zub machine control AG
Buzibachstrasse 31
6023 Rothenburg
Schweiz

Telefon +41 (0)41 541 50 40
info@zub.ch
www.zub.ch

Getting Started:

MiniMACS6-AMP4

with AposslDE / AposssC / SDK

Date	10. September 2024
Document number	11645158
Document index	01
Specific use	Document for external use

Content

1 About this document

This document shows how a MiniMACS6-AMP4 motion controller from maxon | zub is used.

We cover the basic workflow on how to connect and run one motor on a MiniMACS6-AMP4 controller with help of the AposslDE and AposssC programs. The use of the AposssC SDK (software development kit) is described and the use of the AposslDE tool “Oscilloscope”, for monitoring process values during program execution and motor motion, is explained.

Nowadays, the most common motor configuration is an EC motor with hall sensors and incremental encoder. In this tutorial we concentrate on how to use such a motor configuration.

For keeping this document short and easy to understand the description in here doesn’t cover all the functionality and different types of MACS controllers. In this document the use of the MiniMACS6-AMP4 with internal amplifiers is described. Some hints and links are given for more advanced operation.

1.1 Symbols & Signs

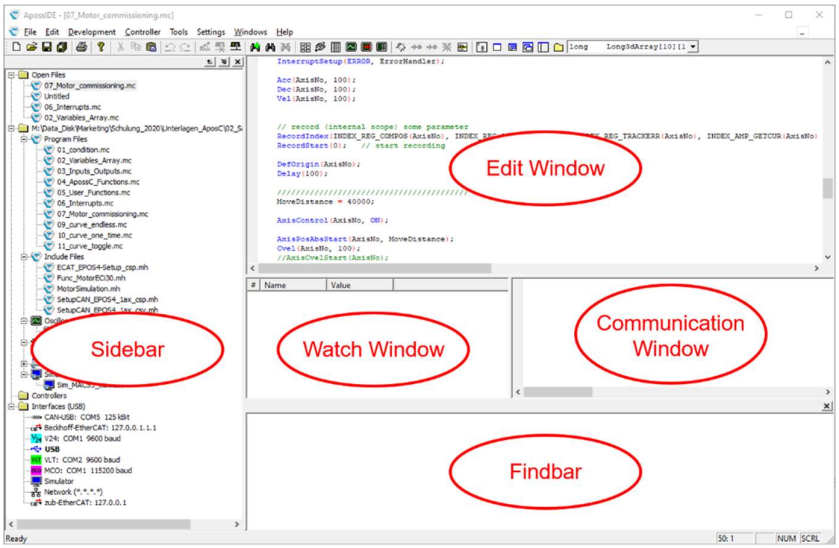
Type	Symbol	Meaning	
Safety alert	 (typical)	DANGER	Indicates an **imminent hazardous situation**. If not avoided, it **will result in death or serious injury**.
		WARNING	Indicates a **potential hazardous situation**. If not avoided, it **can result in death or serious injury**.
		CAUTION	Indicates a **probable hazardous situation** or calls the attention to unsafe practices. If not avoided, it **may result in injury**.
Information		Requirement / Note / Remark	Indicates an activity you must perform prior continuing, or gives information on a particular item you need to observe.
Information		Material Damage	Indicates information particular to possible damage of the equipment.

2 Software and Hardware

Before we can start, we need some software and hardware.

2.1 AposslDE

AposslDE is the integrated development environment for MACS controllers. The AposslDE is used to program the MACS controllers in AposssC. AposssC is a programming language based on standard C syntax with some limitation but also some extra features. Please check out the AposslDE Help page “Difference of AposssC and C”.
The AposslDE provides a Code Editor with syntax highlighting and different tools.



picture 1: Overview AposslDE

The AposslDE (**AposslDE.zip**) can be downloaded from the maxon website following this link: https://www.maxongroup.com/maxon/view/product/control/Motion-Control/MiniMacs/001755?download=show

Extract the zip-file and Install this AposslDE on a Windows PC with a USB Interface available.

2.2 AposssC_SDK

AposssC_SDK (software development kit) is a bundle of functions that can be used in AposssC application programs. It also includes example on how to use the AposssC / SDK features and functions.

