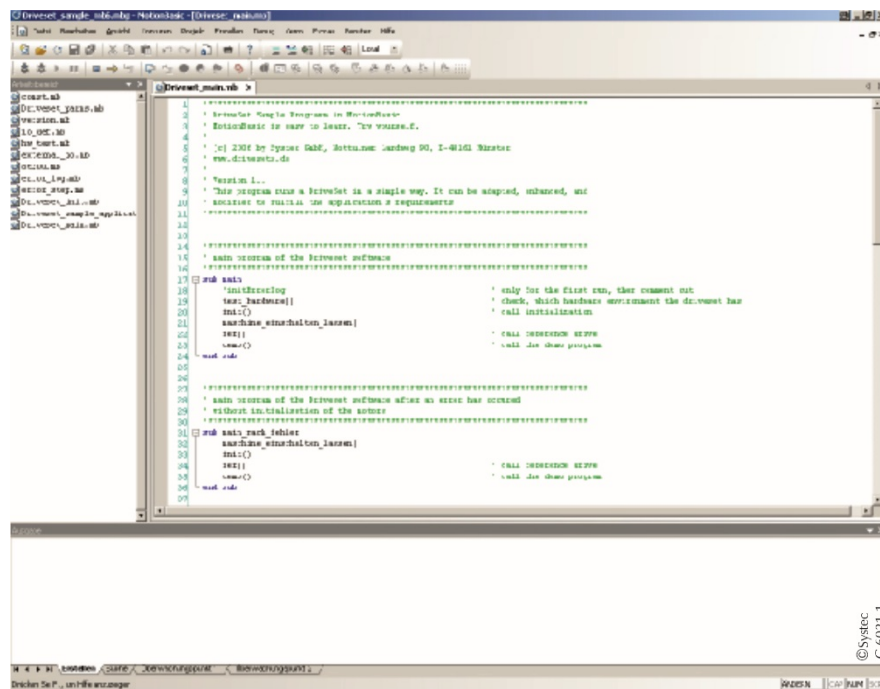


MotionBasic 6 IDE User Manual



Introduction to the professional handling of the MotionBasic IDE

Systemec Elektronik and Software GmbH

Nottulner Landweg 90
48161 Muenster-Roxel
Germany

Phone: +49 / 25 34 / 80 01-70

Phone: +49-700-SYSTEMEC-DE

Fax: +49 / 25 34 / 80 01-77

Internet: www.systemec.de

E-mail: info@systemec.de

MotionBasic 6 IDE User Manual

Doc status: 11 2017

DocuNo: 875-11-1.5

Last documented software version: 6.5.6

Translation of the original manual

Copyright and all other rights to this document remain with Systemec GmbH. Systemec does not take any responsibility for the correctness and/or completeness of the contents. We reserve the right to make technical changes.

You can download this document from the Systemec website free of charge. For this document, Systemec GmbH grants you the simple charge-free right, unlimited in space and time, for all known and not yet known types of use. All rights with respect to patent grants or industrial design registration and further rights remain unaffected.

You may duplicate this document. Distribution is only allowed with the clear indication of the copyright held by Systemec GmbH. You may not process, modify or change this document in any other way. In order to distribute this document for commercial reasons and to make it available, you will require prior written authorization from Systemec GmbH.

