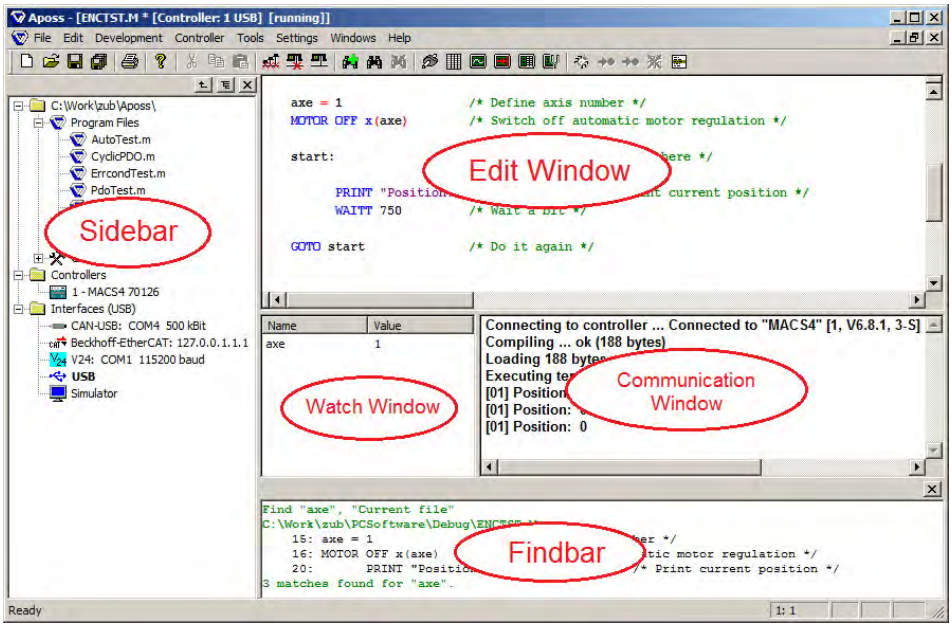
APOSS User Interface

You should be familiar with the basic functions and terminology of the Microsoft Windows interface. This chapter only explains the specifics of the APOSS User Interface.

The APOSS Window

The APOSS Window allows simultaneous access both to an APOSS program and to a controller. Multiple APOSS windows can be opened, each accessing a different APOSS program and a different controller.



From top to bottom:

Title Bar — The **Title bar** displays the name of the APOSS program. If a controller is connected, then the controller ID number and the connection interface type will also be displayed. In the event of a controller error, the error is also shown in the title bar.

Menu Bar — The Menu Bar offers the selection of the corresponding functions.

Tool Bar — The **Tool bar** offers "single-click" access to several commonly used functions. These functions are described later in the manual.

Sidebar — The **Sidebar** provides the user with quick access when working with multiple programs, multiple controllers, and multiple connection interfaces. This window is described in more detail in APOSS Sidebar.

Edit Window — The **Edit Window** displays the current APOSS program being edited and allows it to be changed. Many functions are provided through the mouse and through keyboard shortcuts in order to assist the user in the editing process. These are described below.

Placing the mouse on a command or function keyword will cause a small popup window to be displayed showing the syntax of that command or function. An option in the **Settings → Editor** dialog can be used to disable these popup windows. Pressing **F1** when the mouse is on a command or function keyword will cause the online help page for that command or function to be displayed.

Different colors are used to distinguish between comments, program sections, operators, numbers, etc. You can alter the colors using **Settings → Editor**.

Watch Window — The **Watch Window** is useful during debugging. It allows the user to "watch" various aspects of both the controller and the executing program while the controller is operating. See **Development → Watch Add** for a description of how to use the Watch Window.

Communication Window — The **Communication Window** displays both messages from the APOSS-IDE (e.g. compiler messages) and messages from the controller (e.g. programmed PRINT commands). Messages from the controller will be prefixed by the controller ID number (e.g. "[01]" in the above example).

The Communication window can be edited. This allows users to add comments, copy to/from the Windows clipboard, and search for strings.

After a program has been compiled, double-clicking on an error message displayed in the Communication Window will cause the edit cursor to be moved to the line containing the error. If the error is in an include file, then that file will be opened.

Findbar

The **Findbar** displays the results of string searches when searching in multiple files. This window is described in more detail in APOSS Findbar.

Status Bar

The undermost bar shows left the stauts, e.g. "Ready" and on the right side the line numbers as well as the cursorposition.

Within your program you can use the **line numbers** for orientation purposes. For example, the "Syntax Check" command not only places the cursor at the first line corresponding to an error, but also displays the line numbers containing errors in the Communication Window.

The current line number can be found in the status bar at the bottom right of the APOSS Window as is shown below. For example "13:1" means that the cursor is located on line 13 at position 1 (i.e. before the first character).

Keyboard and Mouse Functions

Function Keys

Frequently used functions are assigned to function keys:

F1	Access on-line help
F2	Skip to the next bookmark
F3	Find next (if "Find" has been used)
F4	Check the syntax of the program
F5	Execute the program (see also **Execute**)
F9	Single-step through the program (debug mode only)
F11	Access on-line help for system process information
F12	Open the Command List for simplified programming

The function keys will be described later in this chapter where appropriate.

Esc key

In standard Windows applications, the **Esc** key normally closes the currently active window. However, in the APOSS Window, the **Esc** key will automatically open a connection to the default controller if no controller has yet been connected. If a controller has already been connected, then the **Esc** key will abort any program currently executing on the controller.

!!!

If an executing program is aborted while the drive is rotating, then the drive will slow down with the maximum allowed deceleration. Stopping an operating drive in this way may cause damage to the system to which the drive is connected.

To help avoid the possibility of this happening as a result of the user accidentally pressing the **Esc** key, an option can be set in the **Settings → Options** dialog to disable the abort function. If set, then the user is still able to abort an executing program by pressing **Shift-Esc** (which is less likely to be pressed by accident).

Reconnect a Lost Connection

When an active connection to a controller is lost (for example, if the controller is powered off or the communication line is disconnected), then the APOSS Window connected to that controller will display a "Lost connection" message in its title bar. However, the window will remain "associated" with that controller. Pressing **Esc** (or executing any command that needs to talk to the controller) will cause the APOSS Window to attempt to reconnect to the same

controller. This behavior becomes relevant only when there are multiple controllers available on the communication line (e.g. CAN bus).

In certain cases, it is beneficial to have APOSS attempt to automatically reconnect to a lost controller. For example, this can be useful when an unattended executing program is generating debugging information. An option can be set in the **Settings → Options** dialog to enable automatic reconnection.

Mouse functions

The Edit Window supports the following functions using the mouse buttons.

L-click	Change the cursor position and clear any current text selection.
R-click	Display the popup edit menu.
L-double click	Select the word under cursor.
L-down and drag	Select text.
Alt-L-down and drag	Select a column of text.
L-down on selection and drag	Move the selected text.
Ctrl-L-down on selection and drag	Copy the selected text.
L-click in left margin	Select the entire line.
L-down in left margin and drag	Select multiple lines.
Spin wheel	Scroll window vertically.
Single-click wheel	Select the word under cursor.
Double-click wheel	Select the entire line.
L-down on splitter and drag	Split the window into multiple views or adjust the current splitter view.
L-double click on splitter 	Split the window in half or un-split the window if already split.

Popup Menus

Popup menus are provided at certain program locations if the right mouse button is clicked. For example, a right click in the Edit Window will display the Edit popup menu. When popup menus are available, they will be described in the following sections.

The popup menus are closed automatically when the selected function is executed or if any other location on the screen is clicked with the left mouse button.

Keyboard functions

The Edit Window supports the following functions using keyboard shortcuts. Note that many of these functions are only available using keyboard shortcuts

!!! Please note that some keyboard functions are vendor dependent and may not perform the desired function. If this is the case, please ask your system administrator.

Go to	
Home	Go to start of line.
End	Go to end of line.
Ctrl-Home	Go to start of program.
Ctrl-End	Go to end of program.
Ctrl-←	Go to start of word.
Ctrl-→	Go to end of word.
Ctrl-Alt-→	Go to start of next blank line.
Ctrl-Alt-←	Go to end of previous blank line.
Ctrl-G	Go to line (opens dialog).

Ctrl-B	Go to matching brace ("{([" or "})]").
F2	Go to next bookmark.
Shift-F2	Go to previous bookmark.
Ctrl-F2	Toggle bookmark on current line.
Text Selection	
Shift-←	Extend selection left.
Shift-→	Extend selection right.
Shift-↑	Extend selection up.
Shift-↓	Extend selection down.
Ctrl-Shift-←	Extend selection to start of word.
Ctrl-Shift-→	Extend selection to end of word.
Shift-Home	Extend selection to start of line.
Shift-End	Extend selection to end of line.
Shift-PageUp	Extend selection up one page.
Shift-PageDown	Extend selection down one page.
Ctrl-Shift-Home	Extend selection to start of program.
Ctrl-Shift-End	Extend selection to end of program.
Ctrl-Alt-F8	Select line under cursor.
Cut / Copy / Paste	
Ctrl-C	Copy selection to clipboard.
Ctrl-Insert	Copy selection to clipboard.
Shift-Delete	Cut selection to clipboard.
Ctrl-X	Cut selection to clipboard.
Ctrl-Y	Cut line to clipboard.
Ctrl-V	Paste from clipboard.
Shift-Insert	Paste from clipboard.
Ctrl-Alt-K	Cut all lines from previous blank line to next blank line to clipboard.
Find / Replace	
Alt-F3	Find.
Ctrl-F	Find.
F3	Find next.
Shift-F3	Find previous
Ctrl-F3	Find next word under cursor.
Ctrl-Shift-F3	Find previous word under cursor.
Ctrl-R	Find and replace.
Modify	
Insert	Toggle typing between "insert" and "overwrite".
Ctrl-Z	Undo last change.
Alt-Backspace	Undo last change.
Tab (with selection)	Indent selected lines.
Shift-Tab	Un-indent selected lines.
Ctrl-Backspace	Delete to start of word.
Ctrl-Delete	Delete to end of word.
Ctrl-U	Make selection lowercase.
Ctrl-Shift-U	Make selection uppercase.
Ctrl-Shift-N	Insert a new line above the current line.

Ctrl-Alt-R	Repeat the next command multiple times (opens dialog).
Ctrl-Shift-R	Start recording a macro.
Scroll Window	
Ctrl-↑	Scroll window up.
Ctrl-↓	Scroll window down.
Ctrl-PageUp	Scroll window left.
Ctrl-PageDown	Scroll window right.

Undo Function

You can use **Alt-Backspace**, or **Ctrl-Z**, or **Edit → Undo** to undo the last action in the **Edit Window**.

!!! Note that both **File → Save** and **File → Save as** clear the Undo memory.