The AposSC_SDK (AposSC_SDK.zip) can be downloaded from the maxon website following this link:
https://www.maxongroup.com/maxon/view/product/control/Motion-Control/MiniMacs/001755?download=show

Unzip and copy it into your working folder and make sure that you can open a sample program and run the syntax check successfully.

2.3 Hardware

To work with this document, you need a MiniMACS6-AMP4 motion controller.

Check for the Hardware Reference Manual of the MiniMACS6-AMP4 **"MiniMACS6-AMP-4/50/10 Hardware Reference"**. It can be found in the download section of the maxon catalog website, following this link:
https://www.maxongroup.com/maxon/view/product/control/Motion-Control/MiniMacs/001755?download=show

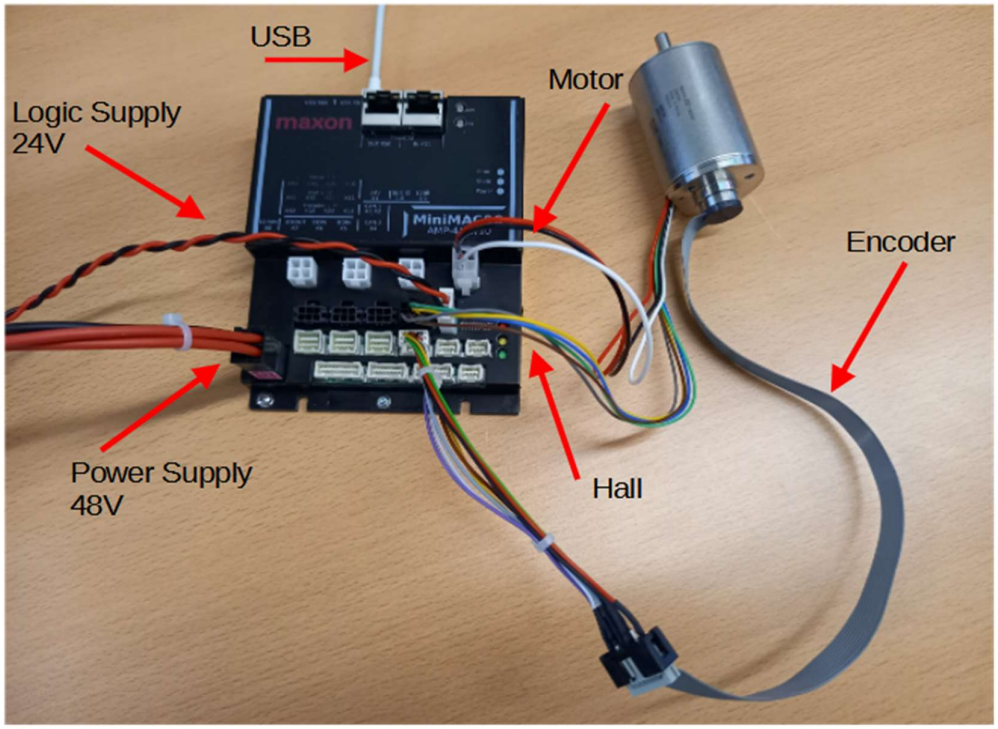
Read and understand the Hardware Reference manual.

Make sure you can connect the MiniMACS6-AMP4 to the AposIDE using the USB interface. Then you must power the MiniMACS6-AMP4 with the logic supply 24V on the connector X1.

	**Hot plugging the USB interface may cause hardware damage** If the USB interface is being hot-plugged (connecting while the power supply is on), the possibly high potential differences of the two power supplies of controller and PC/Notebook can lead to damaged hardware. • Avoid potential differences between the power supply of controller and PC/Notebook or, if possible, balance them. • Insert the USB connector first, then switch on the power supply of the controller.

Furthermore, connect the power supply (12V-50V) on X8.

