Getting Started

Safety Tips and Requirements 2
System Architecture 3
Installing the APOSS-IDE 8
Wiring the Controller 9
Starting and Exiting APOSS 10
Connecting to the Controller 12
- V24 (Serial) Connections 12
- CAN-LPT Connections 13
- CAN-USB Connections 15
- USB Connections 18
- EtherCAT Connections 19

Executing a Simple Test Program 21
Setting of Parameters 23
Connecting and Checking Encoder 25
Execute a Test Program 27

Safety Tips and Requirements

Please read these operating instructions in full. In particular, in order to be able to work with the system safely and professionally, please observe the hints and cautionary remarks highlighted with "!!!" in this manual and in the hardware manuals.

It is absolutely essential to be familiar with, and observe, the safety tips in the hardware manual.

At all times, it must be possible to switch off the controller and the motor with an EMERGENCY STOP button.

The motor must be able to turn completely freely so that a sudden jolt can not cause damage.

Requirements

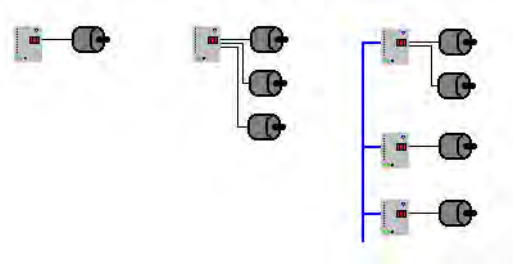
APOSS is supported on standard PC computers running the Microsoft Windows XP, Windows Vista, or Windows 7 operating system.

You should be familiar with the basic functions and terminology of the Microsoft Windows interface since this manual does not explain these basics. The manual covers the specifics of the APOSS user interface only.

System Architecture

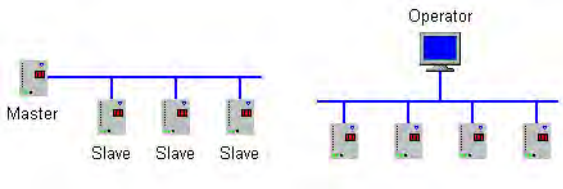
System Configuration

zub motor controllers are designed to be used in a wide variety of system configurations. These systems may be as simple as a single controller and a single motor. Or, depending on which controller is being used, there may be multiple motors connected to a single controller. Even more complex systems can be created by networking multiple controllers together. This can be done, for example, using a Controller Area Network (CAN).



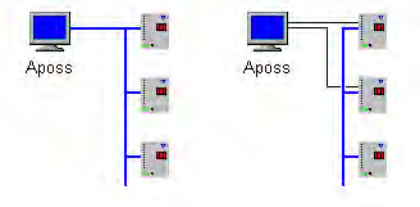
Systems are frequently designed as turnkey systems, operating on their own without an attached PC computer. These systems function by reacting to external “events” or “triggers” that are signalled to the controller by means of digital input pins. Limit switches, optical sensors, pressure sensors, etc., are examples.

Systems may also be configured in a master-slave arrangement, with a single master controller overseeing multiple slave controllers. Communication between the master and slaves can be done using the network connection, digital input and output pins, or both. Another system configuration will include an “operator console” of some sort, often a PC computer running a custom application program.



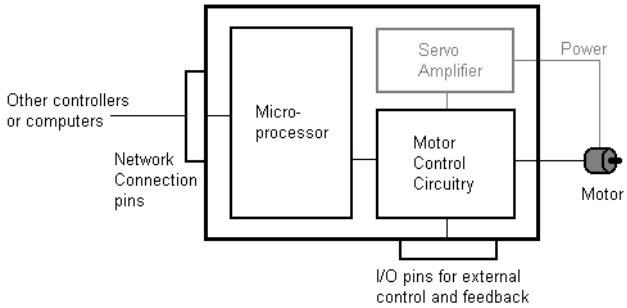
zub motor controllers will handle all of these different configurations with ease.

Each controller in a system will run its own custom application program. These programs will then tailor the behavior of the individual controllers to the exact requirements of the overall system. These programs, written in the APOSS programming language, can be created by the end user himself for his own specific application. This is done using the APOSS-IDE which has been installed on a PC computer. The PC computer is then connected to the controllers either with a private connection or with the shared network. All APOSS programs are developed, downloaded to the controllers, tested, and debugged, using the APOSS-IDE.



Inside the Controller

Like any modern computer, the controller contains a microprocessor, memory, and network communication interfaces. However, the controller also contains specialized hardware tailored specifically for motion control tasks. This includes digital I/O interfaces and other specialized external interfaces that can be used to connect encoders, sensors, and actuators (e.g. signal lamps, lifting magnets, valves, etc.). Some models of controller also include integrated servo amplifiers which allow direct connection to the motor’s power terminals.



The operating system (firmware) built into the controller's microprocessor, is an enhanced operating system that provides APOSS programs with access to all internal and external devices, interfaces, and motor control functions. The operating system provides "high-level" programming interfaces to complex functions like motor positioning, synchronization, CAN-communication, etc. These functions are then executed by the operating system as autonomous background tasks, freeing the APOSS program to focus on more important tasks. For example, position information from absolute or incremental encoders is processed and the speed, positioning, and synchronization profiles are calculated and tracked automatically.

Typical industrial and PC network interfaces (e.g. CAN, USB, and EtherCAT) allow data exchange with other systems and peripherals. Digital and analog I/O interfaces provide an additional means for APOSS programs to control external hardware, receive feedback from external hardware, and communicate with other controllers and computers. These network interfaces and I/O interfaces offer the possibility to use the controller, not only as a stand-alone unit, but also as part of a powerful and complex control system consisting of multiple interconnected devices such as PCs, PLCs, servo amplifiers, frequency converters, and motors. The system can be as simple or as complex as the application requires.

Microprocessor Memory

The controller contains a block of volatile RAM memory, which keeps its data just in power on state, and a block of non-volatile so-called flash memory, which is able to keep its information (e.g. programs, data) even in power off state. Basically, the controller executes a program and processes data in the RAM. The RAM memory is always cleared and re-initialized each time the controller is reset (e.g. power off / on). This re-initialization is done by copying the appropriate information from the flash memory to the RAM memory. All data in the flash memory is "permanent" and is preserved across power off/on cycles.

Various functions in the APOSS-IDE and in the APOSS programming language allow the user to explicitly save information from RAM memory back into flash memory. This provides the means for the user to pre-configure the controller so that there are appropriate control settings present or an application program starts automatically for the "next time" that it is powered on.

Parameters

Three different sets of parameters are maintained by the controller:

Global Parameters - Parameters affecting the controller as a whole.

Axis Parameters - Parameters affecting each single axis (i.e. each motor).

User Parameters – Parameters intended as user-defined or application dependent process configuration values.

In fact there is a copy of each parameter dataset in the flash and RAM memory. The dataset located in the flash memory keeps its settings even after a reset (e.g. power off / on). The dataset located in the RAM memory keeps its modified settings just during one program run. The dataset located in flash is copied into the one in RAM at power-up of the control unit and each time (!) a program starts. This has the advantage, that a program always starts with identical parameters data in the flash and RAM memory.

Parameter values can be changed either manually using the APOSS-IDE or by an APOSS program running in the controller.

All parameter values, which are modified by using the APOSS-IDE, are automatically updated in the RAM and flash dataset, i.e. this modification of parameters is maintained through power down and power up. On power up, they will have the same value that they had when the controller was last powered down.

Parameter values, which are modified by the APOSS program itself (by using the SET command or the SYSVAR array access) are updated in RAM dataset only and must be explicitly saved in flash memory (using the SAVE command) if that is the user's intent. Otherwise, these modified parameter settings are not present anymore after a reset or the next start of a program.

The APOSS-IDE can be used to back up the entire dataset of parameter values located in flash memory to a file on the PC hard drive. This is done using **Controller → Parameters → Save to file**. These values can then be restored at a later time by the user with **Controller → Parameters → Restore from file.** At any time, the user can reset the parameter values back to their original factory default values with **Controller → Reset → Parameter**.