Tabs

Use tabs and indentation to visually structure your program. The tab size can be set in **Settings → Editor**.

Recording and Executing Macros

Frequently used commands or command chains can be recorded as "macros" and then assigned to keyboard shortcuts. The shortcuts can then be used to repeat the commands whenever required. Up to 10 different macros can be defined. These macros are saved and restored the next time that the APOS-IDE is started.

To start recording a macro, press **Ctrl-Shift-R**. The following "Record Macro" dialog will indicate that the recording has started. Then type whatever you want recorded in the macro. This can include normal keyboard keys, keyboard shortcuts, and menu commands.

To end the recording, click on the black button in the "Record Macro" dialog. This will display the dialog shown below. Press the keyboard shortcut that you want to assign to the macro (e.g. Ctrl-Shift-M). Each shortcut can be up to two characters in length. Then select one of the available "Save As" alternatives.

!!! When specifying the macro shortcut, almost any keyboard key or shortcut can be used. However, be careful not to use a keyboard key or shortcut that already has an existing meaning. Doing so will cause the original meaning to be lost and in some cases cause unpredictable results. For example, using the ↑ key will make it impossible to use that key anymore to move the cursor! Using the "Alt" key (e.g. Alt-A, Alt-X, etc.) is a good option since most of these will not conflict with previously defined shortcuts.

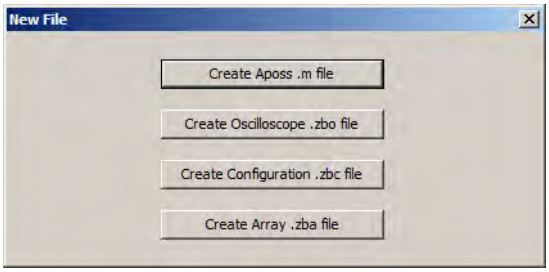
All currently defined macros can be cleared using **Edit → Clear Macros**.

File Menu

The **File** menu contains all the commands necessary to create, open, save, and print programs. **File** → **Open**, **Close**, **Save**, **Save as**, **Print**, and **Print setup** are used in the usual way.

New

To create a new program, click on **File** → **New** in the menu bar or click on the toolbar icon. Then click **Create Aposs .m file** in the subsequent dialog:



An APOSS Window is opened with the name "Untitled" and you can begin to write your program.

Sample

APOSS contains several sample programs. Any of these sample programs may be freely used, modified, or incorporated into other programs by the user. **File** → **Sample** can be used to edit any of these sample programs. Please see **Program Samples** for a list of the available sample programs.

Save and Save as

APOSS program files always have the extension ".m".

A "**Save All Source**" toolbar button will save all currently open program files. It is no longer necessary to save files one at a time.

Exit Program

The APOSS application can be ended by clicking on **Exit Program** or on the icon in the far upper right corner of the window. If you have not yet saved a new file, or saved changes to an old file, then you will have the chance to do this.

!!! **Exit Program** does not end a program running in the controller. You can only abort or end a program with **Esc** or with **Development** → **Break**. In order to do this, the file which is linked with the controller must be open or re-opened.

!!! Disconnecting from very early versions of controllers (for example, as a result of **File** → **Exit Program**), may cause the controller to stop executing if the controller is using PRINT commands to send messages to the PC. This happens once the controller's internal print buffer becomes full. This is not a problem for all newer controllers. Newer controllers will continue to execute even after the print buffer becomes full; print messages will simple be discarded once the buffer is full.

Edit Menu

The **Edit** menu offers the necessary editing help for programming. Most of these commands can also be reached via certain keys and key combinations. **Edit** → **Undo**, **Cut**, **Copy**, and **Paste** are used in the usual way.

Paste Array Assignment

This can be used to automatically paste array assignment statements into an APOSS program based on array value tokens on the Windows clipboard. For example, if the array name is set to "list", the start index set to "1", and the clipboard contains the 4 tokens "1 3 5 7", then the following will be pasted into the program:

```
list[1] = 1
list[2] = 3
list[3] = 5
list[4] = 7
```

Note that the start index can be any string. For example, it could be set to "offset+1" which would produce the indices [offset+1], [offset+2], [offset+3], etc.

The value tokens on the clipboard can be copied from any other Windows application using the normal "copy" mechanism. For example, a column of values could be copied from a spreadsheet application.

Find and Replace

Click on **Edit** → **Find** or press **Ctrl-F** and enter the search string into the following dialog field. Use **F3** to jump from one instance of the string to the next.

Click on → **Mark All** instead of → **Find** and all instances found are immediately "bookmarked" with a blue triangle in the left margin. **F2** can then be used to jump to successive bookmarks.

The regular expression implementation in search and replace functionality handles the following syntax:

Wildcards	? for any single character . for any single character + for one or more or something * for zero or more of something
Sets of characters	Characters enclosed in square brackets will be treated as an option set. Character ranges may be specified with a – (e.g. [a-c]).
Logical OR	Sub expressions may be OR-ed together with the | pipe symbol.
Parenthesized sub expressions	A regular expression may be enclosed within parentheses and will be treated as a unit.
Escape characters	Sequences such as \t, etc. will be substituted for an equivalent single character. \\ represents the backslash.

Examples:

10	Search for the string "10".
10+	Search for "1" followed by at least one "0" (e.g. 10, 100, 1000, etc.).
10*	Search for "1" followed by zero or more "0" characters (e.g. 1, 10, 100, 1000, etc.).
vel[xyz]	Search for "velx", "vely", or "velz".
vel[1-3]	Search for "vel1", "vel2", or "vel3".
(vel)|(acc)	Search for "vel" or "acc".
vel[\t]*=	Search for "vel", followed by any number of spaces or tabs, followed by "=" (i.e. search for assignments to the variable "vel").

Find in Files The Find in Files function will display a dialog allowing the user to search for strings within files. The user can choose to search either files that are currently open in APOSS or files that are on disk. All occurrences of the string which are found will be displayed in the **APOSS Findbar**. If the **Findbar** is not currently open, then it will be opened. Double-clicking on any occurrence in the list will open that file and position the cursor on that line.

The following options are available:

Find: Enter the string to be found. This can contain wildcard characters if "Regular expressions" are being used (see below). The dropdown box to select previously used search strings.

Match case: This option specifies whether a "case-sensitive" or "case-insensitive" search is done.

Match word: When this option is enabled, the search will only match complete "words" (i.e. the character before and the character after the search string must be a "non-word" character such as punctuation or spaces). This is most useful when searching for variables; it prevents matches from being found when the string is just a substring within another longer string.

Use regular expressions: Enabling this option allows wildcards to be used in the search string. The search string will be interpreted as a standard "Regular expression" (see below) .

Match limit: The search will be aborted after this many string matches have been found. This prevents excessive numbers of matches from being found which can make the Findbar unusable.

Where: Select which files are to be searched. If "Directory" is selected, then enter (or browse to) the directory to be searched. Note that this field also supports "drag and drop" file and directory names. If a filename is dropped, then the directory containing the file is searched.

File filter: List the types of files to be searched. This is available only if a "Directory" search has been selected. Two wildcard characters are available:

* will match any number of characters and

? will match any single character.

Multiple file patterns can be separated by space characters. For example, "*.m *.mi" will search all files with file extensions of either ".m" or ".mi".

Include subdirectories: If enabled, then subdirectories are also searched. This is available only if a "Directory" search has been selected.

Regular Expressions The following standard "Regular expression" wildcards are supported:

.	Match any single character.
?	Match the preceding item (character or expression) zero or one time.
*	Match the preceding item (character or expression) zero or more times.
+	Match the preceding item (character or expression) one or more times.
^	Match the beginning of the line.
$	Match the end of the line.
\	The next character is a literal (i.e. not a wildcard).
[abc]	Match any character listed in the brackets. This can also be one or more character ranges such as "[a-z]" or "[A-Za-z]", etc. If the first character is ^, then it will match any character NOT in the list. If the ^ character itself is to be matched, then it must not be placed first in the list. If the] character is to be matched, then it must be placed first in the list. If the - character is to be matched, then it must be placed last in the list.
|	Match the expression on the left or right.

	( )	Used as expression delimiters if necessary.
	@	(Non-standard) The following characters are case-sensitive.
	~	(Non-standard) The following characters are not case-sensitive.
	!	(Non-standard) The exclamation character is reserved and must not be used as the first character in the regular expression. If needed as the first character, then it must be preceded by the literal character (i.e. \!).
Examples	a.b	Match any sequence of characters that contains an “a” and “b” separated by exactly one character (e.g. “axb”, “a&b”, etc.).
	a\.b	Match any sequence of characters that contains the string “a.b”.
	ab*c	Match any sequence that starts with “a”, contains any number of “b” characters, and ends with “c” (e.g. “ac”, “abc”, “abbc”, etc.).
	^abc.*xyz$	Match all lines that begin with “abc” and end with “xyz”. Any number of characters may be between these strings.
	a[0-9]+	Match any sequence that starts with “a” and is followed by one or more digits (e.g. “a1”, “a999”, etc.).
	abc|xyz	Match any sequence containing “abc” or “xyz”.
	(abc|xyz)[0-9]+	Match any sequence containing “abc” or “xyz” followed by one or more digits (e.g. “abc1”, “xyz99”, etc.). Note that "abc|xyz[0-9]+" would match “abc” and it would match “xyz” followed by digits, but it would not match “abc” followed by digits.

Bookmarks Bookmarks allow the user to flag particular lines that are of interest to him and then quickly jump between them.

If bookmarks have been used in the editor, then they are saved and restored along with the program file.

Next Bookmark F2 If bookmarks have been used in the editor, then this function or **F2** will scroll the program and position the edit cursor on the next line that contains a bookmark.

Previous Bookmark If bookmarks have been used in the editor, then this function or **Shift+F2** will scroll the program and position the edit cursor on the previous line that contains a bookmark.

Toggle Bookmark This function or **Ctrl+F2** will toggle the bookmark on the current line. If the line contains no bookmark, then a bookmark is placed on the line. If the line already contains a bookmark, then the bookmark is removed from the line.

Clear All Bookmarks Click on **Edit → Clear All Bookmarks** to clear all existing bookmarks from the editor.

Convert Tabs Clicking on this menu item will replace tab characters in the file currently being edited, with space characters.

The tab spacing set in the **Settings → Editor** dialog is used.

Clear Macros This menu item will clear all editor macros defined using the **Ctrl-Shift-R** keyboard shortcut. Please see Recording and Executing Macros for a description of editor macros.

Keyboard Clicking this menu item will display a dialog allowing characters from other languages to be inserted into the program.

Development Menu

This menu provides various functions used during the program development phase of applications. This includes functions to compile, execute, break, and debug programs. It also includes functions that allow controllers to be connected and disconnected.