You need an EC motor with incremental encoder and hall sensors.
Connect a motor (X10) with its incremental encoder (X12) and hall signals (X11).
Please refer to the hardware reference manual **"MiniMACS6-AMP-4/50/10 Hardware Reference"** for details of the wiring.



picture 2: MiniMACS6-AMP4 with correct wiring

	It is highly recommended to use a motor that can spin freely for the first commissioning.

	The MiniMACS6-AMP4 can also operate with DC motors, stepper motors, or other EC motor commutating types (BLDC and PMSM).

3 Information about motor and encoder

Some information about the motor and the encoder are needed, before we can setup the correct configuration for the MiniMACS6-AMP4.

3.1 Motor information

The following motor information are needed:

- Number of pole pairs
- Maximum Speed
- Nominal current
- Maximum current
- Winding resistor
- Winding inductance

All this information can be found in the datasheet of the used motor.

3.2 Encoder information

The information we need of the incremental encoder is, how many pules the encoder generates during one rotation. It is found in the datasheet under “encoder resolution”. For example 500 inc/turn. Because we have channel A and B of the incremental encoder, we can quadruple the position information. Such a pulse we call quadcount (qc). For a 500 Inc/turn encoder we have 2000qc/turn. The MiniMACS6-MACS4 works internally with quadcounts.

	In this document we describe the use of an incremental encoder only. SSI-encoder, SinCos-encoder, or the use of hall-signals only as position feedback are also possibilities.

4 AposSC application program

The MiniMACS6-AMP4 is a fully programmable motion controller. That means that a compiled application program can be downloaded and executed on the controller. The programming language is **AposSC** which is an extended C syntax. All parameter settings are normally done in the AposSC code. The parameters are written to the MiniMACS6-AMP4 firmware during the program execution.

	All parameters and process data are SDO objects listed in the SDO dictionary. The objects are organized with index and subindex (0x2300/02 = VELMAX) They can be set/read over any interface (USB, CAN, EtherCAT, Ethernet) from a PC, PLC, or other MACS controllers. Access to all objects can also be done internally (from AposSC code). The #include <SysDef.mh> with a bundle of defines and macros allows the use of nice names and easy to read access to the objects in the AposSC code.

A program can be simple executed on the controller from RAM.

- Use **F4** or “Development -> Syntax Check” to check if the program compiles properly.
- Use **F5** or “Development -> Execute” run the program out of RAM (the program is not flashed)

To flash the program, make sure your program window is in the foreground and “Controller -> Programs” and save the program with a name of max 8 characters. By setting the “Autostart on” flag, the program starts automatically when the controller is powered up.

4.1 Hello World program

It is recommended to start with simple application programs without using axis control features to get used to programming the controller. Open the AposIDE click on “File” -> “New” and choose “.mc Program” (AposC). Add the “Hello World” print statement as shown below. Press F5 to run this code. You should see the message “Hello World” in the communication window.

Hello_World.mc

```
1   #include <SysDef.mh>
2
3   long main(void)
4   {
5       print("Hello World");
6       return(0);
7   }
8
```

picture 1: Hello World AposC program

4.2 Motor commissioning program

In this step an application program is made for setting up motor/encoder configuration, using a copy of the SDK example “Maxon_ECi30_1ax_SC_Hall_Inc.mc”.

The used example is a small program showing the motor, encoder and control settings for one axis. The axis then is moved in a back-and-forth motion using AposSC positioning commands. It also shows a simple error handler and how to use an internal data recorder.

For using a motor with the internal amplifier of the MiniMACS6-AMP4 we must set some parameters correctly, including amplifier parameters (HWAMP), encoder parameters (HWENC and VIRTCOUNTIN) and axis parameter (AXE_PARAM). To do so we just use SDK function calls.

For additional information about the SDK and the use of it refer to the document “AposSIDE_AposSCSDK_HowToUse.pdf” in the SDK root folder.

Navigate to the example “Maxon_ECi30_1ax_SC_Hall_Inc.mc” in the AposSC_SDK folder:

AposSC_SDK\Example\Amplifier\MACS_Internal\BLDC_Motor\Sinusoidal_Commutation\Hall_Alignment

Best practice is to start with a copy of this .mc file and rename it. In this case the path to the SDK library is already set correctly and the original file still exists for reference.