Please keep in mind, that any parameter values, which were just modified in the dataset located in RAM are not automatically saved in flash memory and are not part of a backup file.

Global Parameters

These parameters affect the controller as a whole. For example, they define the communication attributes of the controller and how the controller behaves when it powers up. *See* Global Parameters in Detail for a complete list of the global parameters and their meanings. Note that individual parameters may or may not be available depending on the type and version of controller.

There is one set (in RAM and flash) of global parameters per controller.

See **Controller → Parameters →** Edit *in chapter "APOSS User Interface"* or Using Parameters *in "Basics" in chapter "Parameter Reference"* for a description of how to change the global parameters using the APOSS-IDE. Parameter values can also be viewed and modified using the Array Editor tool.

The SAVE GLBPARS command can be used by APOSS programs to save global parameters from RAM into flash memory.

Axis Parameters

These parameters affect the axis (i.e. motor) being driven by the controller. For example, they define maximum speeds and accelerations, acceleration profiles, encoder tick rates, homing and synchronization characteristics, etc. See Axis Parameters in Detail for a complete list of the axis parameters and their meanings.

Note that individual parameters may or may not be available depending on the type and version of controller.

See **Controller → Parameters →** Edit *in chapter "APOSS User Interface"* or Using Parameters *in "Basics" in chapter "Parameter Reference"* for a description of how to change the axis parameters using the APOSS-IDE. Parameter values can also be viewed and modified using the Array Editor tool.

The SAVE AXPARS command can be used by APOSS programs to save axis parameters permanent. This command copies the settings of each axis parameter dataset from the RAM into the flash memory. It is not possible to save the dataset of just a single axis.

User Parameters

There are 100 user parameters available to APOSS programs in the controller. These may be used on an "as-needed" basis. The meanings and values of these parameters are entirely user-defined. Typically, these are used as custom configuration or initialization parameters by APOSS programs.

There is one set (in RAM and flash) of user parameters per controller and all programs in the controller will share access to them.

An APOSS program can read and write user parameters by the pre-defined array SYSVAR[0x01220100 + 1 … 100]. If the motion controller is part of network, it is possible that a PLC, PC or any other control unit can get access to the user parameters by the SDO 0x2201 / subindex 1…100 via bus communication.

User parameters can be viewed and modified using the **Array Editor** tool.

The SAVE USRPARS command can be used by APOSS programs to save user parameters permanent. This command copies the content of the user parameter dataset form the RAM into the flash memory.

Note that some very old controllers do not support user parameters.

APOSS Programs

APOSS programs are compiled using the APOSS-IDE and then downloaded to the controller and started immediately in RAM or saved permanent in flash memory. The controller simultaneously supports a single "temporary" program saved and executed in RAM and, depending on the amount of available memory in the controller and the size of the programs, up to 91 different "permanent" programs saved in flash memory.

Each time the **Development → Execute** command is executed the current APOSS program is compiled and downloaded to the "temporary" program memory space in RAM. Since only one temporary program at a time is supported, executing the command again will overwrite any previous temporary program. Note that temporary programs are never saved in flash memory; they will always be lost if the controller is reset (e.g. power off / on).

Programs can be saved "permanently" by using **Controller → Programs**. This will download and save the program in both RAM and flash memory. Hence, "permanent" programs will persist through power down and power up cycles. Program source can also be optionally downloaded and saved. **Controller → Programs** can be used to view and manage all the "permanent" programs saved on the controller.

The controller is designed to be able to "auto start" any of the "permanent" programs during power up. As well as starting a predefined program, the controller also has the ability to select a program to be started based on signals connected to the controller's input pins. These pins could be set by a PLC, for example. The configuration of the auto start program is done by setting various global parameters to appropriate values. For details, see **Controller → Parameters →** Edit *in chapter APOSS User Interface*.

DIM Arrays

The APOSS programming language supports different types of arrays: DIM arrays, global arrays, and local arrays. See also Variants Comparison: "Arrays, Variables: global, local?".

All DIM arrays (i.e. arrays defined by the DIM statement at program start) will be shared by all programs on the controller. DIM arrays are maintained and managed in memory independently from any program that uses them.

DIM arrays are internally numbered consecutively starting at 0 and any number of arrays can be defined (limited by the amount of available memory on the controller). Arrays can be viewed and managed using the **Array Editor** or the **CAM Editor.** Arrays will also be created automatically if they do not already exist when an APOSS program that uses them is executed.

The number of DIM arrays in use may be defined differently in different programs and not all programs need to define or use all arrays. But the size of each DIM array in use must correspond for all programs! The naming of the DIM arrays has no meaning. DIM arrays are internally identified by their consecutive number, given by the position within the DIM definition section of a program. Hence, it is possible to assign different names to the same DIM array in different programs, but it is not possible to assign a different array size. DIM arrays can be used to exchange data in between different programs or to save a large amount of data.

DIM arrays created or changed using the APOSS-IDE are also saved in flash memory and are therefore maintained through power down and power up cycles. On power up, they will have the same value that they had when the controller was last powered down. DIM arrays created or changed using an APOSS program are changed in RAM memory only and must be explicitly saved in flash memory (using the SAVE ARRAYS command) if that is the user's intent.

The APOSS-IDE can be used to back up the entire set of DIM array values to a file on the PC hard drive. This is done using **Controller → Parameters → Save to file**. These values can then be restored at a later time by the user with **Controller → Parameters →** Restore from file**.** At any time, the user can delete all DIM arrays with **Controller → Reset → Arrays**.

Installing the APOSS-IDE

The APOSS-IDE software may be freely downloaded from the **www.zub.ch** website. It may also be requested free of charge by e-mail (info@zub.ch), by telephone (+41-41-3480032), or by fax (+41-41-3480039).

Download and install the software as follows:

1. Visit the **www.zub.ch** website and select the **Downloads** page. Press the **Login** button and then select the **Software** page.
2. Download the APOSS-IDE install program to your hard drive.
3. Execute the install program by double-clicking on it.
4. Follow the instructions displayed by the install program.

See Requirements for details of supported operating systems.

The APOSS-IDE can be installed over top of a previously installed version of the APOSS-IDE. It is not necessary to uninstall the old version before installing the new version.

!!! If you will be using a CAN-LPT module to connect to controllers and this is the first time that the APOSS-IDE has been installed on the computer, or if a very old version of the APOSS-IDE is being replaced, then you will need to reboot the computer in order to complete the installation of the Windows device drivers.

!!! If you will be using a CAN-USB stick to connect to controllers or using a USB connection to MACS4 controllers, then you must install the APOSS-IDE before plugging in either the CAN-USB stick or the MACS4 controller for the first time. Otherwise, Windows will install the wrong device driver software and it will not be possible to connect to the controller.

Uninstalling the APOSS-IDE

The APOSS-IDE software can be uninstalled in the normal way using the Windows Control Panel. Do this as follows:

1. Ensure that all CAN-USB sticks and MACS4 controllers have been unplugged. Otherwise, Windows will not unload the device drivers and the drivers will not be uninstalled.
2. Start the Windows **Control Panel** and select → **Add or Remove Programs** (for Windows XP). Later operating systems such as Windows Vista or Windows 7 will be slightly different.
3. Look for all of the following in the list. If found, then select them and remove them.
   - APOSS Software
   - APOSS Version #.#
   - Windows Driver Package - zub USB COM Driver Package
   - Windows Driver Package - zub USB Driver Package
   - Windows Driver Package - FTDI CDM Driver Package

Wiring the Controller

Before starting APOSS, the controller should be wired according to the "Installation Manual and Hardware Reference" corresponding to the specific type of controller being used. These documents may be downloaded from the **www.zub.ch** website. They may also be requested free of charge by e-mail (info@zub.ch), by telephone (+41-41-3480032), or by fax (+41-41-3480039).