Table of contents

1	Overview of MotionBasic and the Xemo controller	7
2	Introduction to MotionBasic	9
3	Introduction to the IDE	12
3.1	Components of the integrated development environment IDE	12
3.2	The MotionBasic IDE operational modes	13
3.3	Offline programming.....	14
3.4	Online programming.....	16
4	Installation, starting up and ending.....	17
4.1	Installing MotionBasic 6	17
4.2	Starting MotionBasic 6	17
4.3	Ending MotionBasic 6	17
5	The MotionBasic 6 work area.....	18
5.1	The application window	18
5.2	Areas and functions of the application window.....	18
5.3	Editor functions.....	19
6	File organization in MotionBasic 6	21
6.1	Introduction.....	21
6.1.1	File organization principles in MotionBasic	21
6.1.2	Advantages of file organization.....	22
6.1.3	Project files	23
6.1.4	Program files	23
6.1.5	Administration of project and program files – schematic display	24
6.2	Generating and editing project files	26
6.3	Generating and editing program files	29
7	Compilation.....	30
7.1	Compilation of the MotionBasic program	30
7.2	Using the XCode.....	32
8	Connecting the PC with the controller.....	33
8.1	Setting the interface parameters.....	33
8.2	Program start and run	34
9	Error search with the IDE	37
9.1	Kinds of errors.....	37
9.2	Methods of error search	38
9.3	Monitored program execution	40
9.4	Monitoring variables	41
9.5	Breakpoints.....	42
10	Menus of the IDE.....	45
10.1	Overview	45
10.2	“File” menu	45
10.3	“Edit” menu	49
10.4	“View” menu.....	50
10.5	“Contours” menu.....	52
10.6	“Project” menu	59
10.7	“Build” menu.....	60

10.8	“Debug” menu	61
10.9	“Xemo” menu.....	62
10.10	“Tools” menu	65
10.11	“Windows” menu.....	68
10.12	“Help” menu	69
11	Online operation with Xemo!Go	71
11.1	Xemo!Go’s application window.....	71
11.2	Digital inputs and outputs.....	72
11.3	Axis settings and running commands	73
12	Appendix	75
12.1	Compatibility between MotionBasic 5 & MotionBasic 6	75
12.2	Bibliography	76
13	Index.....	77

About this manual

After a general introduction into the development environment this user manual of the MotionBasic IDE gives you an introduction to all work steps, which you do with the IDE: from installation, over file organisation to flashing programs in your Xemo controller.



Remark

You find the MotionBasic programming guide with quick start guide and detailed language reference in a separate document, [SYSTEC717].

Chapter 1 – Overview of MotionBasic and the Xemo controller
Outline of the programming language and the controller

Chapter 2 – Introduction to MotionBasic
Explanation of the basic elements of MotionBasic

Chapter 3 – Introduction to the IDE
Components and modes of operation of the development environment

Chapter 4 – Installation, starting up and ending
Installation, starting up and ending of MotionBasic 6

Chapter 5 – The MotionBasic 6 work area
Overview of the IDE work area

Chapter 6 – File organization in MotionBasic 6
Principle, creation and editing of project and program files

Chapter 7 – Compilation
Conditions, store or flash XCode in Xemo controller

Chapter 8 – Connection the PC with the controller
Interface parameter, program start and process

Chapter 9 – Error search with the IDE
Error types, methods for troubleshooting

Chapter 10 – Menus of the IDE
Description of the menus in the IDE

Chapter 11 – Online operation with Xemo!Go
Short description of the surface of Xemo!Go

Chapter 12 – Appendix
Compatibility to MotionBasic 5, bibliography

Important symbols in this manual

 **DANGER**

indicates a hazardous situation which, if not avoided, will result in death or serious injury.

 **WARNING**

indicates a hazardous situation which, if not avoided, could result in death or serious injury.

 **CAUTION**

indicates a hazardous situation which, if not avoided, may result in minor or moderate injury.

NOTICE

indicates a property damage message.



Identifies an instruction.



Remark

Please read passages, which are marked with this symbol, definitely. Get important information about dealing with these instructions and conditions or limits for the use of the MotionBasic IDE.



Tip

Learn additional facts and practical tips in sections, which are marked with this symbol.

[SYSTECxxx]

The literature abbreviation [SYSTECxxx] refers you to other manuals by Systec. See the bibliography in Chap. 12.2.