	All the SDK functions are visible and open for users. This allows see the source code and necessary changes can be made if needed. It is recommended to change the function name when changes are done.

The example program uses #define for the motor and encoder settings. Adjust those #defines according to your needs. The #defines are use in this example program as parameters for the SDK functions.

Inside the function “setupEC_i30_SC_Hall_Inc(…)” the following SDK functions are called:

sdkSetupIncEncoder(…)
sdkSetupAmpHallPmsmMotor(…)
sdkSetupCurrentPIControl(…)
sdkSetupAxisMovementParam(…)
sdkSetupAxisDirection(…)

sdkSetupPositionPIDControlExt(…)
sdkSetupAxisUserUnits(…)
sdkSetupVirtualI2T(…)

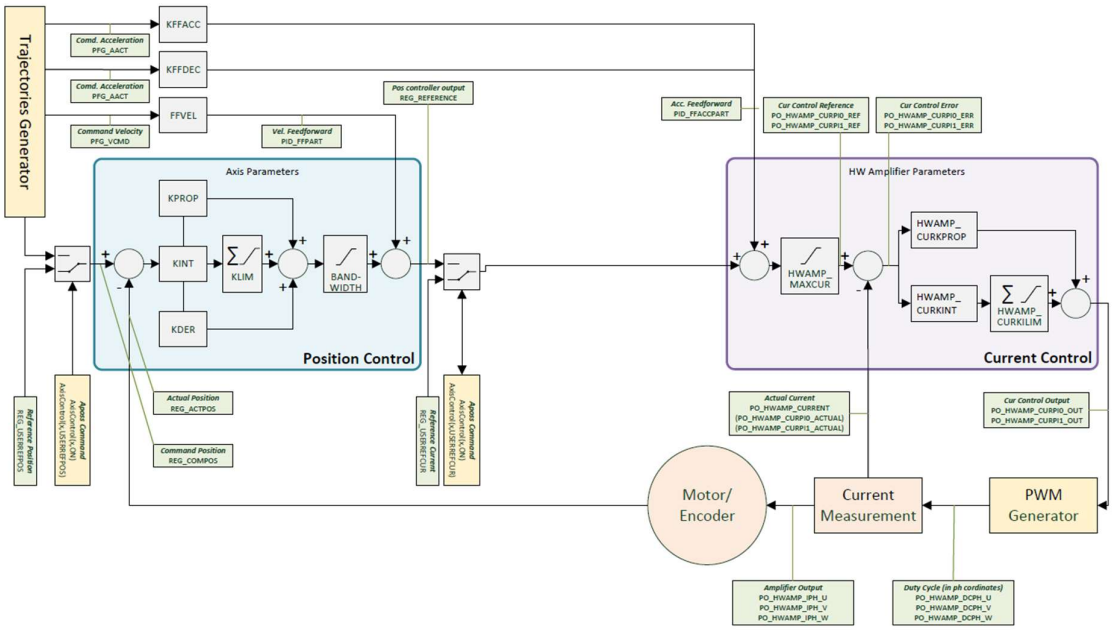
A full list of all SDK functions and the description can be found in the document “AposssC_SDK_Help_V0x.xx.pdf” or “AposssC_SDK_Help_V0x.xx.html” in the AposssC_SDK/Help/ folder.

Check the next chapters for information on finding settings for the current controller, position controller and settings for user units and control modes.

5 Special axis settings

5.1 Mode of operation

In this example we run the motor in the most used way – in the position-controlled mode. The setting of this mode is done with HWAMP_MODE_POS_CUR in the #define EC_i30_CONTROLMODE.
The commanded position is generated by the internal trajectory generator. That allows us to move the axis to a target position with a given acceleration and velocity. For the actual position the encoder feedback is used.
The example also activates the current control loop underneath. So, we must do some settings for the current controller first.

