator]

Type defines the used manipulator (stages). The two manipulators, which can be used with the software are M-230K040 and M-230.10. Do not change the value without changing the manipulator.

motor\_limit Defines the limit, when the servo will switched off, in case of overloading (Standard: 2000)

acc defines the acceleration of the M-230.10 stage (Standard 200)

acc2 defines the acceleration of the M-230K040 stage (Standard 2000)

dec defines the deceleration of the M-230.10 stage (Standard 200)

dec2 defines the deceleration of the M-230K040 stage (Standard 2000)

i\_limit defines the integral-gain for the axis

ErrorMsgIntern if 1 error messages will appear in a popup window, if 0 no error messages will appear, the programmer has to handle the messages by using the function saxGetErrorString after a function returns false.

RefKalinOn do not change this

Correct do not change this

RefModeNr type of referencing the Six Axis Unit, do not change it without calling M-Tech, because each Six Axis Unit is mounted after referenciong with the type defined here.

### Section [Limits]

MaxTrans defines the maximal value for translation movement in mm in world-coordinates. This value is not used in the DLL, but you can use it in your main-program for limiting the allowed moving area.

MaxRotationAngle defines the maximal value for rotation in degrees in world-coordinates. This value is not used in the DLL, but you can use it in your main-program for limiting the allowed moving area.

### Section [UserLimits]

Do not change this section, it will be used in future version.

### Content of saxu.ini:

```
[Geometry]
a=20.0
b=14.0
dx=0.0
dy=0.0
dz=0.0
max_move=4.5
psi_min=60.0
max_vel=1.4
l=10.0
s=34.0
r=20.0

[Limits]
MaxTrans=4.0
MaxRotationAngle=7.0

[Manipulator]
Type=M-230K040
motor_lim=2000
acc=200
dec=200
acc2=2000
dec2=2000
i_limit=200
ErrorMsgIntern=1
RefKalibOn=0
Correct=1
RefModeNr=1

[UserLimits]
Enable=0
P1=-10.0,7.0,0.0
Lim_P1=1.0,1.0,1.0
P2=+10.0,7.0,0.0
Lim_P2=1.0,1.0,1.0
P3=+10.0,-7.0,0.0
Lim_P3=1.0,1.0,1.0
P4=-10.0,-7.0,0.0
Lim_P4=1.0,1.0,1.0a=20.0
```

## 1.3 Files created by the program

### **pivot.ini**

In this file the Pivot-Points for rotation are stored. The user can define for each Six Axis Unit different pivot points.

### **lastpos.dat**

In this file the program stores the last position of the Unit. After restarting the program you can move the Unit to the last position.

## 2 Six Axis Unit Interface: Function Overview

Your main-program can call the following functions, which are exported from the **SixAxisUnit.dll**. The declaration of the functions are in **expSixAxisUnit.h**. You have to include this file in Your Source-Code.

### 2.1 Connection board functions

```
bool saxInitBoards(long lUnit);  
bool saxDisconnectBoards(long lUnit);
```

### 2.2 Referencing functions

```
bool saxReferenceSixAxisUnitExt(long lUnit, bool bSaveMode, bool bFast, bool bDirection);  
bool saxReferenceSixAxisUnit(long lUnit);  
  
bool saxReferenceSixAxisUnitQuickSlowNeg(long lUnit);  
bool saxReferenceSixAxisUnitQuickSlowPos(long lUnit);  
  
bool saxReferenceSixAxisUnitSecSlowPos(long lUnit);  
bool saxReferenceSixAxisUnitSecFastPos(long lUnit);  
  
bool saxReferenceSixAxisUnitSecSlowNeg(long lUnit);  
bool saxReferenceSixAxisUnitSecFastNeg(long lUnit);
```

### 2.3 Moving functions

```
bool saxMoveRel(long lUnit, double *dPos, long nPivot);  
bool saxMoveAbs(long lUnit, double *dPos, long nPivot);  
bool saxGoToReferencePos(long lUnit, long nPivot);  
bool saxRestoreLastPos(long lUnit, double *dPos, long *nPivot);
```

### 2.4 Other functions

```
void saxGetLimits(double *dTrans, double *dRot, double *dMaxVel);  
bool saxSetVelocity(long lUnit, double *dVel);  
bool saxGetStagePos(long lUnit, double *dPos);  
bool saxSetStagesPos(long lUnit, double *dPos);  
bool saxGetTargetPos(long lUnit, double *dPos);  
bool saxSaveActPos(long lUnit, long nPivot);  
  
bool saxGetErrorString(long lUnit, char *ErrStr);  
  
bool saxConfigurePivotPoints(long lUnit);  
bool saxSetPivotPoints(long lUnit, int nType, double a, double b, double *dA, double *dB,  
                        double *dC, double *dD, double *dE);
```