Many of these functions require a controller to be connected before the function can be executed. A controller can be connected either by pressing **Esc** to connect to the default controller or by using **Development → Select Controller** to connect to a specific controller if more than one controller is present. If no controller has yet been connected when a function is executed, then most of these functions will automatically connect to the default controller.

For information on debugging programs, please see Debugging Programs at the end of this chapter.

Execute F5

This will execute the program currently being edited. This involves the following steps:

1. If a controller is not currently open and connected to the APOSS Window, then an attempt is made to connect to the currently defined default controller. If no controller is connected, then the **Execute** is aborted.
2. A check is made to see if the controller is already executing a program. If so, then the user is asked if the existing program can be stopped. The controller can only execute one program at a time. If the controller is not "idle", then the **Execute** is aborted.
3. If the user has made changes to the program being edited, then the changes are automatically saved to the PC disk. If this is a "new" program, then the user is requested to enter a filename for the program. Note that it is not necessary to enter a filename at this time. If no filename is given, then a temporary file is used.
4. The program being edited is then "compiled". This produces a "machine code" version of the program that is customized for that particular controller. Note that this "machine code" is not human-readable and cannot be edited. If the compile fails for any reason, then the **Execute** is aborted.
5. The "machine code" (i.e. not the original text file being edited) is then downloaded to the controller and placed in the controller's temporary memory. Note that this temporary memory will be lost if the controller is powered off. To save a program permanently in the controller, use **Controller → Programs**.
6. After the download completes, the controller will start to execute the program.

Each subsequent time that **Execute** is used, the previously downloaded program will be overwritten with the newly downloaded program. This allows you to quickly and easily make changes and re-test during debugging.

Run Programs in Several Controllers

You can execute different programs on different controllers all at the same time simply by opening each program in a different APOSS Window. Then use **Development → Select Controller** in each window to select the desired controller for that program. Finally, use **Development → Execute** in each window to download and start the programs.
Note that you must use **Select** to open the controllers before using **Execute** since **Execute** will always connect to the default controller if no controller is yet connected.

Each APOSS Window can be connected to only one controller at a time and each APOSS program can be edited by only one APOSS Window at a time. Hence, if you want to execute the same program on several controllers, then use one of the following methods:

– Use **Development → Select Controller** to select and connect to the first controller. Then use **Development → Execute** to download and start the program executing on the controller. Once the program is executing, then use **Development → Select Controller** a second time to select and connect to the second controller. This will disconnect from the first controller but leave the program executing. Then use **Development → Execute** to download and start the program on the second controller. Repeat the process for as many controllers as desired.

– If the program is to be started on several controllers that all share the same network connection (for example, several controllers on a CAN network), then **Development → Download all** can be used. This allows the program to be compiled, downloaded, saved in permanent controller memory, and started, all in a single user operation.

Break [Esc] and Continue

Click on **Development → Break** or press **Esc** in order to stop any program currently executing on the controller.

!!!

If an executing program is aborted while the drive is rotating, then the drive will slow down with the maximum allowed deceleration.

Development → Break all can be used to stop the execution of multiple controllers at the same time.

Click on **Development → Continue**, in order to continue a program which was just aborted. In doing so any positioning processes which were interrupted will be completed.

If a program with an error message was aborted, you can **Continue** it again with this function once you have removed the error and/or erased the error message.

Messages -> Log file

This function can be used to start the logging of all messages displayed in the Communication Window to a file. Similarly, **Stop logging** will stop the logging of messages.

Messages are cached and may not always be written to the file immediately. As a result, if the file is edited or copied while logging is in progress, then the most recent messages will not always be included. To avoid this, use the **Update Log file** command immediately before editing or copying the file. This will force all messages cached up until this point to be written to the file.

Note that **Stop logging** and **Update Log file** will be enabled only if logging has been started.

Debugging Commands

The following commands are designed to be used to assist the user in debugging newly developed programs. For detailed information about debugging and the use of these commands, please see Debugging Programs.

Start Debugger

This command will prepare both the APOSS-IDE and the controller for debugging. This includes compiling the program in “debug” mode, inserting breakpoints, and downloading the program.

Go to Breakpoint

This command will start execution of the program at the current program line and continue executing until the next “user breakpoint” is reached.

Singlestep

This command will execute a single program line and stop at the next line. The next line is not executed.

Stop Debugger

This command removes the APOSS-IDE and the controller from “debug” mode.

Remove Breakpoints

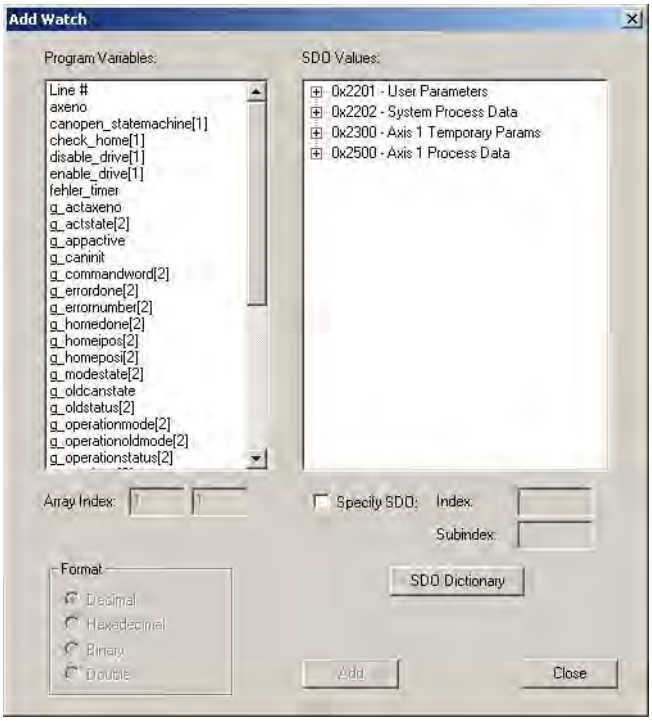
This command removes all “user breakpoints”.

Watch Add / Start / Stop

This function enables online monitoring of the variables, arrays, system and axis processing data and axis parameters.

All values currently being watched are maintained in the **Watch Window**. New values can be added to this list using **Development → Watch Add**. Existing values can be removed from the list by clicking on the value and then pressing the **Delete** keyboard key. They can also be removed by right-clicking and using the popup menu.

Values can be written to the controller (when a controller is connected) by right-clicking on an item in the list and selecting **→ Set Value** from the popup menu. Monitoring does not need to be active for values to be written. “**Set Value**” will only be enabled if a controller is currently connected.



The Watch Add dialog is shown above. All program variables and arrays are shown on the left and all system values are shown (in a tree) on the right. Add a value to the Watch Window by clicking on it, selecting the format to be used to display the value, and then pressing the **Add** button. Values can also be added simply by double-clicking on the value. Multiple values can be added before closing the dialog.

If an array value is being added to the Watch Window, then the array indices must be specified. These fields will be disabled if the selected value is not an array. Note that only the first 250 elements of an array can be watched.

Activate and stop the monitoring with → **Watch Start,** → **Watch Stop** or click  .

!!! When monitoring is active, the Watch Window is updated constantly. This consumes both controller resources and network connection resources. Hence, the number of values being watched should be limited to a reasonable number (usually, not more than 10-15 values, depending on network connection speed). If more values than this are required, then the **Oscilloscope** should be used.

Syntax Check F4 Clicking on **Development** → **Syntax Check** will do a "test compile" of the program being edited. This can be a useful function to use while creating new programs or modifying existing programs. It is a quick way to find syntax errors in the program without having to download the program to the controller and execute it. If a syntax error is found, then the line number and an error description are displayed in the Communications Window and the cursor is automatically placed at the position of the error.

The **Syntax Check** produces a debug file in addition to checking the syntax. This file will have that same name as the program but with the file extension ".ad$".

Compile to file The **Execute [F5]** command will compile a program into a "machine-readable" binary version which is then downloaded to the controller and executed. The **Compile to file** command is similar except that the compiled binary version is not downloaded and executed. Instead, the binary version is saved in a ".bin" file on the PC hard drive. Binary ".bin" files can be managed using **Controller** → **Programs**.

When **Compile to file** is selected, a dialog will be displayed allowing the user to specify the exact controller hardware for which the program is to be compiled. Note that compiled binary versions are hardware-dependent and must be compiled for the hardware on which they will be executed. After this, a **Save As** dialog will be displayed allowing the user to select the file name to be used to save the file. The file name will default to the name of the program with ".bin" as a file extension. Only after this is the program compiled and saved on disk.

!!! This feature is available only if **Binary file support** in **Settings → Options** is enabled.

Break all If programs are executing in several controllers, then click on **Development → Break all** to abort all the executing programs. Note that this will only abort the programs running on controllers that share the same connection interface as the controller connected to this APOSS Window (e.g. multiple controllers on a CAN bus). Controllers connected using other interfaces are not affected. For example, if this APOSS Window is connected using CAN-LPT, then any controller connected using USB will continue to execute any previously executing program.

Also note that programs will be aborted even on those controllers that are not currently connected to an APOSS Window. As long as the controller is using the same connection interface and is within the original connection ID scan range, then execution is stopped. For example, if controllers 1 and 2 are both on a CAN network but APOSS is currently only connected to controller 1, then **Break all** will still abort programs running on both controllers 1 and 2.

!!! If an executing program is aborted while the drive is rotating, then the drive will slow down with the maximum allowed deceleration.

Start all The **Start all** menu will send an 'Execute' command to all connected controllers that share the same connection interface as the controller connected to this APOSS Window (e.g. multiple controllers on a CAN bus). If you are testing software that runs on multiple controllers in a network, then this is a quick way to start all controllers at the same time.

Download all **Development → Download all** displays the **APOSS Download Mode** dialog. This allows the .m file currently being edited to be downloaded to multiple controllers.

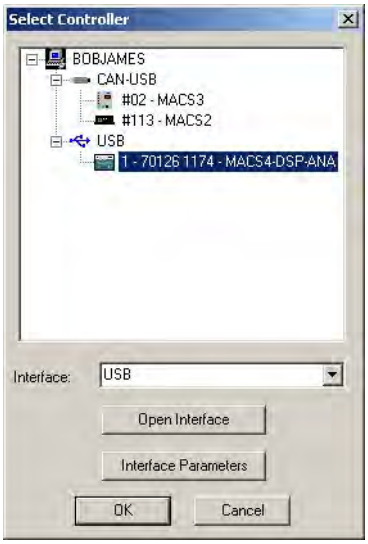
Please read **Download → Programs** for more details and options.

Delete Programs all **Development → Delete Programs all** will delete all programs saved on all controllers that share the same connection interface as the controller connected to this APOSS Window (e.g. multiple controllers on a CAN bus). Controllers connected using other interfaces are not affected.