1 Overview of MotionBasic and the Xemo controller

In the normal controlling of machines, a distinction is made between two kinds of information processing: trajectory and switch functions. Moving an axis belongs to the trajectory functions, clamping a workpiece is a switch function. There are two systems for controlling both functions. The "CNC" control – Computer Numerical Control – is responsible for the trajectory functions, where as the PLC - Programmable logic controller (PLC) – manages the switch functions. Each control system has its own programming language; they communicate with one another via interfaces.

The Systemec Xemo controllers combine both control functionalities in one device. To make uniform programming possible, Systemec has developed the programming language "MotionBasic". BASIC, an easily learned programming language, provides the foundation for MotionBasic. Systemec expanded the BASIC language standard with commands for controlling both trajectory and switch functions.

The first step to take for working with MotionBasic is to install the IDE (Integrated Development Environment) on the PC (see Chap. 4.1). As soon as that has been installed, you can start to work with MotionBasic. The Xemo controller does NOT have to be connected for practise purposes and/or to write programs.

If the PC, the Xemo controller, and the machine are connected to one another, the system is basically ready to run. The Xemo controller permits two forms of operation.

With the one, individual MotionBasic commands can be sent direct to the controller. That is made possible by the user interface "Xemo!Go." This functionality is quite helpful for initial set-up or test purposes, as you do not have to write a complete program. For this kind of operation, the controller must be constantly connected to the PC; for that reason it is called "online operation."

Furthermore, you always work in online operation when transferring HPGL or ISO (G code) commands or programs by using the Xemo DLL.

In contrast to that, you can write a complete MotionBasic program with the IDE on the PC. You can also test it from there. After you have completed it, you can permanently store it in the Xemo controller's memory. Afterward, you can detach the connection to the PC. Your program will now run independent of the PC in the so-called "Offline operation."

How does the Xemo controller know which machine it needs to function with? Depending on the individual application, the number of axes, the trajectory velocity, the performance data of the motors, the accuracy of the positioning, etc. all vary.

All this data is defined as "system parameters." This is deposited in so-called registers. You can view these registers just like mail boxes. Each register


contains a system parameter and can be addressed via a number or a name. The velocity of an axis, for example, is stored in the register with the number 2000 and the name "_Speed." You need to determine the specific data of the machine before putting it into operation.

At first, all system parameters are preset with default values so that the controller is in a defined status after being switched on. These presettings are selected in such a way that the positioning control usually operates without or with only a few parameter settings. If the parameters are to have other values than the presettings, you need to modify them correspondingly in the MotionBasic program.

For example, add a subprocedure call-up at the beginning of your MotionBasic program and write the specific parameters of your application into this subprocedure.

Individual parameters can be defined online for the start-up procedure or for test purposes.

2 Introduction to MotionBasic

MotionBasic is a modern, structured programming language for the  multiaxial, positioning and trajectory controller 'Xemo'. MotionBasic combines the BASIC standard language with elements of the Programmable Logic Controller (PLC) and language rules for numerical controllers (DIN 66025, ISO-Code: G-Code for trajectory conditions)