## 3 Changing Board-IDs

### 3.1 Systematic of Board-IDs

All the functions can control up to 4 Six Axes Units. The first parameter is always the number of the unit. You have to configure the board-identification-nr. of the different units as described in the following table:

<i>Unit-Nr.:</i>	<i>2-stages-Board-ID-Nr.</i>	<i>4-stages-Board-ID-Nr.</i>
1	1	2
2	3	4
3	5	6
4	7	8

### 3.2 Mechanism to change the board identification number

If you want to use more than one C-843 cards inside one PC, the cards must be made distinguishable. The mechanism is comparable to set jumpers on an ISA card. But instead of a jumper their is a serial EEPROM on the motor controller card. For reprogramming this EEPROM the following steps have to be done:

1. Place the file EPROM.exe (you find it on the CD in the directory drivers) on a MS-DOS boot disk.
2. Put the motor controller card which should be reprogrammed inside your PC. For the programming only one motor controller card at the same time may be inside your PC.
3. Boot your PC from disk and start EPROM.exe and press key 1 to 4 to program this number inside the EEPROM of the card. Now this card has the entered identification number.

## 4 Exported Functions of SixAxisUnit.dll

### 4.1 General Parameters

long IUnit	Number of the Six Axis Unit you want to move
double dPos[6]	actual position or parameter for the movement (depends on the function you call) of the Six Axis Unit dPos[0] ... X dPos[1] ... Y dPos[2] ... Z dPos[3] ... Angle X dPos[4] ... Angle Y dPos[5] ... Angle Z
long nPivot	determines the Pivot-point for the Rotation. Point A: nPivot = 0 Point B: nPivot = 1 Point C: nPivot = 2 Point D: nPivot = 3 Point E: nPivot = 4

### 4.2 Description of the functions

#### **SIXAXISUNIT\_API bool saxInitBoards(long IUnit)**

Connects the two Boards of the Six Axis Unit *IUnit*. Call this function first in your programm, but do not call this function a second time for the same Unit, without disconnecting (saxDisconnectBoards()) it first.

#### **SIXAXISUNIT\_API bool saxDisconnectBoards(long IUnit)**

Disconnects the two Boards of the Six Axis Unit *IUnit*.

#### **bool saxReferenceSixAxisUnitExt(long IUnit, bool bSaveMode, bool bFast, bool bDirection);**

References the six stages of the two Boards of the Six Axis Unit *IUnit*.

bSavedMode	determines the mode of referencing bSavedMode=true limit mode bSavedMode=false ref mode
bFast	determines the velocity of the movements during referencing bFast=true fast motion bFast=false slow motion, more accurate

bDirection determines the direction of the movement of the stages during reference

bDirection = true	positive direction
bDirection = false	negative direction

Not all combinations of these three parameters are allowed. Moving in ref mode only works with slow motion, but the ref mode is the fastest mode for referencing. Therefore it is the standard mode and should be used. In sum there are 6 possibilities. You can use one of the function-calls below, which represent all possibilities for referencing.

### **SIXAXISUNIT\_API bool saxReferenceSixAxisUnit(long IUnit)**

Referencing is done in the mode defined in the parameter RefModeNr in the SAXU.INI file.

RefModeNr = 1:	Ref mode, slow, negative
RefModeNr = 2:	Ref mode, slow, positive
RefModeNr = 3:	limit mode, slow, positive
RefModeNr = 4:	limit mode, fast, positive
RefModeNr = 5:	limit mode, slow, negative
RefModeNr = 6:	limit mode, fast, negative

### **bool saxReferenceSixAxisUnitQuickSlowNeg(long IUnit);**

Referencing is done in Ref mode, slow, negative (RefModeNr = 1).

Works like a call of: `saxReferenceSixAxisUnitExt(lUnit, false, false, false);`

### **bool saxReferenceSixAxisUnitQuickSlowPos(long IUnit);**

Referencing is done in Ref mode, slow, positive (RefModeNr = 2).

Works like a call of: `saxReferenceSixAxisUnitExt(lUnit, false, false, true);`

### **bool saxReferenceSixAxisUnitSecSlowPos(long IUnit);**

Referencing is done in limit mode, slow, positive (RefModeNr = 3).