Download them as follows:

1. Visit the **www.zub.ch** website and select the **Downloads** page. Press the **Login** button and then select the **Manuals Hardware** page.
2. Click on the link for the appropriate document to start the download.

!!! If you will be using a CAN-USB stick to connect to controllers or using a USB connection to MACS4 controllers, then you must install the APOSS-IDE before plugging in either the CAN-USB stick or the MACS4 controller for the first time. Otherwise, Windows will install the wrong device driver software and it will not be possible to connect to the controller.

Before turning the controller on…

⚠ For safety reasons, at this point in time it should not be possible for the motor to accidentally drive equipment. A newly installed controller can drive a motor in unpredictable ways and damage equipment before the wiring has been tested and the internal controller parameters set for proper operation. Make sure that the motor is not connected to equipment before turning the controller on.

Wiring the Controller to the Computer

Controllers can be connected to the computer running APOSS, using any of several different types of physical connections. APOSS can connect to more than one controller at the same time if different physical connections are used for each controller. Some physical connections (e.g. CAN and USB) also support connections to multiple controllers using the same physical connection. Note that not all types of controllers support all physical connections.

The following types of physical connections are available:

V24	Serial connection using either an RS232 COM port or a USB port with a USB-to-RS232 converter
CAN-LPT	CAN network connection using an LPT port with a CAN-LPT module
CAN-USB	CAN network connection using a USB port with a CAN-USB stick
USB	Direct connection using a USB port
EtherCAT	EtherCAT network connection

Please see the "Installation Manual and Hardware Reference" for the specific controller to determine which physical connections are supported by that controller and for instructions on how to wire the actual physical connection.

Once the physical controller connection is complete, then the controller can be powered on and APOSS started.

If this is the first time that a CAN-USB stick or a MACS4 controller is connected, then one or more messages similar to the following will be displayed at the bottom-right corner of the screen. These indicate that the computer has recognized the device and loaded the device drivers for it.

!!! If you see a "Found New Hardware Wizard" dialog, then the APOSS-IDE has not been installed. In this case, click the "Cancel" button, unplug the controller, install the APOSS-IDE, and finally plug the controller back in.

Starting and Exiting APOSS

APOSS can be started in any of the following ways:

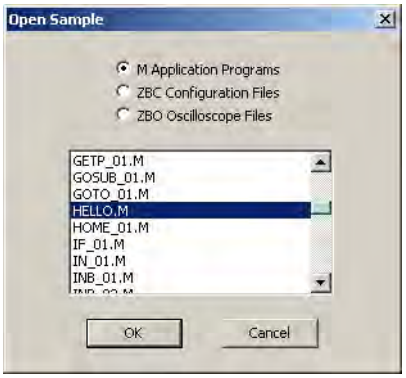
1. In the Windows task bar, click on **Start** → **All Programs** → **zub machine control AG** → **APOSS** to start APOSS. Note that "zub machine control AG" is the default install location. The user is free to choose a different location during the APOSS installation.
   If a desktop icon was created during the APOSS install, then double-clicking on the icon will start APOSS. The APOSS icon is shown below.

2. If a Quick Launch icon was created during the APOSS install, then single-clicking on the icon will start APOSS. Quick Launch icons appear in the bottom-left corner of the Windows screen as is shown below.

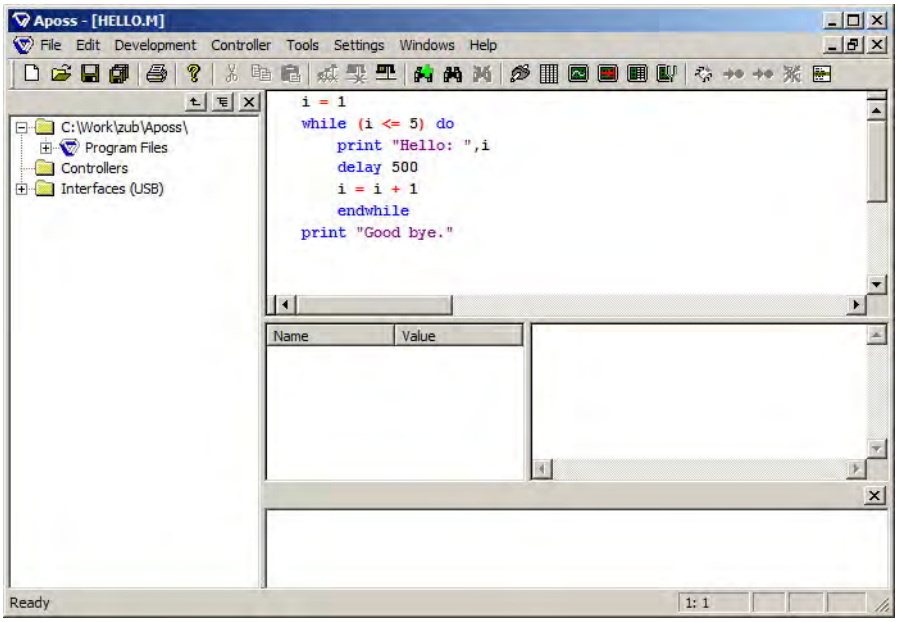
3. If an APOSS file (i.e. an ".m", ".zbo", ".zbc" file, etc.) already exists, then double-clicking on the file in the Windows folder should automatically start APOSS and open that particular file.

When APOSS starts, it will try to automatically open the files that were previously open when APOSS was last used (unless APOSS was started by double-clicking on an APOSS file, in which case that file is opened).

If APOSS cannot find any last file to open, then APOSS may display the "File Open" dialog. This can happen when APOSS is used for the first time since there will be no last file. In this case, the "File Open" dialog should be cancelled and **File** → **Sample** used to open a sample ".m" file. This will display the following dialog.



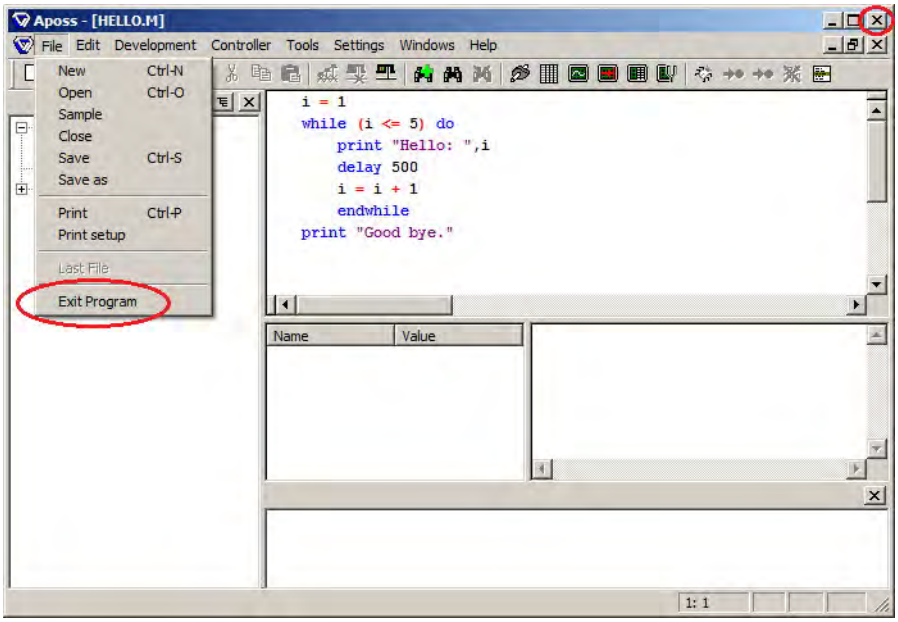
Select "HELLO.M" and click the → **OK** button. The initial APOSS window should be similar to that shown below.



If this is the first time that APOSS has been installed on the computer, then APOSS will start in the language selected during the APOSS installation. This language can be changed after APOSS starts, by clicking on **Settings → Language** and selecting one of the available languages in the subsequent dialog. Note that the new language will only become active when APOSS is started the next time. So click **File → Exit Program** and then start APOSS again.