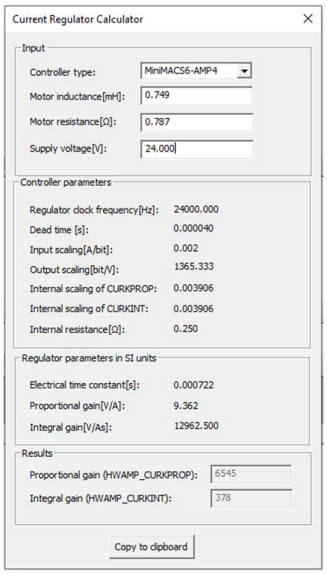


picture 4: Controller Cascade Position - Current

	Other controller principles are also possible when working with the MiniMACS6-AMP4. For example, a velocity controller is available. Check the AposslDE help file for the setting of HWAMP_COMMTYPE.

5.2 Current Controller

The current controller is a PI controller. There is a proportional-gain value and integral-gain value that must be set by the user. In the program we can use the SDK function "**sdkSetupCurrentPIControl**". The PI values of the current controller can be calculated with the motor's winding resistance and the winding inductance using the tool "Current Regulator Calculator". In addition, we also must know the power supply voltage.
Find the "Current Regulator Calculator" tool in the AposslDE under "Tools".



picture 5: Current Regulator Calculator

The results from the tool must be copied to the defines
#define EC_i30_CURKPROP
#define EC_i30_CURKINT

Those are used in the SDK function **sdkSetupCurrentPIControl()**.

The integral limit can be set to default value (32767).
#define EC_i30_CURKILIM

5.3 Encoder and User Unit settings

Encoder Settings

In this example we use the incremental encoder as position counter. The unit of this counter is quadcounts [qc]. Note, that the datasheet of the encoder shows counts per turn of one channel. So quadcounts are 4 times higher than the encoder counts.

The incremental encoder is the default, so no special settings are required. Also, the encoder on terminal X12 is linked to the axis 0 by default.
We use the function **sdkSetupIncEncoder,** anyway.

User Unit Settings

In the example program settings are done, that the user units of the axis are hundreds of degrees. So, one rotation of the motor shaft (no gear is used) is 36000 user units [UU].
The SDK function for the user unit settings is **sdkSetupAxisUserUnits** .

	By changing the values POSFACT_Z, POSFACT_N it is possible to set a gear factor. See: AposslDE Help: Motion Control topic > Scaling > User Units [UU]

Encoder test

The encoder feedback can be tested by checking the encoder counter value. The HW counter for the incremental encoder is available in the diagnostics window of the AposslDE. “Controller” -> “Diagnostics”.
Turn the motor shaft by hand by one rotation and check the value change in the lower right corner of the Diagnostics window. This value is displayed in qc.

5.4 Position controller

The position controller works with the calculated deviation from the command position (commanded from the profile generator) and the actual position (encoder feedback). This value is the actual tracking error and is the source for the position PID controller. The output of the position PID controller is the reference value and is fed into the current controller as a current setpoint value.
That the position controller works correctly, it is essential that a positive current setpoint results in a positive encoder position value change.

The MiniMACS6-AMP4 offers no auto tuning algorithm. That means, the position controller must be tuned manually.
For tuning the positioning PID controller with its values:

- KPROP
- KINT
- KDER

use an endless back-and-forth motion of the axis (as in the example program) and
the controller must be activated in the application program with the command AxisControl(axis, ON).

	It is highly recommended that the motor can spin freely to perform this task the first time.