Also note that programs will be deleted even on those controllers that are not currently connected to an APOSS Window. As long as the controller is using the same connection interface and is within the original connection ID scan range, then programs will be deleted. For example, if controllers 1 and 2 are both on a CAN network but APOSS is currently only connected to controller 1, then **Delete Programs all** will still delete programs on both controller 1 and 2.

This command is intended to be used in conjunction with **Development → Download all**.

Select Controller If more than one controller is accessible from the PC, then **Development → Select controller** can be used to select the specific controller to which the APOSS Window is to be connected. All currently available controllers will be displayed in a tree view. Select the desired controller and click on **OK** to connect to that controller.



If no controllers are shown or the desired interface is not present, then select the desired interface from the Interface dropdown box and click on → **Open Interface**. The desired interface can then be selected. Note that choosing an interface in this way does not change the default interface specified using **Settings** → **Interface**.

If a controller is already connected to the APOSS Window, then **Development** → **Select controller** can be used to switch the APOSS Window to another controller. Simply use the Select Controller dialog to select the new controller. The existing connection to the previous controller will then be closed and a connection to the new controller will be opened.

Since any specific controller can be connected to only one APOSS Window at a time, selecting a controller that is already connected to another APOSS Window will cause that controller to be switched from the other APOSS Window to this APOSS Window.

Close Interface / Close All Interfaces

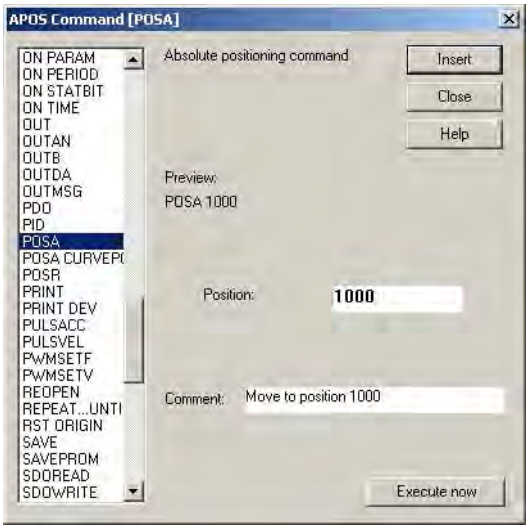
If there is a controller currently connected in this APOSS Window, then → **Close Interface** can be used to close the connections to all controllers that share the same connection interface as this controller (e.g. all the controllers on a CAN bus). A message will be displayed indicating which controller connections will be closed.

Close All Interfaces is similar to the **Close Interface** function except that all controllers on all connection interfaces are closed.

Command List

The **Command List** offers a list all commands with their syntax, allowing them to be either inserted into the existing program or executed directly. This can be useful for those more infrequently used commands that may not be so well remembered by the user.

You can find detailed information on all commands simply by selecting the command and clicking on **Help** or pressing **F1**.



For example POSA: Enter the position for the axis in the field. The preview shows you the exact syntax of the command.

Insert or Execute now

Move the text cursor (e.g. by clicking with the mouse) to the position in the Edit Window where you want to insert one or more new commands. Then click on **Development → Command List** and select the desired command (e.g. POSA). Enter any necessary parameter values and an optional comment and then click on → **Insert**. The completed command will then be inserted into the APOSS program at that position.

Click on → **Execute now** to test this command before inserting it into your program.

!!! Depending on the command, **Execute now** may cause drives to start up, stop, etc.

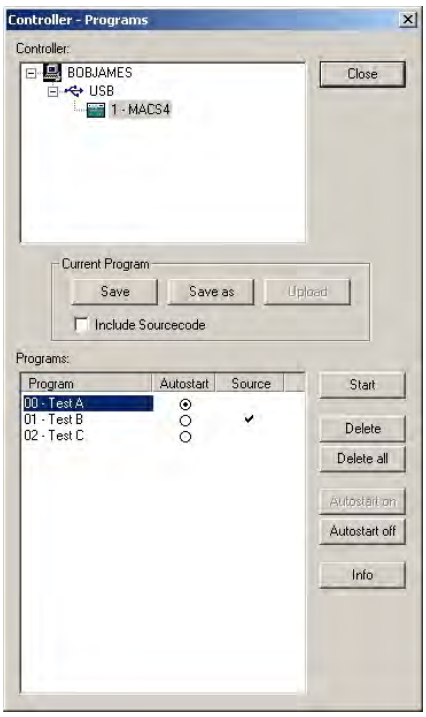
Controller Menu

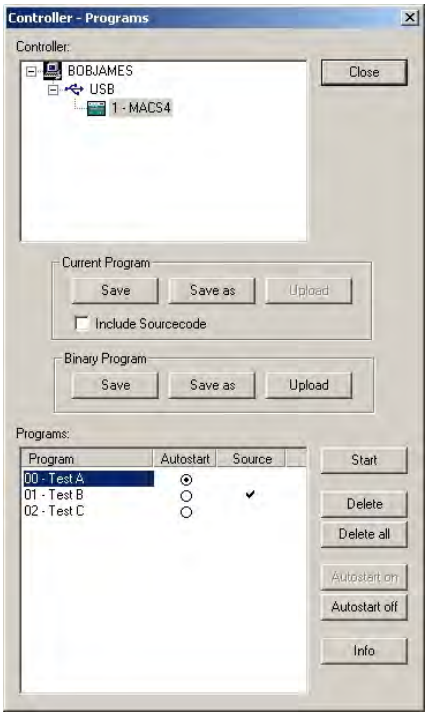
This menu provides various functions used to view and set the controller configuration and to manage various aspects of the controller's onboard memory. This includes functions to manage saved programs (and mark programs for **Autostart**), view and set global and axis parameters, and save and reset the controller EPROM memory.

Programs

Click on **Controller** → **Programs** to see all the programs currently stored on the controller in permanent memory. You can also select other currently connected controllers to see (and manage) the programs on those controllers.

!!! If **Binary file support** in **Settings** → **Options** is enabled, then the window shown on the right is displayed. Otherwise, the window on the left is displayed.





Under **Programs** is a list of all programs currently saved on the controller. Each program will be identified by a program number and a program name. Depending on available memory and the size of the saved programs, up to 91 programs may be saved at any one time. If a program has been designated as the **Autostart** program, then a ⦿ will appear in the Autostart column. If source code for a program has been included on the controller, then a checkmark will appear in the Source column.

Save a Current Program

Whenever you execute a program (i.e. with **Development** → **Execute**), it is compiled and downloaded into a temporary block of memory in the controller. This is overwritten with each subsequent **Execute** and it is lost if the controller is powered off. However, using **Controller** → **Programs**, you can permanently → **Save** the current program in the APOSS Window. The program is compiled and downloaded into a permanent block of memory in the controller. The new program will be appended to the end of the list of existing programs.

!!! Note that saving a program on the controller does not automatically save the original source code for the program. You should always use the normal **File** → **Save** menu command to save the program source to the PC hard drive.

Click on → **Save** and enter a name in the subsequent dialog field or confirm the file name suggested. The program number will be assigned automatically.

For controllers with firmware version 6.01.10 or later, names are saved using UTF-8 encoding. This allows any character to be used in names, even characters in foreign languages. However, names may not be longer than 8 bytes (n.b. some UTF-8 characters require more than one byte). For controllers older than 6.01.10, characters are restricted to ANSI characters

in the range 0x20 to 0x5F (n.b. accented characters may not be used) and the name cannot be longer than 8 characters.

Click on → **Save as** and you can also assign the program number (0 to 90) yourself, in addition to the program name.

Include Sourcecode

When this checkbox is enabled, the original source code for the program being saved will also be downloaded and saved on the controller. This source code can later be uploaded back from the controller and edited again on the PC.

When program source code is downloaded, then **Include** files used by the source code are also downloaded along with the source code. This allows a complete program, rather than only a partial program, to be stored on the controller.

All programs that have saved source code associated with them will have a checkmark in the Source column.

!!! If insufficient memory is available on the controller for saving the source code, then a message is displayed and other programs must be erased before saving the new program.

Upload a Current Program

All programs with a checkmark in the Source column have the associated source code for them saved on the controller. This source code can be uploaded back from the controller to the PC for subsequent use.

Select the desired program and click on **Current Program** → **Upload**. The default program name will contain the date and time when the original source code was saved on the controller. This is done to help avoid potential problems caused by overwriting existing files on the PC.

Save or Upload a Binary Program

These functions are displayed only if the **Binary file support** checkbox in the **Settings** → **Options** dialog is checked.

The **Save** and **Save as** buttons will display a dialog allowing a compiled .bin file to be selected. Input a program name and number and then download and save the binary file on the controller.

The **Upload** button will read the currently selected binary program from the controller.

!!! Controllers older than version 6.1.15 do not support the uploading of binary files. In this case, the **Upload** button will be disabled.

Start Programs

Using the Programs window, you can start any saved program directly. Simply select the program to be started and click on → **Start**.

Autostart

In order to support the use of the controller in "stand-alone" or "turnkey" applications (i.e. without user input), the controller can be configured so that an APOSS program is started automatically when the controller is powered on. Typically, in these types of applications, the controller must also never be "stopped". So the controller can also be configured to restart an APOSS program in the event that the executing program terminates.

The first part of this mechanism is the "Autostart program". The Autostart program (if one is defined) is normally always started automatically when the controller is powered on. If no Autostart program has been defined, then no program is started when the controller is powered on. Note that when the controller is powered on, it will run various self-test checks. If any of these fail, then the controller will not attempt to start the Autostart program.

The second part of the autostart mechanism is the "Restart program". With some exceptions, the Restart program is started automatically when the currently executing program terminates. In particular, this includes the termination of the Autostart program.

!!! If the Autostart program, Restart program, or any other program is aborted by the user, then the autostart mechanism will be deactivated. The controller will not automatically start the Autostart program or any Restart program again until the controller is powered off and on.

!!! Note that a program designated as an Autostart program can always be manually started by the user. However, in this case, the program will execute as a normal program, not as an Autostart program, and no Restart program will be started when the program terminates.

Autostart program

The controller determines the autostart program as follows:

1. If a program has been explicitly marked as the "Autostart program" (see below), then this program will be started when the controller is powered on.
2. Otherwise, if I_PRGSTART (103) is set to a digital input pin, then I_PRGCHOICE (104) will determine the program number of the Autostart program. The controller will wait until the I_PRGSTART input pin is activated and the Autostart program will then be started.
   Note that the pin may already be activated when the controller is powered on.
   These parameters are designed to be used when the Autostart program is being selected by some external source (such as a PLC).
3. Otherwise, there is no Autostart program.