Exiting APOSS

Exiting APOSS can be done at any time simply by clicking on the Windows "Close" button in the top-right corner of the main APOSS window or by clicking on **File → Exit Program**.



!!! Exiting APOSS will <u>not</u> abort or end any program executing on the controller. If you want to abort or end the program executing on the controller, then you must do so <u>before</u> exiting APOSS. To do this, you must have an APOSS Edit Window open and you must have opened a connection to the controller. Pressing the **Esc** key will then terminate the program executing on the controller.

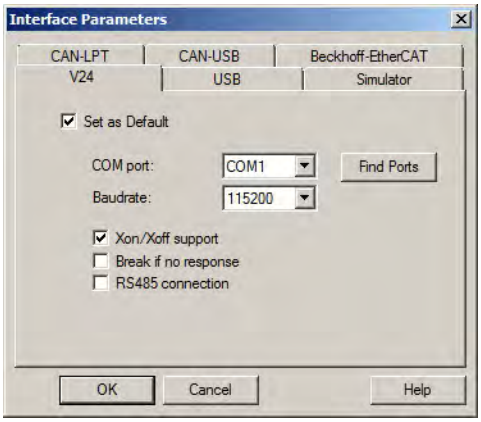
Connecting to the Controller

Before trying to connect APOSS to the controller, check whether the APOSS interface connections settings are correct for the physical connection being used by the controller. Do this as follows:

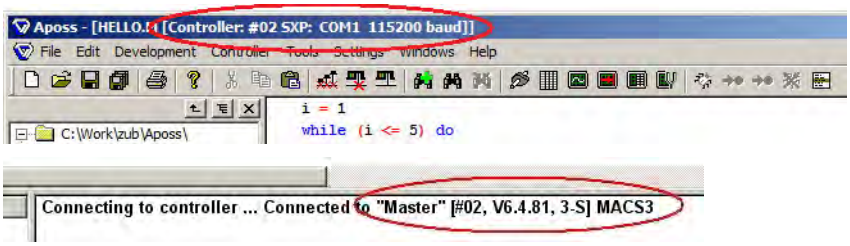
1. If an APOSS Edit Window is not already open, then open an APOSS Edit Window by either creating a new APOSS program (.m) file, opening an existing APOSS program file, or opening a sample APOSS program file. Do this using the **File** → **New**, **File** → **Open**, or **File** → **Sample** menu command.
2. Click on **Settings** → **Interface** to open the Interface Parameters dialog.
3. Select the tab according to the type of physical connection being used for the controller. Then set the interface parameters as described in one of the following sections.

V24 (Serial) Connections

V24 connections are serial connections that use either an RS232 COM port or a USB port with a USB-to-RS232 converter.



1. Check the → ☑ **Set as Default** box.
2. Select the **COM port** that you will be using. If the desired COM port is not listed, then click the **Find Ports** button and check again for the COM port. If a USB-to-RS232 converter is being used, then it will be necessary to plug the converter into the computer before Windows will be able to find the associated COM port.
3. Choose the **baudrate** that the connection will use. Note that some older controllers support only 9600 baud. For these controllers, the baud rate setting is ignored.
4. ☑ **Xon/Xoff support** should normally be enabled.
   It should be disabled if a USB-to-RS232 converter is being used.
5. **Break if no response** should be disabled unless it is necessary to enable it. APOSS cannot successfully establish a connection to some older controllers if the controller is executing a program. If APOSS returns the message "No Controller found!" when an old controller is properly connected to the correct COM port, then you should enable the → ☑ **Break if no response** setting and try again. Note that this will abort the program executing on the controller!
6. ☑ **RS485 connection** should be enabled if an RS232-to-RS485 converter has been inserted into the connection. This allows the APOSS-IDE to adjust the communication timing to compensate for differences between RS232 and RS485.
7. Click the **OK** button. This will close the dialog and return you to the APOSS Edit Window.
8. In the Edit Window, press the **Esc** key. APOSS will then try to connect to the controller. If this is successful, then the controller will be identified in both the Edit Window title and the Communication Window.



The above example shows a successful connection. The window title shows that the controller has a CAN ID number of 2 (#02), that a serial protocol connection (SXP) is being used on COM1, and that the connection is running at 115200 baud. The Communication Window shows the controller name (Master), CAN ID number (#02), firmware version (6.4.81), hardware options (3-S), and the type of controller (MACS3).

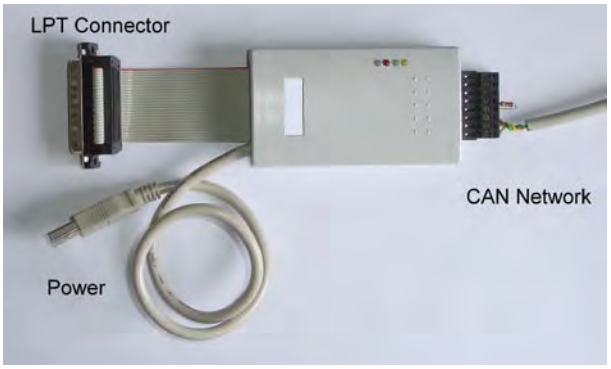
If "No controller found!" is reported in the Communication Window, then the connection was unsuccessful. Please check for the following:

1. The controller is powered on.
2. The serial cable is firmly connected to the computer.
3. The serial cable is connected to the correct COM port. See below if a USB-to-RS232 converter is being used.
4. The serial cable is correctly wired to the controller serial pins (see the "Installation Manual and Hardware Reference" for the controller).
5. For some very old controllers, APOSS cannot successfully establish a connection if the controller is executing a program. Enable the → ☑ **Break if no response** setting and try again. Note that this will abort the program executing on the controller!

!!! If a USB-to-RS232 converter is being used, then the correct virtual COM port must be selected in the Interface Parameters dialog. The correct COM port can be identified by opening the Windows **Control Panel**, selecting **System**, selecting **Hardware**, clicking the **Device Manager** button, and finally opening the **Ports (COM & LPT)** item. Look for the COM port that corresponds to the USB-to-RS232 device.

CAN-LPT Connections

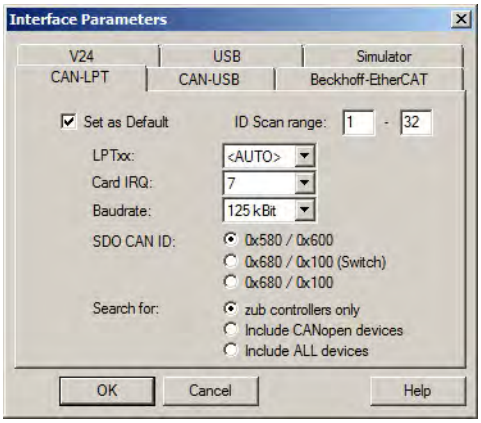
CAN-LPT Adapter with power supply via USB:



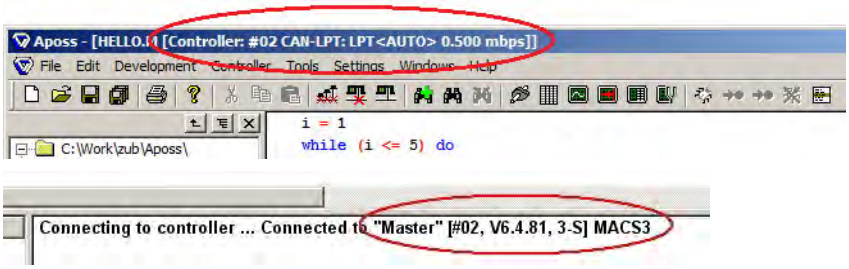
Art. Nr. 001019

CAN-LPT connections will use a CAN-LPT adapter (similar to that shown above) to connect the computer's parallel printer port (LPT port) to a CAN network.

!!! CAN-LPT adapters are not supported on Microsoft Windows 7 and later systems.