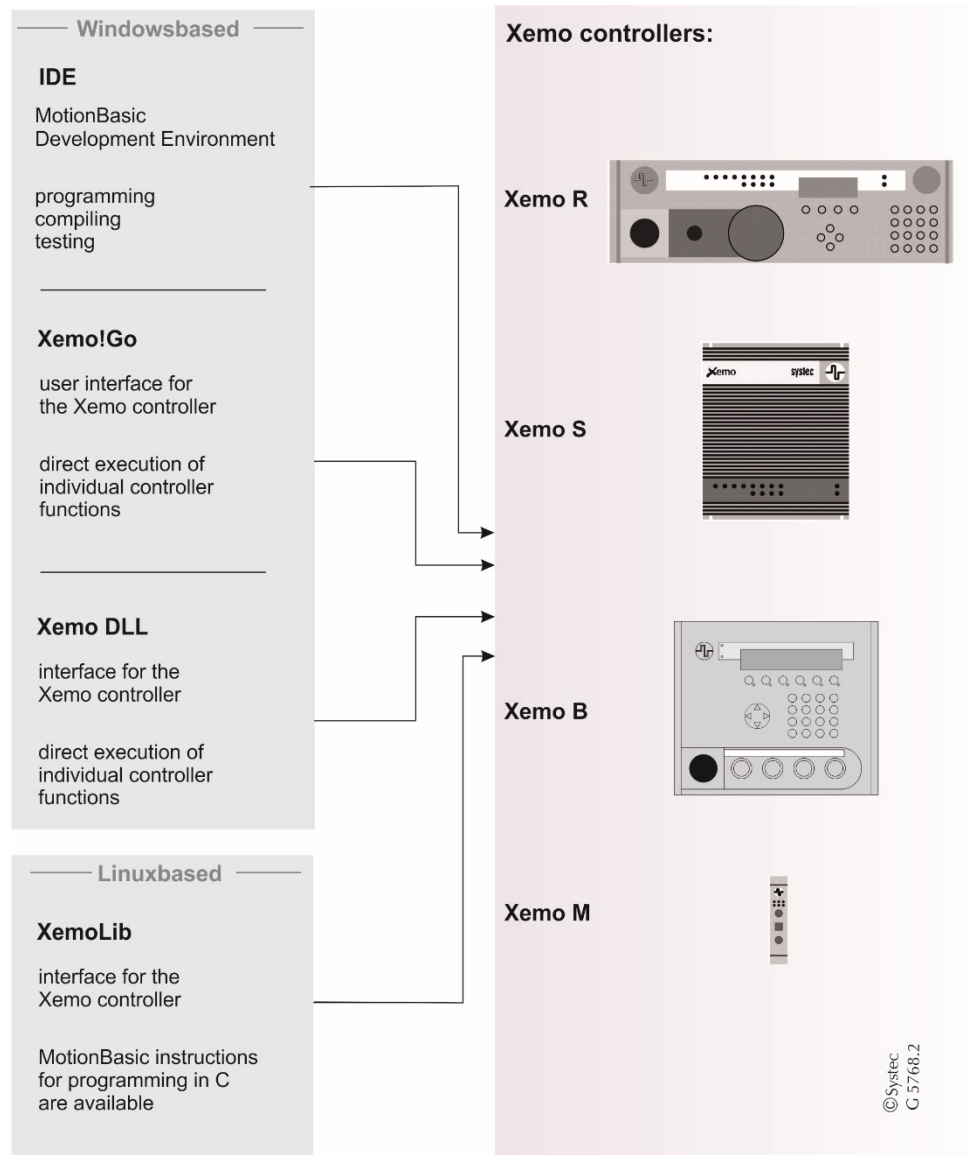


Fig. 1 Overview of different possibilities of addressing the Xemo controllers

MotionBasic is not only a programming language. Incorporated into MotionBasic are the Integrated Development Environment (can be run on a PC), the user interface Xemo!Go (online tool) for the Xemo controllers as well as the Xemo DLL with which programmers can integrate MotionBasic commands into their programs. The positioning and trajectory controller Xemo naturally belong to MotionBasic as well. Fig. 1 shows a selection of Systemec Xemo controllers.