Any saved program can be designated as the Autostart program by selecting it and clicking → **Autostart on**. The Autostart program will be marked with a ⊙ in the Autostart column. The autostart designation can be cleared again simply by selecting the program and clicking → **Autostart off**. Only one program at a time can be designated as the Autostart program. So if a second program is chosen as the Autostart program, then the Autostart designation is cleared from the previous program.

One of the main purposes of an Autostart program is to select which of multiple programs is to be executed in applications that require different programs in different situations. The Autostart program will identify (usually based either on configuration parameters or on digital input settings) which program is to be executed. The Autostart program then designates that program as the Restart program and terminates. The controller will then automatically start the designated program.

!!! The I_PRGSTART mechanism does not function properly in some controllers with firmware versions prior to 6.7.19. If this affects the application, then please contact zub machine control AG for updated firmware.

Restart program

If the controller executes an Autostart program when it is powered on, then the controller's autostart mechanism will be activated. When the autostart mechanism is active, then the controller will automatically start the designated Restart program whenever the currently executing program terminates. This will continue until the autostart mechanism is deactivated.

The autostart mechanism will be deactivated if any of the following happen:

1. The user has executed a "break" (i.e. ESC) command.
2. A program has terminated with an error (other than one of the errors listed below).

Once deactivated, the autostart mechanism cannot be reactivated again with powering the controller off and on again.

The autostart mechanism will not be deactivated if any of the following errors occur:

1. Position error (error 8)
2. Limit switch error (error 25)
3. Software limit switch error (error 11)

The controller determines which program is the Restart program as follows:

1. If PRGPAR (102) is set (i.e. not -1), then this is the program number of the Restart program.
2. Otherwise, if I_PRGSTART (103) is set to a digital input pin, then the controller will wait until that input pin is activated. Once activated, then I_PRGCHOICE (104) will determine the program number of the Restart program.
   Note that the pin may not already be activated when the currently executing program terminates; activation is triggered only on a rising signal.
3. Otherwise, if the Autostart program is the only program to have been executed, then the Autostart program will be the Restart program (i.e. the Autostart program will be executed repeatedly).
4. Otherwise, there is no Restart program and the controller will not autostart any program.

The PRGPAR parameter is designed to be used for "linking" programs together. The currently executing program simply needs to set PRGPAR to specify which program is to be executed next. Note that if PRGPAR is not reset, then the current program will be executed repeatedly.

The I_PRGSTART and I_PRGCHOICE parameters are designed to be used when the next program to be executed is being selected by some external source (such as a PLC).

!!! Please note that the originally designated Autostart program is restarted only if PRGPAR and I_PRGSTART have not been used. If either PRGPAR or I_PRGSTART have been used, then the Autostart program will only be executed once when the controller is powered on. This "one-time" execution can be useful for executing functions such as HOME. Note that the PRGPAR parameter can still be used to explicitly restart the Autostart program.

!!! The I_PRGSTART mechanism does not function properly in some controllers with firmware versions prior to 6.7.19. If this affects the application, then please contact zub machine control AG for updated firmware.

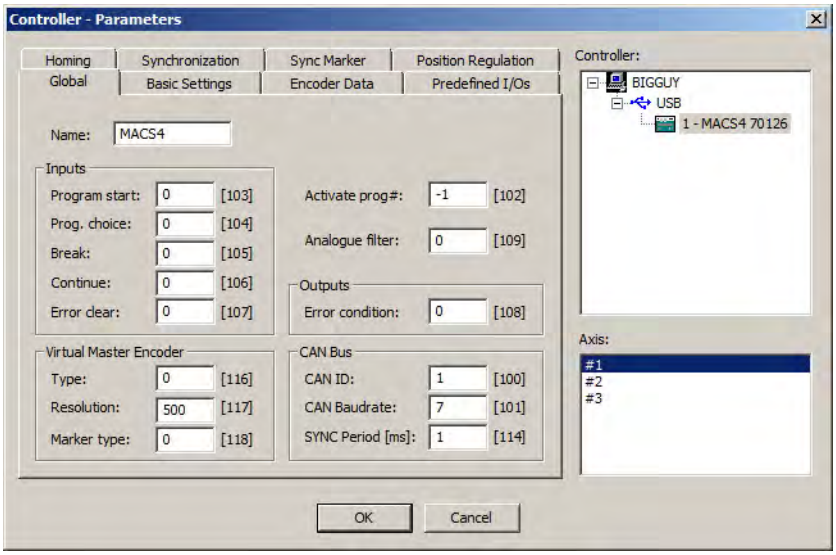
Delete programs Select any program and click on → **Delete** if you want to delete that individual program from the controller. This permanently deletes the program from the controller so you should ensure that you have previously saved the program on the PC.

Click on → **Delete all** if you want to delete all the programs in the controller. This permanently deletes the programs from the controller so you should ensure that you have previously saved the programs on the PC.

Info Select any program and click on → **Info** to view the compiler options used when compiling the program. Note that this function is not available on some older controllers.

Parameters The controller maintains global parameters and axis parameters. Global parameters are valid for the entire controller. Axis parameters can have different values for each axis.

Parameters > Edit When parameters are being edited, then a dialog similar to the following will be displayed. A tab will be displayed for each of the various groups of parameters.



Different axes and different controllers can be selected. When selected, then the parameters for that axis and controller will be displayed and can be modified. Note that no changes are written to any controller until the "OK" button is pressed. When the "OK" button is pressed, then the modified parameters will be written for all controllers.

In all of the parameter tabs, the internal parameter number is listed after the name of each parameter. To find detailed information, parameter values, and initialization values for any parameter, please use this number and refer to chapter Parameter Reference. Or simply press **F1** when the text cursor is in one of the input fields and information about the corresponding parameter will be displayed.

!!! The global parameter "*Name*" allows you to assign a name to a controller or to change the existing name of a controller. The name is displayed in various dialogs in the APOSS user interface, allowing you to more easily differentiate between multiple controllers.

For controllers with firmware version 6.01.10 or later, names are saved using UTF-8 encoding. This allows any character to be used in names, even characters in foreign languages. However, names may not be longer than 8 bytes (n.b. some UTF-8 characters require more than one byte). For controllers older than 6.01.10, characters are restricted to ANSI characters in the range 0x20 to 0x5F (n.b. accented characters may not be used) and the name cannot be longer than 8 characters.

Parameters > Save to file

For backup purposes, the entire controller configuration (i.e. global parameters, axis parameters, user parameters, and arrays) can be uploaded and saved as a file on the PC. This file can then be downloaded again to the controller at a later time if it becomes necessary to restore the configuration.

Select the controller, click on → **Save**, and then enter a name in the subsequent "Save As" dialog. If the parameters of multiple controllers are to be saved, then simply select another controller and → **Save** again.

Parameters > Restore from file

If a controller's configuration has been backed up as a file on the PC, then the configuration can be restored to the controller again by downloading the configuration file.

Click on **Parameters → Restore from file**. This will display an "Open" dialog. Select the file containing the data to be loaded. In the subsequent dialog, select the controller into which the data should be loaded. Use the checkboxes to specify which sets of parameters are to be loaded. Then click the → **Restore** button. The specified parameters are then loaded into the controller. Any previous parameter values are overwritten.

If the same data is to be loaded into more than one controller, then simply select another controller and click → **Restore from file** again.

Parameters > Save ALL to files

If there are multiple controllers on a single connection interface (e.g. a CAN network), then this function will allow you to simultaneously save the configuration information from all controllers on the network into files. Click on **Parameters → Save ALL to files**. In the subsequent "Save As" dialog, specify the "base" name of the files that you want to save the configuration into. One configuration file will be created for each controller. The files will be named "base-id.zbc" where "base" is the name specified above, "id" is the controller ID number (e.g. CAN ID), and "zbc" is the file extension. All files will have "zbc" as the file extension.

Memory

Save RAM

The → **Save RAM** function saves all programs, parameters, and arrays from RAM into EPROM. This corresponds to the SAVEPROM command. This function is usually needed only to save arrays if necessary since programs and parameters are automatically saved. Any data that have not been saved from RAM into EPROM will be lost when the controller is powered off.

!!!

Some very old versions of controllers did not automatically save programs in EPROM. For these controllers, use this function to save the programs into EPROM.

Delete EPROM

The → **Delete EPROM** function will reset all parameters to their initialization values and delete all arrays and programs. The controller is reset to the original factory setting. However, this is done only after the controller has been turned off and on again.

!!! Remember the following when you delete the EPROM:

1. Check whether you have saved all the necessary APOSS programs on the computer. These will need to be reloaded into the controller again if necessary after the controller has been turned back on.
2. Check whether you have saved the controller configuration into a backup file on the computer (using Parameters → Save to file). With this file, you can reload the previous parameters and arrays if necessary.
3. Click on **Memory → Delete EPROM**.
4. Re-load the necessary configuration parameters and APOSS programs in the controller.

Reset

Parameter

The → **Parameter** function will reset all global and axis parameters in the controller to their initialization values.

Arrays

The → **Arrays** function will delete all arrays in RAM. This function has the same effect as the DELETE ARRAYS command.

!!! Note that if **Memory → Save RAM** is subsequently used, or an APOSS program executes the SAVE ARRAYS command, then the arrays in the EPROM are also deleted!

Complete

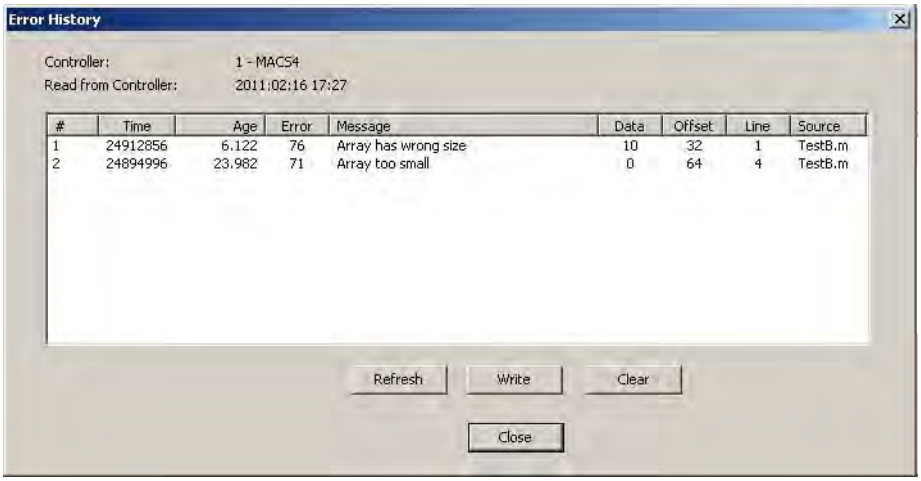
The → **Complete** function will reset all parameters and delete all arrays and programs. The controller is reset to the original factory setting.

!!! **Reset → Complete** happens immediately. This differs from **Memory → Delete EPROM** which happens only after the controller has been turned off and on again.