1. Check the → ☑ **Set as Default** box.
2. The **LPT port** can normally be left set to <AUTO>. It will be necessary to set this to a specific LPT port if multiple CAN-LPT modules are being used.
3. Set the **Card IRQ** according to the manufacturer's installation documentation that came with the CAN-LPT module.
4. Set the **Baudrate** to the baud rate of the CAN network.
5. The **SDO CAN ID** should be left set to "0x580 / 0x600" unless there is a reason to change it. These are the CAN ID numbers that will be used to identify SDO messages sent to and received from the controller. Some controller applications may use these ID numbers for other purposes. In these cases, the alternate ID numbers may be used. Note that some older controllers may not support the alternate ID numbers. If the option labeled "0x680 / 0x100 (Switch)" is selected, then the initial connection will be made using "0x580 / 0x600". If this connection is successful, then APOSS will automatically switch to the alternate ID numbers "0x680 / 0x100". If this switch fails, then APOSS will continue to use the original ID numbers.
6. **Search for** can be left set to "zub controllers only". APOSS is capable of connecting to other types of CAN devices, not only to zub controllers. In these cases, APOSS will have limited functionality.
7. The **ID Scan range** specifies the range of CAN ID numbers that APOSS will use when searching the CAN network for controllers. This range must include the CAN ID of the controller to which the connection is being made. See the "Installation Manual and Hardware Reference" for the controller to see how to set the CAN ID number.
8. Click the **OK** button. This will close the dialog and return you to the APOSS Edit Window.
9. In the Edit Window, press the **Esc** key. APOSS will then try to connect to the controller. If this is successful, then the controller will be identified in both the Edit Window title and the Communication Window. If more than one controller was found in the ID scan range, then a "Select Controller" dialog (similar to that shown below) will be displayed. Select the appropriate controller and click the → **OK** button.



The above example shows a successful connection. The window title shows that the controller has a CAN ID number of 2 (#02), that a CAN-LPT connection (CAN-LPT) is being used on the default LPT port (LPT<AUTO>), and that the CAN network is running at 0.5 mbps. The Communication Window shows the controller name (Master), CAN ID number (#02), firmware version (6.4.81), hardware options (3-S), and the type of controller (MACS3).

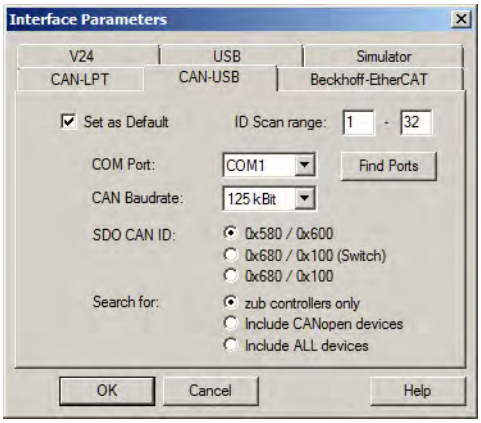
If "No controller found!" is reported in the Communication Window, then the connection was unsuccessful. Please check for the following:

1. The ID Scan range contains the CAN ID of the controller.
2. The baudrate of the CAN network has been set correctly.
3. The controller is powered on.
4. The CAN-LPT module is firmly connected to the computer and is powered on.
5. The CAN-LPT module is connected to the correct LPT port if more than one CAN-LPT module is being used.
6. The CAN network is correctly wired to both the controller CAN network pins (see the "Installation Manual and Hardware Reference" for the controller) and to the CAN-LPT module (see the manufacturer's installation documentation that came with the CAN-LPT module).

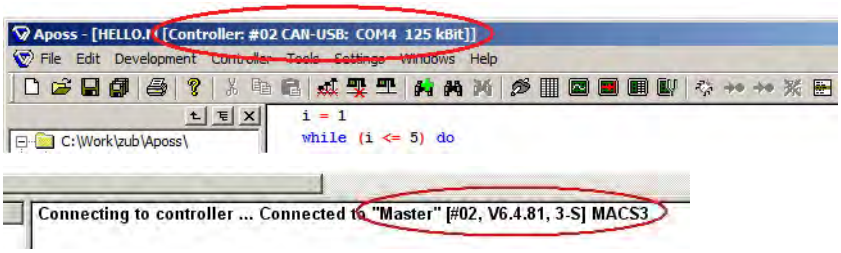
CAN-USB Connections

Art.Nr. 001150 USB CAN-Stick without integrated bus terminator
Art.Nr. 001151 USB CAN-Stick with integrated bus terminator

CAN-USB connections will use a CAN-USB stick (similar to that shown above) to connect a USB port to a CAN network.



1. Check the → ☑ **Set as Default** box.
2. Select the **COM Port** that you will be using. If the desired COM port is not listed, then click the **Find Ports** button and check again for the COM port. It will be necessary to plug the CAN-USB stick into the computer before Windows will be able to find the associated COM port.
3. Set the **CAN Baudrate** to the baudrate of the CAN network.
4. The **SDO CAN ID** should be left set to "0x580 / 0x600" unless there is a reason to change it. These are the CAN ID numbers that will be used to identify SDO messages sent to and received from the controller. Some controller applications may use these ID numbers for other purposes. In these cases, the alternate ID numbers may be used. Note that some older controllers may not support the alternate ID numbers. If the option labeled "0x680 / 0x100 (Switch)" is selected, then the initial connection will be made using "0x580 / 0x600". If this connection is successful, then APOSS will automatically switch to the alternate ID numbers "0x680 / 0x100". If this switch fails, then APOSS will continue to use the original ID numbers.
5. **Search for** can be left set to "zub controllers only". APOSS is capable of connecting to other types of CAN devices, not only to zub controllers. In these cases, APOSS will have limited functionality.
6. The **ID Scan range** specifies the range of CAN ID numbers that APOSS will use when searching the CAN network for controllers. This range must include the CAN ID of the controller to which the connection is being made. See the "Installation Manual and Hardware Reference" for the controller to see how to set the CAN ID number.
7. Click the **OK** button. This will close the dialog and return you to the APOSS Edit Window.
8. In the Edit Window, press the **Esc** key. APOSS will then try to connect to the controller. If this is successful, then the controller will be identified in both the Edit Window title and the Communication Window. If more than one controller was found in the ID scan range, then a "Select Controller" dialog (similar to that shown below) will be displayed. Select the appropriate controller and click the → **OK** button.

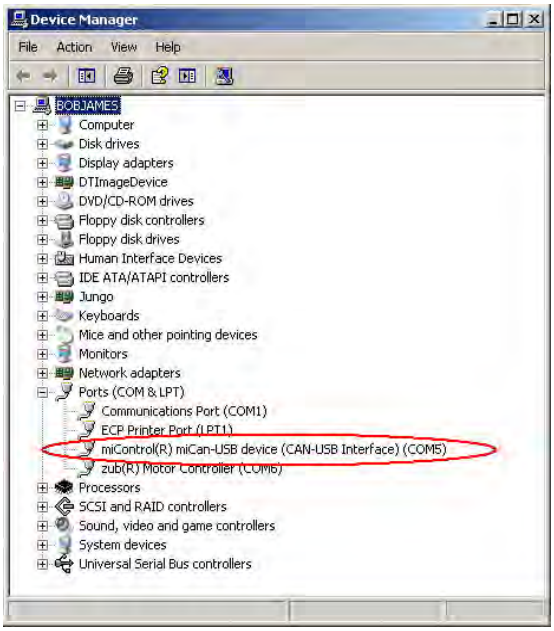


The above example shows a successful connection. The window title shows that the controller has a CAN ID number of 2 (#02), that a CAN-USB connection (CAN-USB) is being used on COM4, and that the CAN network is running at 125 kBit. The Communication Window shows the controller name (Master), CAN ID number (#02), firmware version (6.4.81), hardware options (3-S), and the type of controller (MACS3).