Works like a call of: `saxReferenceSixAxisUnitExt(lUnit, true, false, true);`

### **bool saxReferenceSixAxisUnitSecFastPos(long IUnit);**

Referencing is done in limit mode, fast, positive (RefModeNr = 4).

Works like a call of: `saxReferenceSixAxisUnitExt(lUnit, true, true, true);`

### **bool saxReferenceSixAxisUnitSecSlowNeg(long IUnit);**

Referencing is done in limit mode, slow, negative (RefModeNr = 5).

Works like a call of: `saxReferenceSixAxisUnitExt(lUnit, true, false, false);`

### **bool saxReferenceSixAxisUnitSecFastNeg(long IUnit);**

Referencing is done in limit mode, fast, negative (RefModeNr = 6).

Works like a call of: `saxReferenceSixAxisUnitExt(lUnit, true, true, false);`

**SIXAXISUNIT\_API bool saxMoveRel(long IUnit, double \*dPos, long nPivot)**

Relative movement to the coordinates defined in dPos.

dPos[0] ... delta X  
dPos[1] ... delta Y  
dPos[2] ... delta Z  
dPos[3] ... delta angle X  
dPos[4] ... delta angle Y  
dPos[5] ... delta angle Z

nPivot determines the Pivot-point for the Rotation.

**SIXAXISUNIT\_API bool saxMoveAbs(long IUnit, double \*dPos, long nPivot)**

Movement to the absolute coordinates defined in dPos.

dPos[0] ... X  
dPos[1] ... Y  
dPos[2] ... Z  
dPos[3] ... angle X  
dPos[4] ... angle Y  
dPos[5] ... angle Z

nPivot determines the Pivot-point for the Rotation.

**SIXAXISUNIT\_API bool saxGoToReferencePos(long IUnit, long nPivot)**

Move all six stages of the two Boards of the Six Axis Unit *IUnit* to the Reference position.

nPivot determines the Pivot-point for the Rotation.

**SIXAXISUNIT\_API bool saxGetLimits(double \*dTrans, double \*dRot, double \*dMaxVel)**

Get three values out of the saxu.ini file.

dTrans max. value of translation in mm (standard value 4, have to be equal or less than 4)  
dRot max. value of rotation in degrees (standard value 6, have to be equal or less than 7)  
dMaxVel max. value of the connected stages in mm/s

You can use these values for programming your main-program.

**SIXAXISUNIT\_API bool saxSetVelocity(long IUnit, double \*dVel)**

Set the velocities for the six stages of the two Boards of the Six Axis Unit *IUnit*.

dVel has to be declared as a 1-dim-vector with 6 elements (double dVel[6])

The values have to be equal or smaller than dMaxVel (*refer to saxGetLimits*)

**SIXAXISUNIT\_API bool saxGetStagePos(long IUnit, double \*dPos)**

Returns the position of the six stages of the two Boards of the Six Axis Unit *IUnit*.  
dPos has to be declared as a 1-dim-vector with 6 elements (double dPos[6])

**SIXAXISUNIT\_API bool saxSetStagePos(long IUnit, double \*dPos)**

Moves the stages to the position defined in dPos. This is a direct move of the 6 actuators without using the mathematic of the Six Axis Unit.  
dPos has to be declared as a 1-dim-vector with 6 elements (double dPos[6])

**bool saxSaveActPos(long IUnit, long nPivot)**

Saves the actual position of the Six Axis Unit *IUnit*.  
You can use this function after every movement (e.g. call of saxMoveRel(), saxMoveAbs() and saxGoToReferencePos()). If you do this you can move to the last position after restarting the program.

nPivot determines the Pivot-point for the Rotation.

**bool saxRestoreLastPos(long IUnit, double \*dPos, long \*nPivot)**

Restores the last position of the Six Axis Unit *IUnit*, which were stored with the last call of saxSaveActPos(), before the program was closed. This function only read the positions. After you called this function you have to call saxMoveAbs() for moving the Unit to this position.

nPivot determines the Pivot-point for the Rotation.

**bool saxGetErrorString(long IUnit, char \*ErrStr)**

After one of the other functions return false, a call of this function returns the error message.  
ErrStr has to be declared as a vector of char (char ErrStr[250]).