Error History

Click on **Controller → Error History** to see a history of all the errors that have occurred on the controller.

!!! Some older versions of controller do not support this function. The function will be disabled in this case.



Errors are listed from most recent at the top of the list to oldest at the bottom of the list. The list is cleared each time that the controller is powered on. The list holds a maximum of 50 errors; the oldest error will be discarded when a new error is added to the list.

Double-clicking on an error in the list will close the Error History dialog and highlight this line in the Edit Window.

!!! **Warning:** The APOSS-IDE cannot verify that the program currently being edited matches the program executing in the controller at the time the error occurred. This is the user's responsibility. If the programs do not match, then the wrong line will be highlighted in the Edit Window.

The following information is listed:

Time Controller system time when error occurred. See "sys_clock" in System Process Data.

Age Age (in seconds) of the error when the Error History dialog was opened (i.e. how long ago the error occurred). This will be updated each time the dialog is refreshed.

Error Error number. See chapter "Error Reference and Messages".

Message Error message text.

Data Optional value saved with error. The meaning of this value depends on the error.

Offset Binary offset within the compiled program where the error occurred.

Line Line number within the program being edited where the error occurred.

Source Name of the program being edited where the error occurred.

The following functions are available:

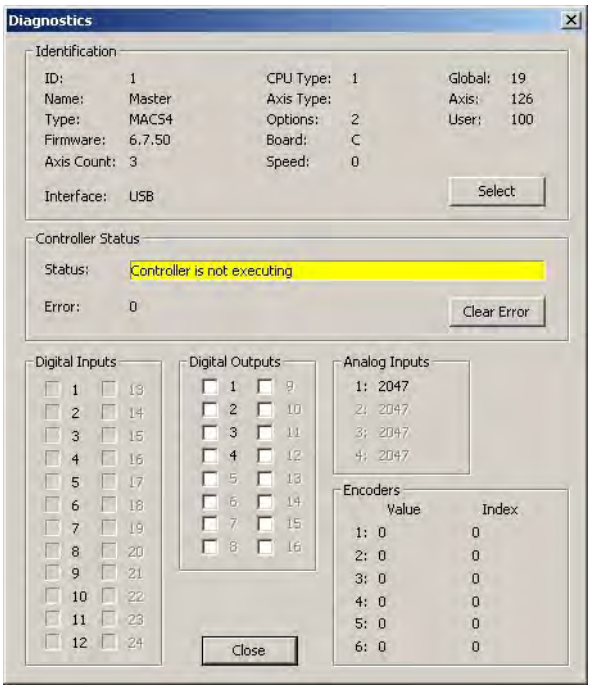
Refresh Update the list with the current error history from the controller. Note that this will update the "age" of the existing errors in the history.

Write Write the error history into an ASCII text file so that the errors can be analyzed at a later time. This file can be edited by any text editor or imported into a spreadsheet application.

Clear Clear the error history.

Diagnostic Report Click on **Controller** → **Diagnostic Report** to create a text file containing the entire current state of the controller. This file is often required by zub machine control AG support personnel when assistance is required to help diagnose problems.

Diagnostics Click on **Controller** → Diagnostics to display a window showing diagnostic information from the connected controller.



The labels in the "Digital Inputs", "Digital Outputs", and "Analog Inputs" boxes will be grayed for inputs and outputs that do not correspond to physical pins on the controller.

!!! Note that in some cases, digital outputs that do not have a physical pin, can still be used as a "virtual" digital output.

Upload Firmware Click on **Controller → Upload Firmware** to upload the firmware currently running on the controller. The firmware can then be saved in a ".bin" file. A controller must be connected before this function is enabled. The function is only available for controllers with firmware version 6.07.68 or later.

Tools Menu The Tools menu offers the following tools:

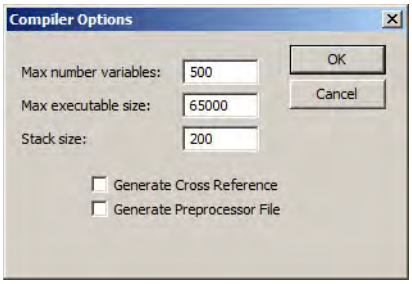
CAM-Editor – see corresponding section in chapter „APOSS Tools"

Array Editor – see corresponding section in chapter „APOSS Tools"

Oscilloscope – see corresponding section in chapter „APOSS Tools"

Settings Menu This menu offers various options and settings.

Compiler The default values for the compiler options are set appropriately for most applications.



Max number variables The Maximum number of variables has a direct effect on the amount of memory available in the controller. The minimum required number of variables can be calculated on the basis of the application program in use:

Minimum number of memory for variables =
Number of global variables
+ 1 variable per DIM array (not depending on the array size)
+ 1 Variable per each (!) element of global arrays
+ Reserve for later enhancements (e.g. 50 variables)

Please read more information about the different types and usage of arrays and variables here: Variants Comparison: "Arrays, Variables: global, local?" in Chapter "Programming with APOSS" in section "Data handling"

An insufficient number of variables is detected during compilation of the program, i.e. before the program is downloaded into the control unit. A compilation error message is given then.

Max executable size The Maximum executable size specifies the maximum size, in bytes, of a program after it is compiled and ready for download to the controller.

Note that some older controllers do not support executable sizes larger than 65000 bytes.

Stack size Stack size specifies how much memory (= so-called stack) has to be reserved for the internal handling of function calls and their local data. If deeply nested function calls or a large number of local variables or local arrays are in use, it is necessary to increase the stack size. The required stack size can be calculated approximately on the basis of the program code and the logical program flow:

Minimum stack size =
Maximum sum of all local variables
of nested function calls which are active at the same time
+ maximum sum of all local array elements,
of nested function calls which are active at the same time
+ Reserve and overhead (e.g. 100 bytes)

Nested function calls are strongly responsible for the required size of stack size. An insufficient stack size due to nested function calls can just be detected during program run. In this case the next critical function call will not be executed anymore and an APOSS #93 error is reported. It is possible to react on such an error inside the defined error handling routine (ON ERROR … GOSUB) and ensure a controlled system behavior.

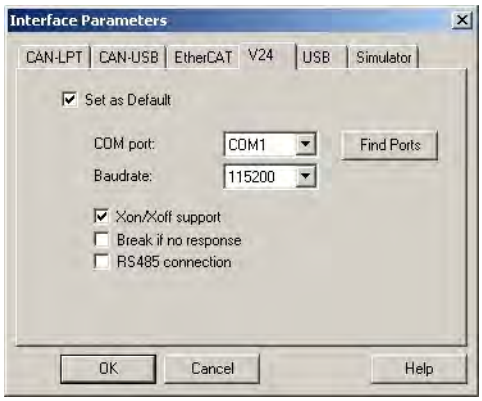
Generate Cross Reference Cross reference files are not necessary for the proper functioning of APOSS and the user may safely disable this option. These files are normally of interest only to zub machine control AG personnel when debugging APOSS compiler issues.

Generate Preprocessor File All APOSS program files are preprocessed (see Preprocessor in chapter "Programming with APOSS") before being compiled. This will make various string substitutions in the original program. If the user needs to see exactly what the compiler is compiling, then this option can be enabled. Enabling the option will cause APOSS to generate a ".txt" file with the same name

and in the same directory as the original file. This file will contain the output from the preprocessor. The file will be created each time that the user compiles the program.

Interface Parameters

Normally, connection interface parameters will have been set correctly when the APOSS-IDE is first installed and used (see chapter "Getting Started", Connecting to the controller). However, if they need to be changed for any reason (e.g. the baud rate or COM port has changed, the controller ID scan range needs to be extended, a new interface type has been added to the system, etc.), then **Settings → Interface** can be used to update the parameters. A dialog similar to the following will be displayed.



Simply select the appropriate interface type and change the necessary parameters.

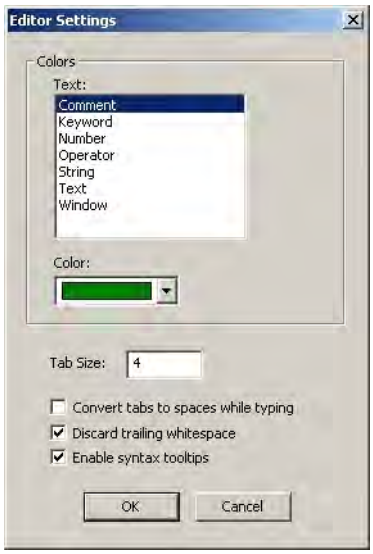
!!! Note that only one interface type can be selected as the default interface. The default interface is used to establish connections to controllers when the **Esc** key is pressed.

!!! Setting the interface parameters will affect only subsequent connections to controllers. Any controller to which there is already an established connection, will remain connected and will not be changed. If an established connection to a controller needs to be changed, then it will be necessary to close the connection to that controller, change the interface settings, and then re-connect to the controller.

Language

If you desire another language, click on **Settings → Language** and choose from the available languages in the subsequent dialog field. **→ Exit program** and start APOSS again. It is necessary to exit the program and then restart again before the language change will take effect.

Editor Settings



In order to provide greater clarity, different colors can be assigned to the various program elements such as comments, key words, numbers, etc. Tab settings can also be chosen to make the program more readable.

Colors	Select the type of program element (e.g. Comment) and select the desired color.
Tab Settings	Choose the preferred tab size. This can be set to 2 or more.
Convert Tabs	Click on → **Convert Tabs to spaces while typing** to convert all tab characters to space characters while typing. Note that this does not affect any tab characters that already exist in the program. Use **Edit → Convert Tabs** if you want to convert all existing tabs in a program to spaces.
Discard trailing whitespace	Click on → **Discard trailing whitespace** to **remove all trailing space and tab characters at the ends of lines before saving files.**
Enable syntax tooltips	Click on → **Enable syntax tooltips** to enable tooltips in the editor. These tooltips will display the syntax of the command when the mouse is paused on a line.
Options	This dialog allows you to select various options that control the behavior of APOSS. When settings are changed in the → **Options** dialog, the settings will take effect the next time that they are used.
Enable Shift-**Esc** break only	Normally in APOSS, the **Esc** key will send a "break" command to a connected controller to abort any running program. If the drive is rotating, then the drive will slow down with the maximum allowed deceleration. Stopping an operating drive in this way may cause damage to the system to which the drive is connected. To help avoid the possibility of this happening as a result of the user accidentally pressing the **Esc** key, this option can be set to disable the abort function. If set, then the user is still able to abort an executing program by pressing **Shift-Esc** (which is less likely to be pressed by accident).
Reopen previous files	If this option is enabled, then when APOSS starts, it will automatically try to reopen all the files that were open when APOSS was last used.
Reopen previous connections	If this option is enabled, then when APOSS is started and previously open files are reopened, then APOSS will automatically try to reconnect the files to controllers for those files that were connected when APOSS was last used.
Auto-reconnect lost connections	If this option is enabled, then APOSS will try to automatically reconnect to controllers when connections are lost. APOSS will attempt to reconnect once every two seconds for up to one minute.
Open windows maximized	If this option is enabled, then all windows opened in APOSS will be opened full-size. They can subsequently be reduced in size, if desired; they will simply be opened full-size. If this option is not enabled, then only the first window opened will be opened full-size. If a second window is opened, then all windows will be made smaller so that all windows can be seen simultaneously.
Autosave	Enabling this option will cause APOSS to automatically save the current file being edited to disk before compiling and executing it. If not enabled, then APOSS will use a temporary file for compiling the current file.
Binary file support	If enabled, then APOSS will allow the direct handling of binary compiled program images in the **Controller → Programs** window. This includes uploading compiled programs from the controller so that they can be saved on the PC and downloading previously saved binary images back to the controller. Note that these binary images cannot be edited; they can only be saved and restored. Also note that some older controllers do not support this feature. If enabled, this option will also cause the item **Compile to file** to be added to the **Development** menu (once APOSS has been re-started the next time). This item allows to user to manually compile and save the current program file to a binary file. See also BinFileMap in illustrations in chapter Technical Reference.
Print color	If this is enabled, then programs printed from the Edit Window will be printed in color using the same colors as the Edit Window. Otherwise, programs will be printed in black and white.