If "No controller found!" is reported in the Communication Window, then the connection was unsuccessful. Please check for the following:

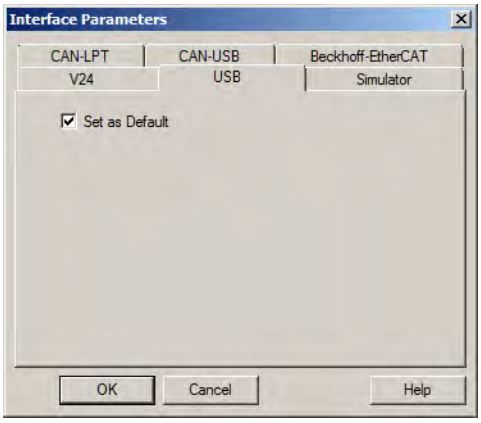
1. The ID Scan range contains the CAN ID of the controller.
2. The baudrate of the CAN network has been set correctly.
3. The controller is powered on.
4. The CAN-USB stick is firmly connected to the computer.
5. The correct COM port has been selected in the Interface Parameters dialog. See below to determine the correct COM port.
6. Some USB hubs do not function properly. If the CAN-USB stick is plugged into a port on a USB hub, then try plugging the CAN-USB stick into a USB port on the actual computer.
7. The CAN network is correctly wired to both the controller CAN network pins (see the "Installation Manual and Hardware Reference" for the controller) and to the CAN-USB stick (see the manufacturer's installation documentation that came with the CAN-USB stick).

!!! The CAN-USB stick uses a virtual COM port. The correct COM port can be identified by opening the Windows **Control Panel**, selecting **System**, selecting **Hardware**, clicking the **Device Manager** button, and finally opening the **Ports (COM & LPT)** item. Look for the COM port that corresponds to the CAN-USB stick. It should appear similar to that shown below.

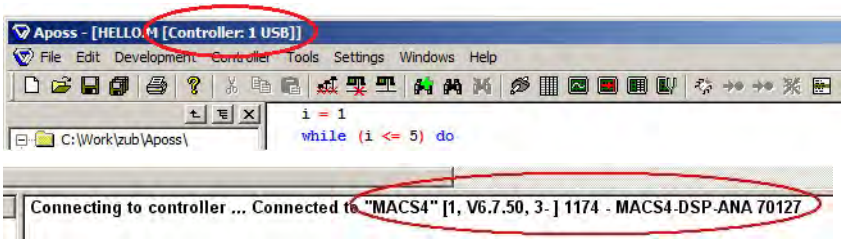


USB Connections

If the controller supports it, then a USB connection can be made directly from the computer to the controller.



1. Check the → ☑ **Set as Default** box. USB connections require no further setup.
2. Click the **OK** button. This will close the dialog and return you to the APOSS Edit Window.
3. In the Edit Window, press the **Esc** key. APOSS will then try to connect to the controller. If this is successful, then the controller will be identified in both the Edit Window title and the Communication Window. If more than one controller was found, then a "Select Controller" dialog (similar to that shown below) will be displayed. Select the appropriate controller and click the → **OK** button.



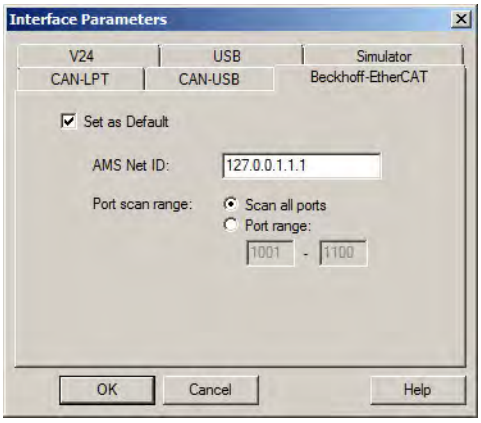
The above example shows a successful connection. The window title shows that the controller has an ID number of 1 and that a USB connection (USB) is being used. The Communication Window shows the controller name (MACS4), ID number (1), firmware version (6.7.50), hardware options (3-), and the type of controller (1174 - MACS4-DSP-ANA 70127).

If "No controller found!" is reported in the Communication Window, then the connection was unsuccessful. Please check for the following:

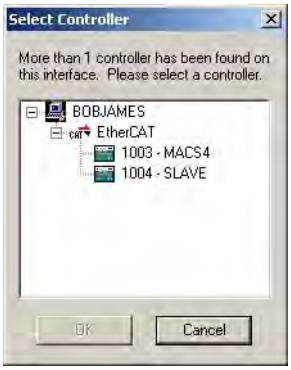
1. The controller is powered on.
2. The USB cable is firmly connected to both the computer and the controller.
3. Some USB hubs do not function properly. If the USB cable is plugged into a port on a USB hub, then try plugging the cable into a USB port on the actual computer.

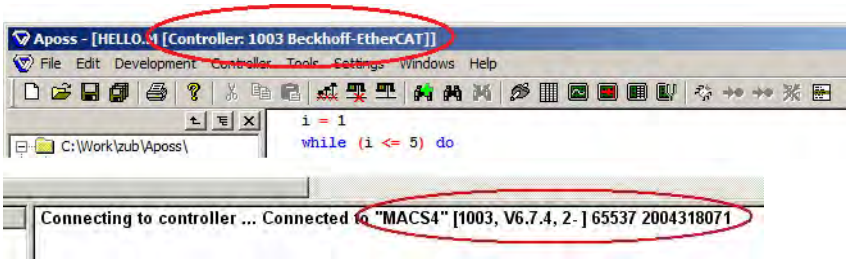
EtherCAT Connections

If the controller supports it, then an EtherCAT connection can be made from the computer to the controller.



1. Check the → ☑ **Set as Default** box.
2. Enter the **AMS Net ID**. This will be an address of six numbers separated by dots (e.g. 5.1.131.8.3.1).
3. The **Port scan range** can normally be left set to all ports unless there is some reason to restrict it.
4. Click the **OK** button. This will close the dialog and return you to the APOSS Edit Window.
5. In the Edit Window, press the **Esc** key. APOSS will then try to connect to the controller. If this is successful, then the controller will be identified in both the Edit Window title and the Communication Window. If more than one controller was found, then a "Select Controller" dialog (similar to that shown below) will be displayed. Select the appropriate controller and click the **OK** button.





The above example shows a successful connection. The window title shows that the controller has an ID number of 1003 and that an EtherCAT connection is being used. The Communication Window shows the controller name (MACS4), ID number (1003), firmware version (6.7.4), hardware options (2-), and the type of controller (65537 2004318071).

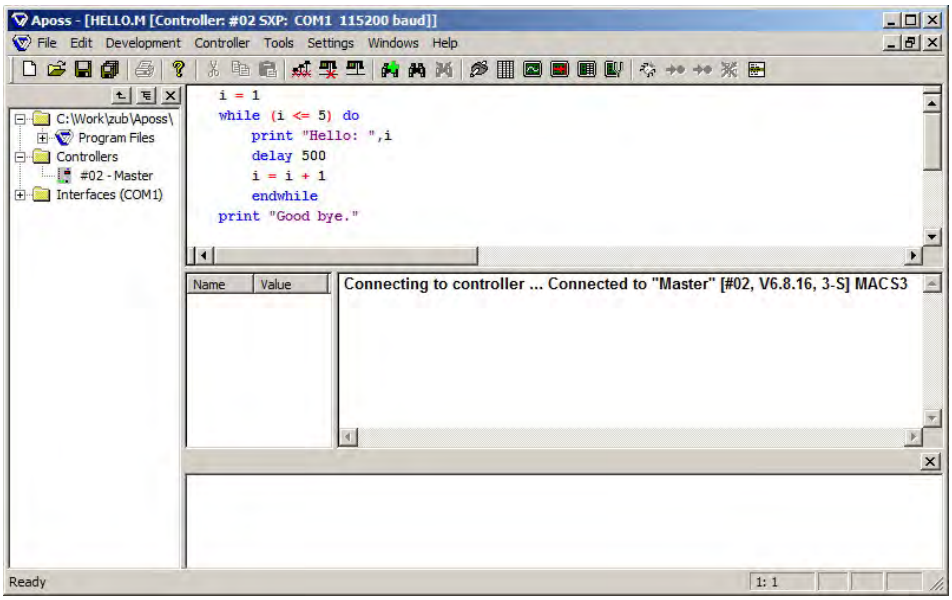
If "No controller found!" is reported in the Communication Window, then the connection was unsuccessful. Please refer to the documentation that came with the EtherCAT hardware and software. Then check for the following:

1. The controller is powered on.
2. The EtherCAT bus is up and running in either “Config” or “Realtime” mode.
3. The bus state of the controller is “OP” (viewable using the EtherCAT System Manager).
4. The AMS Address is correct.