5.5 Steps for the position PID tuning

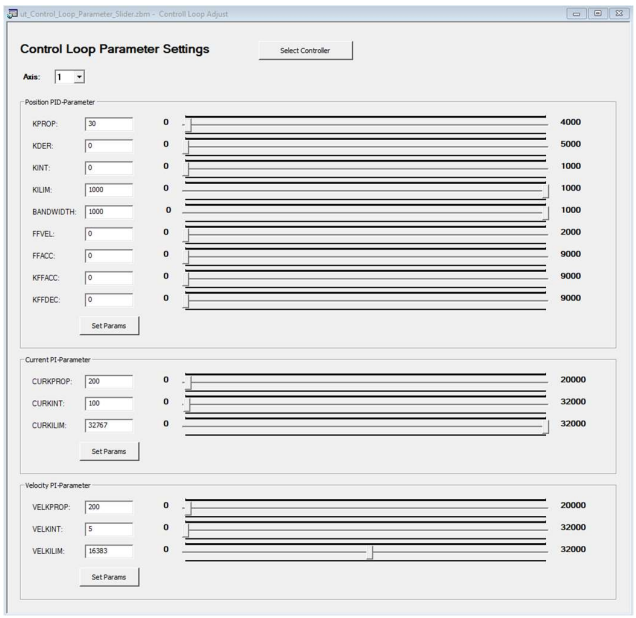
The aim for the tuning is to find values for the P (proportional), I (integral) and D (derivative) part of the controller to keep the actual position as close to the commanded position as possible or in other word keep the track error (following error or deviation) as small as possible. Keep in mind that you never reach zero deviation during the movement. It is also important that the controller is stable and doesn't start to oscillate.

Use the following steps to get a good control:

1) Bring all feed forward values **FFVEL**, **KFFACC** and **KFFDEC** to zero.
2) To avoid any position error during the tuning, increase **POSERR** to a high value.
3) Set integral factor **KINT** and derivative factor **KDER** to zero. Start with a small proportional factor **KPROP** of around 10.
4) Increase the proportional factor **KPROP** in small steps until the motor starts to oscillate. Then reduce **KPROP** again until the motor stops to oscillate.
5) Now increase the derivative factor **KDER** until the motor starts to vibrate. A rule of thumb is that **KDER** is 3 times of the **KPROP** value.
6) After increasing **KDER** it is likely that the **KPROP** can be increased further. So repeat step 4) and 5) until the result is good but there is not oscillation/vibration even the motor shaft is disturbed.
7) Finally, the integral factor **KINT** can be increased. The integral component helps to reduce a persistent deviation in constant velocity or in standstill but tends to swing during motion. It is recommended to use just a small **KINT** value (mostly 4-20).
8) Take the final values and insert them in the program code. Also bring back the **POSERR** to an acceptable value.

Good practice is to disturb the motor during motion and position reached phase. An optimized controller is stable without starting to oscillate. The quality of the positioning controller dispends on the needs of the application. The oscilloscope tool, which is part of the AposslDE, may help to visualize the actual tuning process. See chapter 6.

For manipulating the PID parameter values the monitor file “ut_Control_Loop_Parameter_Slider.zbm” can be used. Find this monitor file in the SDK folder under:
\SDK\Utility\Control_Loop_Parameter_Slider\ut_Control_Loop_Parameter_Slider.zbm



picture 6: ut_Control_Loop_Parameter_Slider.zbm

The position PID settings in the example application are done with the #defines:

- EC_i30_KPROP
- EC_i30_KINT
- EC_i30_KDER

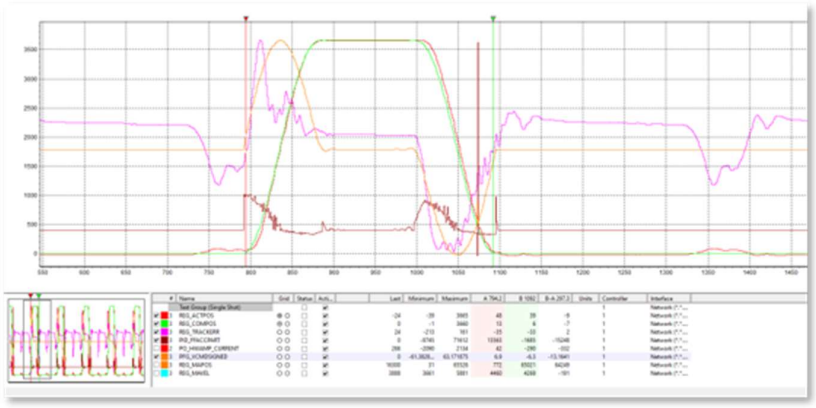
The SDK function **sdkSetupPositionPIDControl** or **sdkSetupPositionPIDControlExt** is used.

The whole mechanical system and the load has an influence on the position PID controller. Therefore, it might be necessary to fine tune the PID controller again in the final application.