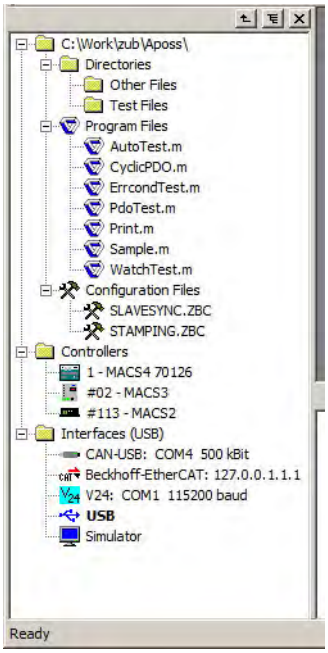
Array Editor parameter support	Normally, the Array Editor will read and display all user parameters and arrays. However, if this setting is enabled, then the Array Editor will also read and display global parameters, temporary and permanent axis parameters, axis process parameters, and system process parameters.
Create debug file	Enabling this option will cause APOSS to generate debug log files. This is normally useful only to product support personal. This option should normally be disabled.
Protocol font	This is the type font that is used to display messages in the Communications Window. Changing this value will only affect newly opened windows.
User Mode	The user mode allows APOSS to be tailored to the experience level of the user. Currently, this mostly applies to the **Oscilloscope** function only. It also affects how many SDO values are offered in the various SDO selection lists. Rarely used SDO values are removed from the lists for non-expert users.
Customized Sections	The **SDO Lookup Tables** allow more experienced users to customize the contents of the SDO selection lists used in APOSS. For example, these are used in the Watch Window and the **Oscilloscope**. This field will accept drag-and-drop filenames. The **State Machine Program** setting allows experienced users to use a customized state machine program in the **Tune Oscilloscope**. This field will accept drag-and-drop filenames. The **Tune Oscilloscope Monitor File** setting allows experienced users to use a customized test window in the **Tune Oscilloscope**. This field will accept drag-and-drop filenames. The **Debug File Directory** allows users to specify where debug log files are to be created. This field will accept drag-and-drop file or directory names. If a file is dragged to this field, then the directory will be set to the directory containing the file.
Oscilloscope	This dialog allows you to select various options that control the behavior of APOSS oscilloscope tool. Please see **Oscilloscope** for more detail.

Windows Menu

The commands on the **Window** menu follow the Windows standards (i.e. **Cascade** for overlapping windows arranged down and across, **Tile vertically** for vertical windows arranged beside each other, or **Tile horizontall**y for horizontal windows arranged above/below each other).

APOSS Sidebar

The Sidebar is shown below. It is intended to be used to provide quick access when multiple programs, multiple controllers, and multiple interfaces are being used.



If the window is not wanted, then it can be closed by clicking the close button at the top right of the window. It can be re-opened at any time by using the **Windows → Show Sidebar** menu command. The width of the window can be adjusted by clicking and dragging the right side of the window in a manner similar to how edit window splitter bars are used.

The window will display all files and directories in the current directory, as well as all currently connected controllers and all available connection interfaces. The currently selected default interface will be shown in parentheses on the Interface title line. If the interface list is expanded, then the default interface will be displayed with a bold font.

Double-clicking on items in the sidebar will have the following affect:

Directory The directory shown in the sidebar is changed to this directory.

File If this file is already open, then the Windows focus is set to this file. If the file is not open, then it will be opened. Note that double-clicking on a configuration file will open the CAM Editor, double-clicking on an oscilloscope file will open the Oscilloscope, etc.

Controller If this controller is currently associated with an Edit window, then the Windows focus will be set to that window. This allows the user to easily "find" the associated window when multiple windows are open. If the controller is not associated with an Edit window, then the Diagnostics dialog for the controller is displayed.

Interface If the window which currently has the focus is an Edit window, then this will automatically open a controller connection using the selected interface.

If the Edit window is already connected to a controller, then the existing controller connection will be closed and a new controller connection will be opened. Note that the selected interface does not need to be the default interface; any interface can be selected. This is a quick way to connect different

controllers to different Edit windows. A warning message is displayed if the current window is not an Edit window.

Right-clicking on an interface will display a popup menu allowing that interface to be selected as the default interface or the interface parameters for that interface to be set.

The sidebar contains the following buttons in the top-right corner:

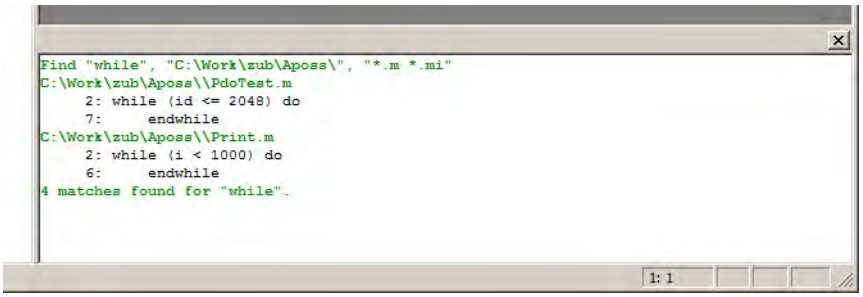
Up The directory shown in the sidebar is changed to the parent of the current directory.

Browse A dialog is displayed allowing the user to select the directory that will be displayed in the sidebar.

Close The sidebar window is closed. It can be re-opened again by using the **Windows → Show Sidebar** menu command.

APOSS Findbar

The **Findbar** is shown below and is used to display the results of the **Edit → Find in Files** menu command. It is always placed at the bottom of the main APOSS window.



If the window is not wanted, then it can be closed by clicking the close button at the top right of the window. It will be automatically re-opened the next time that **Edit → Find in Files** is used. It can also be re-opened at any time by using the **Windows → Show Findbar** menu command. The height of the window can be adjusted by clicking and dragging the top of the window in a manner similar to how edit window splitter bars are used.

Double-clicking on any of the lines in the **Findbar** will automatically open the corresponding file (if it is not already open) and place the text cursor on that line.

Help Menu

The → **Contents** function will start the online help subsystem and display the "Contents" tab.

The → **Index** function will start the online help subsystem and display the "Index" tab.

The → **SDO Dictionary** function will start the online help subsystem and immediately jump to the "SDO Object Dictionary" page.

The → **zub Website** function will start the default Internet browser and direct it to the zub home page.

The → **About Program** function will display the version numbers of the APOSS-IDE, the low level interface driver, and the compiler.

The look and feel of the Help subsystem will depend on the version of the Windows operating system being used.

Context-sensitive Help

The **Command list** and the parameter dialog fields in the **Controller** menu and **CAM-Editor**, offer direct access to help. Select a command in the **Command list** or one of the parameter input fields and press **F1**. Help for that specific item will be displayed.

Download Menu

The **Download** menu provides a simple interface to support the downloading of firmware and programs to multiple controllers. Note that all APOSS Windows, with the exception of the main APOSS-IDE application window, must be closed before this menu will be available.

Download Firmware

For all newer versions of controllers, the controller firmware (i.e. not compiled APOSS programs) can be updated directly from within APOSS.

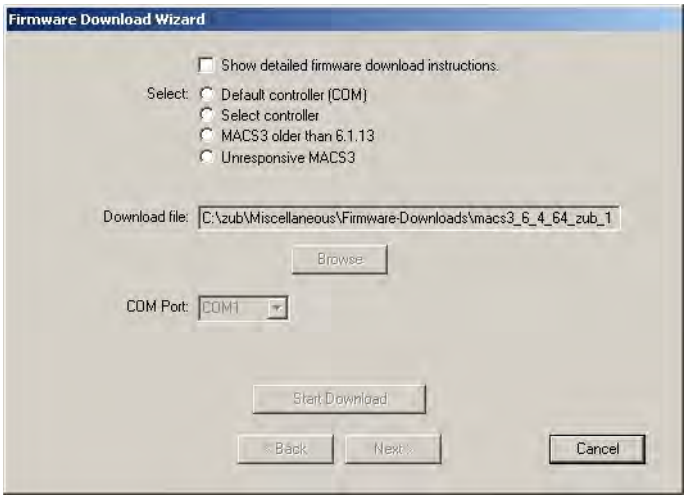
Updated firmware for different types of controllers may be downloaded from the zub machine control AG website (www.zub.ch) if available. Otherwise, please contact zub (info@zub.ch). zub will be happy to provide you with replacement firmware or optimized firmware specifically designed for your hardware and application. On request, zub will also assist in customizing products and will adapt special functions to customize firmware. Stay in contact with zub to get the most out of our products and your machines!

Using the Download Firmware Wizard

Make sure that all APOSS Windows, with the exception of the main APOSS-IDE application window, are closed. Then click on → **Download** and select → **Firmware**. You will be presented with the "Welcome to the Firmware Download Wizard" dialog shown below.

If this is the first time that you have downloaded firmware, or if you are unfamiliar with the firmware download process, then press **Next** and follow the instructions. These will guide you through the download process.

If you are familiar with the download process, then select the → **Firmware expert mode** checkbox and press **Next**. The entire download process can then be completed using a single dialog (shown below) and with a minimum of instructions.



If you are not an “expert” and the “expert dialog” is displayed, then you can switch back to the normal dialogs by selecting the → **Show detailed firmware download instructions** checkbox and pressing **Next**.