**bool saxGetTargetPos(long IUnit, double \*dPos)**

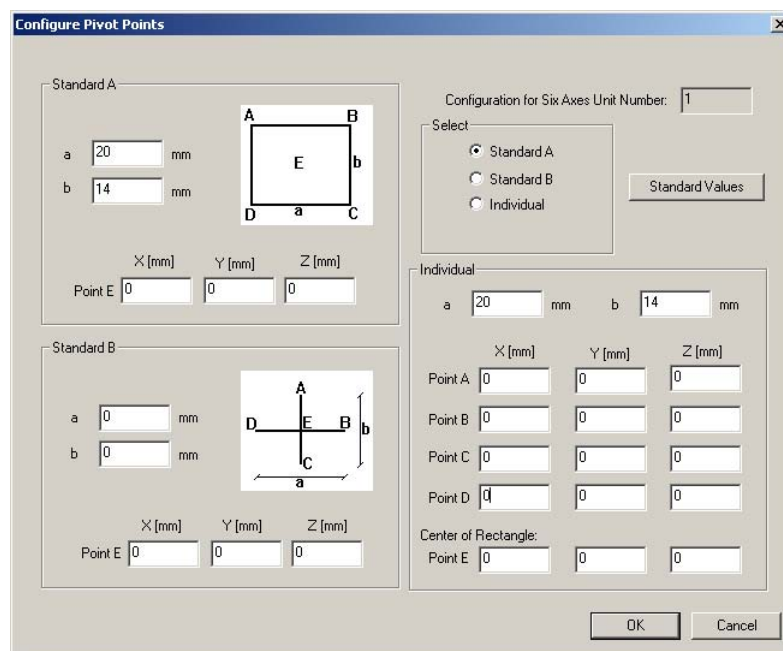
Returns the target position of the 6 stages.  
dPos has to be declared as a 1-dim-vector with 6 elements (double dPos[6])

### **bool saxConfigurePivotPoints(long IUnit)**

This function configures the PivotPoints for the rotation. You can configure for each Six Axis Unit different Pivot Points. Before calling this function you have to go to Reference position first.

In the Configure Dialog you can select 3 types of Pivot Points:

- Standard A** The 5 points are defined as a rectangle with length a and height b and the centre-point E. The program automatically calculates the points A, B, C and D.
- Standard B** The 5 points are defined as a cross with length a and height b and the centre-point E. The program automatically calculates the points A, B, C and D.
- Individual** Here you can define the 5 Pivot-points as needed.



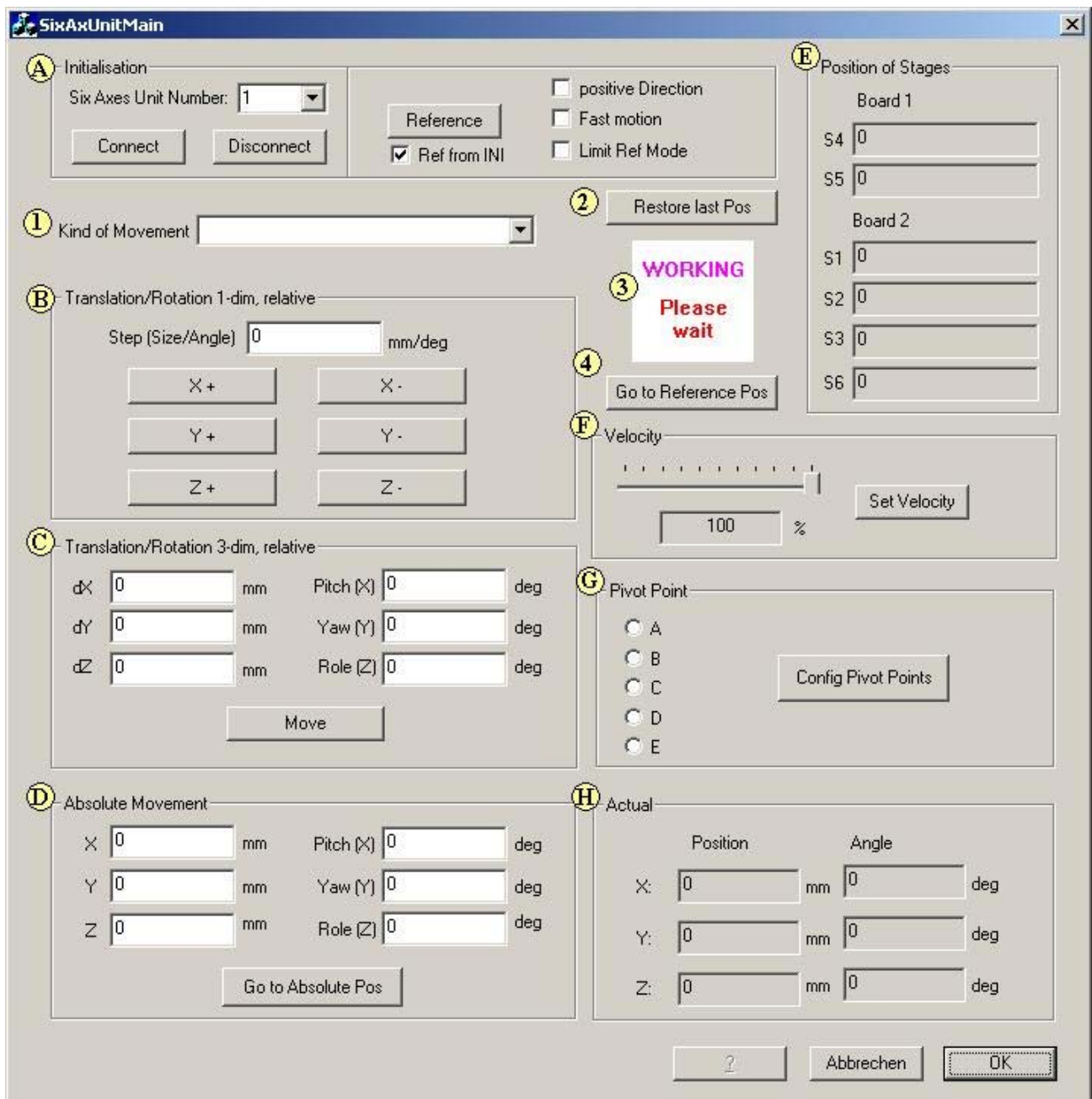
The button [*Standard Values*] restores the standard values for all three types.