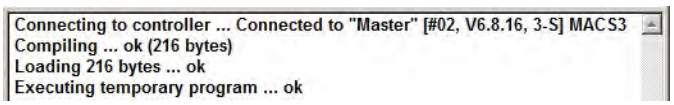
Executing a Simple Test Program

The steps in the following example, illustrate how to execute an APOSS program on the controller. The sample program used here is a very simple program that can be executed on the controller. It does not require any motor to be connected and any motor that is connected will be ignored. The program will simply print “Hello” five times, then print “Good bye”, and then stop.

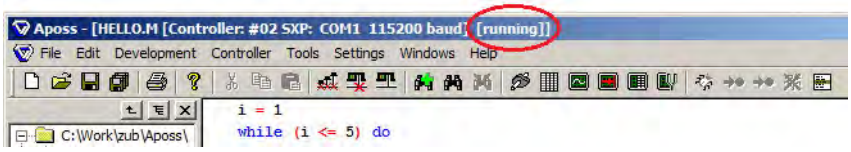
1. Start APOSS, open the **File → Sample** program “HELLO.M” and press the **Esc** key to connect to the controller.
2. At this point, the “HELLO.M” program is ready to be executed and the APOSS Window should appear similar to that shown below.



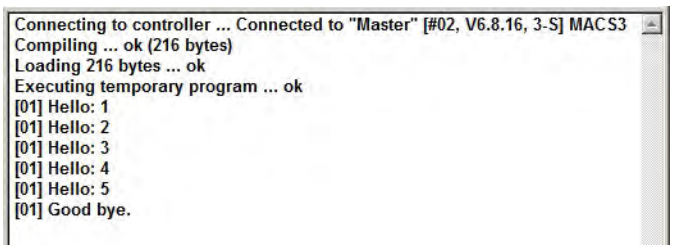
3. Start the program with **Development → Execute** or by pressing the **F5** key. This will automatically do the following: Compile the APOSS program, download the compiled program to the controller, and start the program executing on the controller. Feedback for each of these actions is displayed in the lower-right of the window as shown below.



4. The APOSS Window title will indicate that a program is running. This is shown below. Note that the title is only updated periodically so the “running” status may not be shown if the program starts and ends quickly.



5. As the program executes on the controller, any messages printed by the program will be displayed in the lower-right of the APOSS Window. In this example, the “[01]” at the beginning of each line is the ID number of the controller that is printing the message.



6. The program can be modified by making whatever changes are needed in the upper part of the APOSS Window. Once the changes are complete, the program can be rerun simply by using **Development** → **Execute** or by pressing the **F5** key again.

Stopping Program Execution

At any time, if it is necessary to stop a program executing on a controller, then this can be done by pressing the **Esc** key or using **Development** → **Break**.

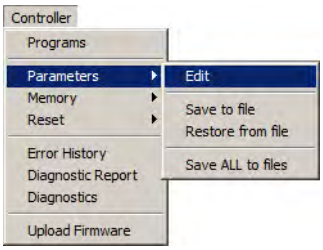
If the controller is operating a motor when program execution is stopped, then the controller will stop the motor using the maximum allowable deceleration.

Setting of Parameters

The following parameters must always be checked and if necessary adjusted. Depending on the requirements of the application, it might be necessary to adjust other parameters as well.

For the other parameters you can use the factory settings at first and then optimize the controller as needed at a later point in time using **Tune Oscilloscope.**

Controller → Parameters → Edit and select the controller and the axis for which you want to adjust the settings.]



Then select the tab for the parameter group that you wish to set (e.g. **Encoder Data**), and enter the values in the corresponding fields. Click on **OK** in order to load the new parameter values into the controller.

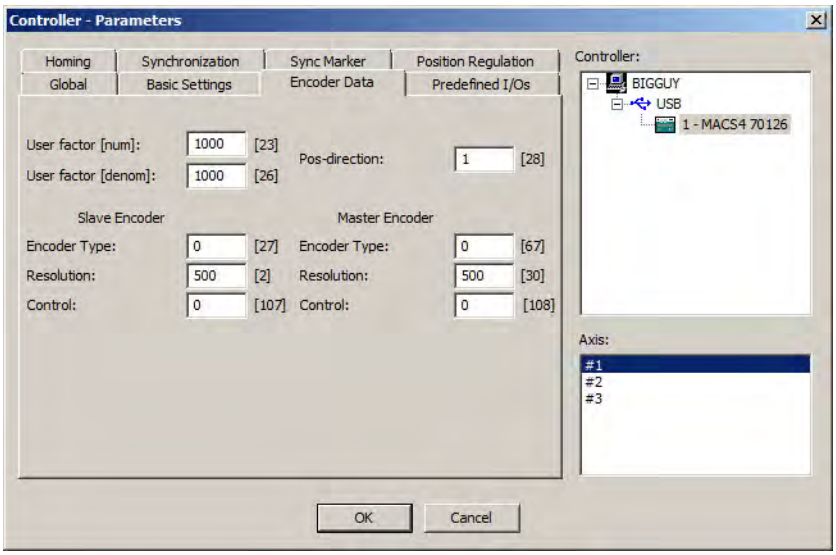
For a detailed description of all global parameters and axis parameters, please refer to *chapter* Parameter Reference and for the dialog field handling refer to *Programming with APOSS* in section **Controller** menu **→ Parameters.**

Encoder Data

The parameters ***Encoder Type, Resolution*** and ***Pos.direction*** must be set:

- Type of the used encoder: ENCODERTYPE (27)
- Resolution of the encoder (counts/revolution): ENCODER (2)
- Slave Position direction: POSDRCT

The parameters for the Master Encoder MENCODERTYPE (67) and MENCODER (30) are only relevant for synchronizing applications.



Basic Settings

The parameters ***Maximum Velocity*** and ***Shortest Ramp*** must be set:

- ***Maximum Velocity*** of the shaft where the encoder is mounted in RPM: VELMAX (1)
  For synchronizing, the setting must be at least the same as the maximum velocity of the master in order to be able to synchronize.
- ***Shortest Ramp*** in ms: RAMPMIN (31)

Homing

Homing is not necessary in standard synchronization applications and applications using an absolute encoder.

When using an incremental encoder the controller must be run to home after being switched on: HOME_FORCE (3). The necessary settings depend on the application: HOME_VEL (7).

Synchronization

The parameters ***Syncfactor M:S*** must be set only for applications with synchronization: SYNCFACTM (49) and SYNCFACTS (50).

Sync Marker

The following parameters are only relevant when using synchronizing with marker correction (SYNCM):

Master- and ***Slave-Marker Count:*** SYNCMARKM (52) and SYNCMARKS (53)

Master- and ***Slave-Marker Distance:*** SYNCMPULSM (58) and SYNCMPULSS (59).