The filename of the last file downloaded by the firmware downloader is saved and will be used as the default, the next time the firmware downloader is used.

!!! Please read the hardware manual for more details and safety instructions.

Download Programs

Select → **Programs** and use the **Browse** button to select the APOSS program that should be downloaded. If the controller interface supports multiple controllers, then enter the range of controller ID’s to which the program should be downloaded.

Enter a program name to be used to identify the program on the controller.

For controllers with firmware version 6.01.10 or later, names are saved using UTF-8 encoding. This allows any character to be used in names, even characters in foreign languages. However, names may not be longer than 8 bytes (n.b. some UTF-8 characters require more than one byte). For controllers older than 6.01.10, characters are restricted to ANSI characters in the range 0x20 to 0x5F (n.b. accented characters may not be used) and the name cannot be longer than 8 characters.

Select the desired operations using the check boxes **Download, Verify**, **Set Autostart**, **Save in EPROM**, and/or **Restart**. For example, you might check **Verify** but not **Download**. In that case the Downloader will only verify that the specified file is the one that is already on the controller.

Start Download Programs

Click the **Start** button. A connection is established to the controllers using the default APOSS interface. Any controller that is currently executing will be stopped. The following steps are performed for each accessible controller. All steps are performed for one controller before going on to the next controller.

- The program is compiled using compiler settings appropriate to the controller.
- All existing programs on the controller are deleted.

- The program is downloaded.
- The program is saved as program number 0 with the specified program name. If no program name was specified, then the first 8 characters of the filename are used.
- If the Verify flag is set, then the downloaded program is uploaded again and the downloaded and uploaded versions are compared. If they differ, then processing is halted for this controller. Note that not all older controllers contain firmware that will support uploading the program. This flag is ignored if the controller does not support uploading.
- The Autostart flag is set for the program if requested.
- All data is saved in EPROM if requested. Note that this will be done automatically for newer controllers.
- The controller is restarted with the new program if requested.
- The connection to the controllers is closed.

If any error occurs during the processing of a controller, then processing is halted for that controller. However, processing will continue for subsequent controllers in the ID range. When the entire download operation is complete, a summary is displayed in the text box listing the ID's of those controllers that encountered no errors and those controllers that encountered errors. Use the scroll bars to scroll up in the text box in order to determine the nature of any problems encountered.

Save Log

The progress of the download is displayed in the large text box. If desired, this information can be saved to a text file by clicking the **Save Log** button.

If the download fails for one or more controllers, then you will get a "Failed for controllers" message. Saving the log will let you go back and review the failures at a later time.

Debugging Programs

The APOSS-IDE contains a powerful built-in debugger. This provides common debug features such as single-stepping, breakpoints, and the ability to read and set program variable values.

The debugger may be used when the controller is not currently executing a program or when the controller is already executing a program. If the controller is not currently executing, then starting the debugger will debug the source that the user is currently editing. If the controller is already executing, then the user must ensure that the source being edited matches the program that is currently executing.

!!! The debugger cannot be used in all cases. For example, it may not be possible to use the debugger with programs that are actively controlling a motor. Stopping program execution at a breakpoint is equivalent to pressing the **Esc** key to break program execution. This will cause the motor to slow down and stop with the maximum allowed deceleration. In many cases, stopping the motor like this will invalidate the test procedure and make the debug result meaningless. As well, if program execution is continued after a breakpoint, then it is unlikely that the motor can be restarted correctly to put the system into the state it was in prior to the breakpoint.

!!! It may also not be possible to use the debugger with programs that rely on ON PERIOD functions. The internal timer that triggers calls to ON PERIOD functions does not stop when program execution stops at a breakpoint. This may leave a pending interrupt which will then trigger an ON PERIOD call as soon as program execution is continued.

For situations like the above where the debugger cannot be used, the Oscilloscope provides excellent debugging capabilities. It can watch and record program variables and system states without having to break program execution. These can then be reviewed afterwards to identify problems. For more information on the Oscilloscope, please see APOSS Oscilloscope in chapter "APOSS Tools".

Starting the Debugger

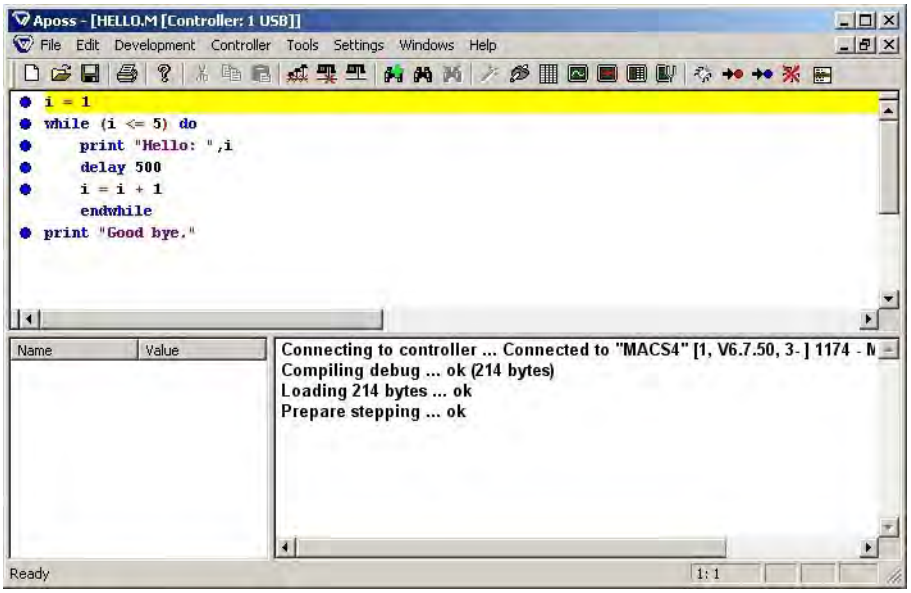
To start the debugger, edit the program to be debugged in the normal way so that it is displayed in the Edit Window. Then click on **Development → Start Debugger** or click  . If the controller is not currently executing a program, then this will take the following actions:

1. The program is compiled in debug mode and downloaded to the controller. However, program execution is not started at this time.
2. A blue dot is placed before each executable statement in the program. These are the positions where the user may insert breakpoints.
3. The next statement to be executed (i.e. when execution is started or continued) will be highlighted in yellow.

If the controller is already executing a program, then starting the debugger will take the following actions:

1. A blue dot is placed before each executable statement in the program. These are the positions where the user may insert breakpoints.
2. The executing program continues to execute.
3. The user may pause the program at any time by pressing the **Esc** key. The next statement to be executed (i.e. when execution is continued) will be highlighted in yellow.

The following diagram shows what the Edit Window might look like.



!!! While the debugger is active, the program should not be modified. Doing so will cause the APOSS-IDE to become out-of-sync with the version of the program executing on the controller and the debugger may no longer be able to properly follow the program execution. If the program must be changed, then the debugger should be stopped and the test restarted from the beginning.

Stopping the Debugger

To stop the debugger and end the debugging session, click on **Development → Stop Debugger** or click . This will do the following:

1. Remove the blue dots marking executable statements.
2. Remove any breakpoints that the user has set.

If the controller is executing a program when the debugger is stopped, then the controller will continue to execute the program. If the program has been stopped, then the user may restart the program at the point where it stopped, by clicking on **Development → Continue**.

Single-Stepping

To single-step through the program, use **Development → Singlestep**, press **F9,** or click . This will execute the next statement (i.e. the statement currently highlighted in yellow) and automatically stop before the next executable statement is executed (i.e. at the next statement with a blue dot). This next statement will then be highlighted in yellow.

While execution is stopped, the user is free to examine and modify the value of any program variable, set and clear breakpoints, etc.

At any time, program execution can be continued without single stepping, by using **Development → Go to Breakpoint** or by clicking .

Using Breakpoints

Breakpoints are set by double-clicking anywhere on the statement in the program (except on the blue dot) where the breakpoint is to be set. The dot will change from blue to red to indicate that the breakpoint has been set.

Double-clicking on a statement that already contains a breakpoint will clear the breakpoint and the dot will change back from red to blue.

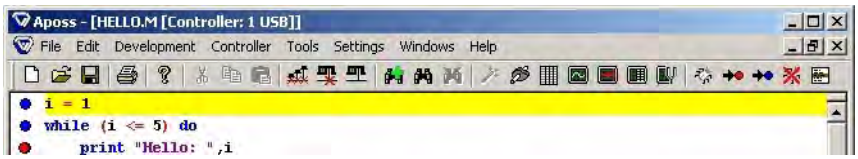
The user can set a "breakpoint" at any executable statement in the program. These are the statements with blue dots. When program execution encounters a breakpoint, execution is immediately stopped prior to executing the statement with the breakpoint. The statement will then be highlighted in yellow since this is the next statement to be executed.

While execution is stopped, the user is free to examine and modify the value of any program variable, set and clear breakpoints, etc.

When ready, the user can click on **Development → Go to Breakpoint** or on  to continue program execution. The program will then execute up until it encounters another breakpoint (i.e. up until the next statement with a red dot). Note that it is also possible to single-step after stopping at a breakpoint.

At any time during program execution, the user may also press the **Esc** key to stop execution. Execution will then stop immediately rather than continuing to the next breakpoint. The next statement to be executed will be highlighted in yellow. Again, the user can continue with either **Development → Go to Breakpoint** or **Development → Singlestep [F9]**.

The following diagram shows what the Edit Window might look like when breakpoints have been set before each "print" statement. Note the red dot on these statements.



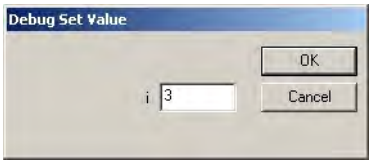
!!! A maximum of 10 breakpoints are allowed.

!!! Beginning with controller firmware version 5.22.00, breakpoints replace #DEBUG on/off commands. However, any existing #DEBUG commands need not be removed from the program; they will simply be ignored.

Displaying and Modifying Variables

At any time while the debugger is active (i.e. whether the program is executing or stopped) the current value of any program variable can be displayed. This is done by clicking on (or immediately after) any instance of the variable with the left mouse button. The current value will be displayed in a yellow popup box. The value will be displayed until the mouse is moved away from the variable. An example is shown below where the mouse button was clicked near the variable "i" in the "print" statement.

At any time while the debugger is active, the current value of any program variable can also be modified. This is done by clicking on (or immediately after) any instance of the variable with the right mouse button. The dialog shown below will be displayed allowing the user to modify the variable value.



Displaying the Executing Line

At any time, whether the debugger has been started or not, the current statement being executed (or about to be executed) by the controller can be displayed by pressing the  toolbar button. The statement will then be highlighted in yellow. This highlight can be cleared simply by clicking on the yellow highlight. This function is useful for determining which part of a program is executing at any given time.