### **bool saxSetPivotPoints (long IUnit, int nType, double a, double b, double \*dA, double \*dB, double \*dC, double \*dD, double \*dE)**

This function can be used instead of sax ConfigurePivotPoint. If you know Your points You can write them with this function direct to the PIVOT.INI file.

## 5 Description of the main-program (SixAxisUnitMain.exe)

### 5.1 Main Dialog



#### 5.1.1 Area A – Initialisation

After starting the program you have to select the Six Axis Unit Number, which you want to use. Then with [**Connect**] you will connect the boards of the selected unit. Afterwards you have to [**Reference**] the selected Six Axis Unit. This will last a few seconds and moves the stages to the Zero-Position.

You have four Check-Boxes (**[Positive Direction]**, **[Fast motion]**, **[Limit Ref Mode]** and **[Ref from INI]**) to decide the type of referencing.

**[Positive Direction]** determines the direction of the movement of the stages during reference  
if *checked* referencing will move first to the positive limit switch

**[Fast motion]** determines the velocity of the movements during referencing  
if *checked* referencing movements will be faster

**[Limit Ref Mode]** determines the mode of referencing  
if *checked* referencing will work in limit mode, otherwise in ref mode

**[Ref from INI]** determines, that the reference mode is read from the SAXU.INI file  
if *checked* the three other check boxes are ignored

Not all combinations of these three parameters are allowed. Moving in ref mode only works with slow motion. So in sum you have 6 possibilities.

The button **[Disconnect]** will disconnect the boards of the selected units. Afterwards you have to **[Connect]** and **[Reference]** them again, if you want to move the unit.

### 5.1.2 Point 1 – Kind of Movement

Select the kind of movement you want to make. Depending to your selection different areas of the dialog will be enabled and disabled.

There are 4 possibilities:

- Rotation 1-dim, relative one dimensional rotation either across X, Y or Z-axes, relative to the actual position
- Translation 1-dim, relative one dimensional translation either in X, Y or Z-direction, relative to the actual position
- Translation/Rotation 3-dim, relative three-dimensional movement – translation and rotation, relative to the actual position
- Absolute Movement movement to the absolute position, defined in X,Y and Z and rotation-angle across X, Y and Z.

At the start of the program the “*Translation 1-dim, relative*” is selected.

### 5.1.3 Point 2 – Restore Last Position

This button will moves the stages to the position, where they have been at the previous ending of the program.

#### 5.1.4 Point 3 – Working

During the process of referencing or moving the Unit, this info will be displayed.

#### 5.1.5 Point 4 – Go to Reference Position

This button moves the Six Axis Unit to the Reference position.