Master- and ***Slave-Marker Type:*** SYNCMTYPM (60) and SYNCMTYPS (61)

Connecting and Checking Encoder

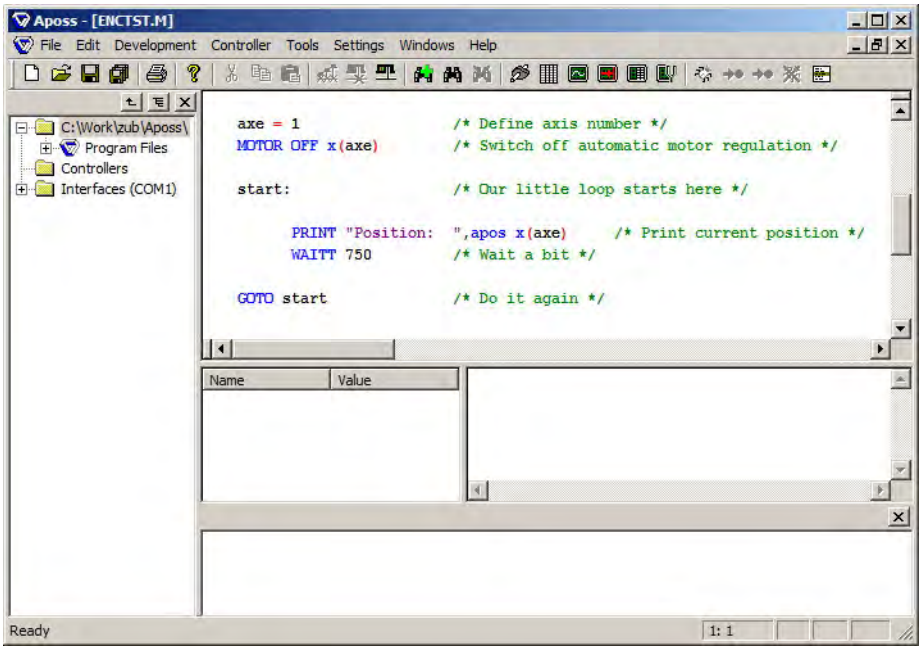
Before trying to run the motor for the first time, verify that the encoder is connected properly and that the encoder parameters have been set properly. This is important since the controller uses the feedback from the encoder to control the motor. If the encoder is not set up properly, then the controller can drive the motor in unpredictable ways causing possible damage.

Remember that the encoder on each axis must be tested if the controller supports more than one axis.

If you have not yet done so, now is the time to connect and test the encoder with each individual axis.

⚠ The motor power should NOT be connected.

Check the encoder connections by means of the test program “Enctst.m”. In the menu **File → Sample**, select the "ENCTST.M" sample program.



Axis 1 is marked. Click on **Development → Execute [F5]** in order to start the test program.

If you want to test another axis, change the axis number defined in the program to the desired axis number, e.g.

```
axe = 3                    /* Define axis number */
```

The position 0 is registered in the communications window.

If you turn the motor by hand (the motor should not be connected!), you can test whether the encoder functions: the position is continuously registered in the communications window. For a full rotation you should receive 4 times the value of the resolution of the encoder. For example, you should see 2000 if the Encoder counts per revolution is 500.

!!! Repeat this test for each axis.

Checking Master Encoder for Synchronizing

To check the master encoder, change the test program: replace the command APOS by MAPOS in “Enctst.m” and start the test program again. Then run the master forward. The master position should also increase.

If the position decreases, then you must swap the master encoder's A and B channel.

Checking Direction of Rotation

By turning the axis, it is possible to verify that the direction of rotation is correct. When viewing the front of the motor (i.e. looking towards the motor along the axis), a clockwise rotation should cause the impulse generator to count up.

If not, then encoder channels A and B, as well as /A and /B, must be exchanged. However, it is often easier to swap two motor-phases. Alternatively, you can simply use the parameter ***Rotational Direction*** POSDRCT (28) to invert the evaluation of the encoder information.

What to do, if the encoder doesn't work?

This could be a result of incorrect cable installation. Measure the signals coming from the encoder and compare them to the values listed in the specifications. Check whether the connection was made according to the specifications.

Ending the Encoder Check

Repeat the test with each axis.

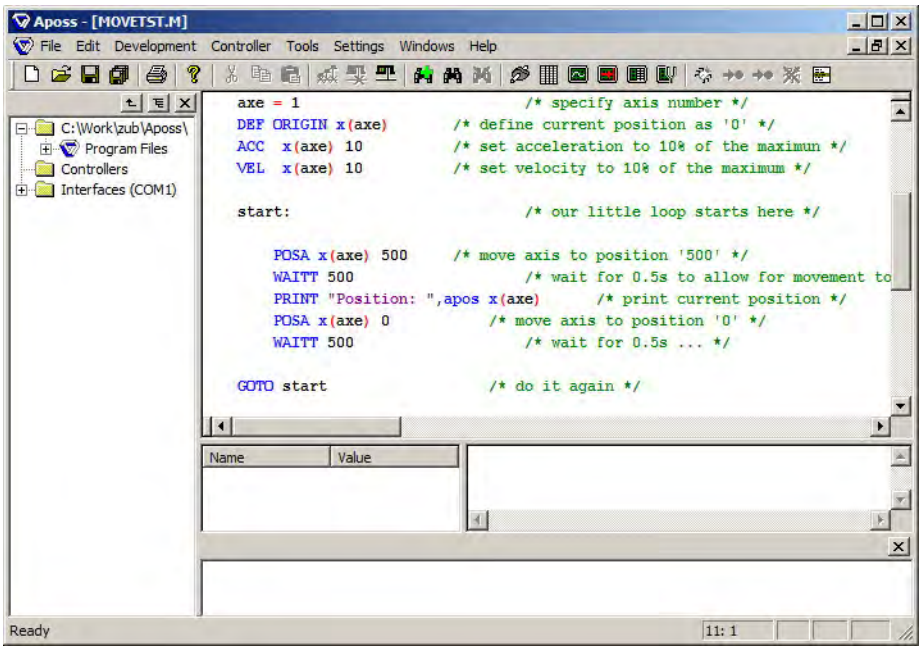
End the test of the encoder with the **Esc** key and close the test program with **File → Close**. A successful test of the encoder is required before any further starting up of operations.

Execute a Test Program

Now connect the motor to the power supply, make sure that the motor can turn completely freely.

⚠ The motor must be provided with an EMERGENCY STOP button. The motor may turn at maximum velocity in the wrong direction if the encoder has been connected incorrectly.

In the menu **File** → **Sample**, select the "MOVETST.M" sample program.



Click on **Development** and start the test program with → **Execute** or **F5**.

The test is successful if the motor runs slowly back and forth and position 500 is registered.

Repeat the execution of the test program with each axis of the controller. Change the axis number defined in the program to the desired axis number, e.g.

```
axe = 3                    /* Define axis number */
```

End the test with **Esc** and → **Close** the **File**.

What to do, if ...? Read the following tips, if there is a problem with the motor or position error is reported.

What to do, if the motor sets off uncontrollably or vibrates heavily?

Turn off the motor immediately with the EMERGENCY STOP button if it vibrates heavily or sets off uncontrollably.

If the motor sets off uncontrollably, but the encoder test was previously successful, then decrease the ***Proportional Factor*** KPROP (11). (See Optimizing the PID control.)

What to do, if the motor does not move?

If the motor doesn't move at all, then the ***Proportional Factor*** of the PID filter is probably too low or the amplifier has not been enabled. Increase the ***Proportional Factor*** KPROP (11) for the PID control. (see Optimizing the PID control).

!!! If you have more than one controller connected, proceed as described above with each individual controller and each axis.

What to do, if the motor vibrates heavily?

If the motor vibrates heavily, then you have to optimize the PID control and to adjust the other parameters of the controller: reduce either the ***Proportional Factor*** KPROP (11) or increase the ***Derivative Factor*** KDER (12).

What to do, if a position error is reported?

If the drive stops due to a "tolerated position error is exceeded" message, it is possible to determine whether the drive was rotating in the wrong direction by comparing the curves of the set and actual values using the **Tune Oscilloscope**.

Check the connections of the motor or encoder. If the connections are correct, then it is necessary to increase the ***Tolerated Position Error*** POSERR (15).