	For more advanced setting you can use feed forward settings to get better results in the positioning control.

6 Using the oscilloscope

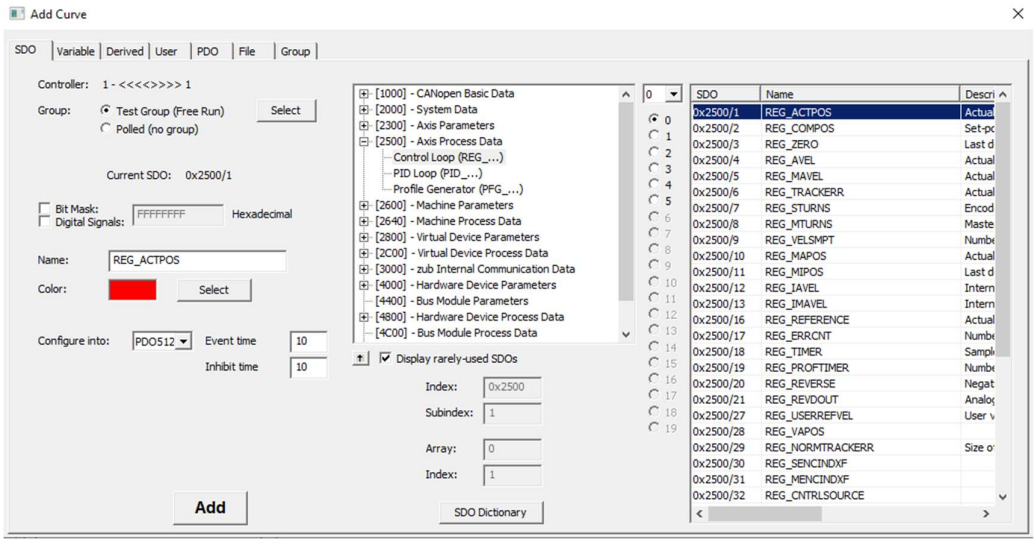
There is an oscilloscope in the AposslDE, which is helpful to display values over time.



picture 7: Oscilloscope

The oscilloscope can be found under “Tools” or just use the icon  in the icon bar.

The “Free Run” oscilloscope is easy to use. Just add values of your choice to the value list. Use the icon “Add curve” to select object for the recording. 



picture 8: Oscilloscope add curve

For your first motor move and the tuning of the position controller the following values are of interest:

- REG_ACTPOS (0x2500 / 1) Actual position in qc
- REG_COMPOS (0x2500 / 2) Commanded position in qc
- REG_TRACKERR (0x2500 / 6) Actual position tracking error (COMPOS minus ACTPOS)
- PO_HWAMP_CURRENT (0x4800 / 2) Actual current in mA

To start the oscilloscope recording press the button “Start”. 

Stop a running recoding with the button “Stop”. 

You can use zoom features and two cursers to navigate and analyze the recorded curves. Note that an oscilloscope can be saved as a .zbo file. It is possible to delete to curves but not the value list and use it for a new recording.

The internal data recording from the example program can be read with the Oscilloscope (Record) using the icon  or select “Tools” -> “Oscilloscope (Record)”.

7 Move on!

At this point one motor is spinning in position-controlled mode. Connect and setup more than one axis if required. The next step is to add more functionality to the application program. Here is a list of some of the features that are supported by the MACS controllers:

- Jerk limited movements
- Setup CAN bus to control some additional axis or communication to another device.
- Setup a Kinematics as Cartesian, H-bot, Scara, Dual Scara or Delta.
- Use CAM or path in your application
- Synchronization with real or virtual master
- Do Marker synchronization with reach features as marker window and marker FIFO.

And many more.

Please find more information in in the AposslDE help file. Visit the maxon support page for general motion control topics or get some help from the maxon support team.
Visit support.maxongroup.com or email to support@maxongroup.com

	If you have questions and contact the maxon support. It is helpful to attach a diagnostic report to your request. This is text file with all the current settings of the MACS controller. The file can be generated using the menu bar **Controller -> Diagnostic Report**. (make sure there is an active connection between AposslDE and MACS)