#### 5.1.6 Area B – Translation/Rotation 1-dim, relative

If you select “*Rotation 1-dim, relative*” or “*Translation 1-dim, relative*” this area will be enabled.

If you selected “*Translation 1-dim, relative*” the value Step will be the translation step in mm. With the buttons [X+],[X-],[Y+],[Y-],[Z+],[Z-] you will start a translation in the positive (+) or negative (-) X,Y,Z direction.

If you selected “*Rotation 1-dim, relative*” the value Step will be the rotational angle in degrees. With the buttons [X+],[X-],[Y+],[Y-],[Z+],[Z-] you will start a rotation across the X-, Y- or Z-axis in positive (+) or negative (-) direction.

#### 5.1.7 Area C – Translation/Rotation 3-dim, relative

If you select “*Translation/Rotation 3-dim, relative*” this area will be enabled.

Here you can define the translation-step dX, dY and dZ for a translation in X, Y and Z-direction and the rotational angle in degrees for a rotation across the X-, Y- or Z-axis. After defining the values [Move] will start the relative movement.

#### 5.1.8 Area D – Absolute Movement

If you select “*Absolute Movement*” this area will be enabled.

Here you can define the absolute position X,Y,Z and the rotational angle in degrees for a rotation to the absolute angle across the X-, Y- or Z-axis. After defining the values [Go to Absolute Pos] will start the movement.

#### 5.1.9 Area E – Position of Stages

Display of the actual position of the stages of the Six Axis Unit. Only informational character.

### 5.1.10 Area F – Velocity

Set the standard velocity of all stages (0 to 100%). After selecting the velocity you have to push [**Set Velocity**] to write the velocity to the boards.

### 5.1.11 Area G – Pivot Point

If you select “*Rotation 1-dim, relative*” or “*Translation/Rotation 3-dim, relative*” or “*Absolute Movement*” this area will be enabled.

Here you can select the Pivot-Point for the rotation and with [**Config Pivot Points**] you can define the position of the Pivot Points (*see: Configure Pivot Points*)

### 5.1.12 Area H – Actual

Here are the actual position and the actual angle of the Six Axis Unit displayed.

NOTE: If you make a lot of relative rotational-movements the value of the actual angle will only take sense, if you haven’t changed the Pivot Point during this rotations.

## 5.2 Configure Pivot Points Dialog

In the Configure Dialog you can select 3 types of Pivot Points:

### Standard A

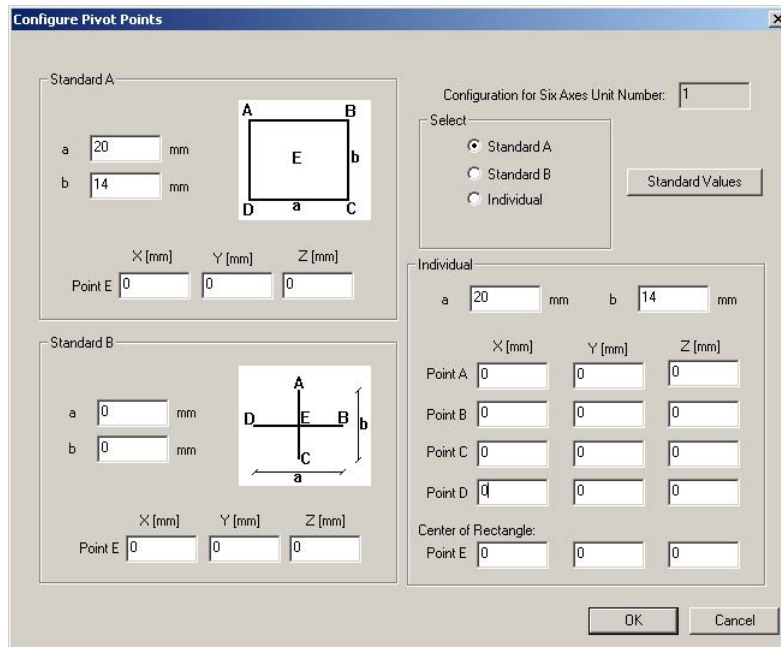
The 5 points are defined as a rectangle with length a and height b and the centre-point E. The program automatically calculates the points A, B, C and D.

### Standard B

The 5 points are defined as a cross with length a and height b and the centre-point E. The program automatically calculates the points A, B, C and D.

### Individual

Here you can define the 5 Pivot-points as needed.



The button [*Standard Values*] restores the standard values for all three types.