Development environment	The Integrated Development Environment IDE contains all of the tools (program editor, compiler, debugger) necessary to write a program. With the IDE you write a program, store it in the controller and test it.
Xemo!Go	Online operation is made possible with the user interface Xemo!Go. Xemo!Go is a kind of software which accepts and interprets online commands and then sends them to the controller. Online programming with Xemo!Go simplifies and shortens the machine start-up. The axes as well as the inputs and outputs can be tested without your having to write an entire program. For such tests, some instructions are even available in the form of buttons; i.e. you don't even need to enter the commands. For the use of Xemo!Go, also see [SYSTEC775].
Online	The Xemo controller permits online and offline operation. Online operation means that the controller must be connected to the PC. Individual MotionBasic instructions are sent from the PC to the controller where they are immediately carried out. Online operation is made possible by the user interface Xemo!Go. Another possibility for online operation is through use of the Xemo DLL with the help of which a PC program created by a programmer can communicate with the Xemo, or through transfer of HPGL or ISO code. For the use of HPGL and ISO codes, see [SYSTEC772].
Offline	Offline-operation, by contrast, means that the controller is functional without connection to the PC. For offline operation you write a complete MotionBasic program on the PC. Upon completion, it is translated (compiled) into a machine language which controllers understand. Then you permanently store the program in the controller's "Flash" memory. The PC can now be separated from the controller ("offline"). The controller executes the programs on its own. This occurs e.g. after switching the controller on or after input via the control panel, depending on how the unit is programmed.
Combined online and offline operation	A possibility which is quite appropriate for testing is a kind of combined online and offline operation. Offline operation means that functional programs or subprocedures are stored in the controller's memory. In online operation, these programs can be called up from within Xemo!Go or another PC program by entering single orders. For this purpose, e.g. system characteristics can be modified. It is likewise possible to enter individual commands parallel to a running program.
Xemo-DLL, Library	For users who would prefer to program their applications on a PC in VisualBasic, VisualBasic for Applications (VBA), Visual C, LabVIEW, etc., MotionBasic provides the necessary tools. Via the Xemo DLL, MotionBasic instructions can be imbedded into these programming languages. The syntax of the library and DLL functions or procedures is described in the MotionBasic DLL manual. For use with Unix (Linux), a functions bibliography in source code (XemoLib) of the programming language "C" is available.

- Subprocedures** Subprocedures which have been stored in the controller can also be called up online. Parameters can be entered through the parameter register. Complex processes can be called up externally.
- Multitasking** MotionBasic capable of multitasking. Multiple program parts (tasks) can be processed at the same time. The number of simultaneously active tasks is unlimited. In this way, MotionBasic is given PLC functionality. All active tasks divide the CPU time evenly among themselves.
- The execution of online commands occurs in a separate task independent of internally running tasks. For that reason, it is possible to execute online commands at the same time as internally running programs and/or tasks.
- Data exchange** If the Xemo controller is, for example, a subordinate part of a larger application, it must communicate with superordinate controllers and/or calculators. Via a series of parameter registers, MotionBasic programs can exchange data with external devices. These registers can be accessed by MotionBasic programs as well as by external systems.

3 Introduction to the IDE

3.1 Components of the integrated development environment IDE

The integrated development environment (IDE) consists of a program editor, a compiler, and a debugger. With the IDE, you can write MotionBasic programs, load them into the motor controller and test them.

The integrated development environment (IDE) consists of the following components:

- Program editor** With the program editor, you write your MotionBasic programs.
- Compiler** The compiler translates your MotionBasic program into a machine language which the controller understands. Because of the fast compiler cycle (edit, compile, execute program), your development work is very effective. With just a single command, your program is automatically compiled and transferred to the controller: the program is then activated and halted at the section just edited. Now you can, for example, execute each individual program step.
- Debugger** Checking out a program and/or searching for errors in it are called debugging. For such debugging, the IDE provides a variety of tools. You can set breakpoints to interrupt the execution of the program at a particular place. You can run through critical sections of the program step by step. When doing that, you can efficiently exclude sections, subprograms and functions which have already been tested and verified. You can oversee any number of variables in two separate “surveillance” windows. Changes to variables through the course of the program become immediately visible.
- Simulating an operator terminal** If you use a controller with an operator terminal, you can also simulate the terminal functions on the PC. In this way, relevant programming can be tested and optimized without having to be connected to the controller.
- Xemo!Go** When initializing complex systems, it is always necessary to create/recreate a certain specific status for the machine. The integrated online tools help you do this. With our autonomous program – Xemo!Go – you can control the entire machine; e.g., position axes at specific locations or configure outputs. This facilitates and shortens the initialization process because the entire functionality of the machine does not have to be executed for each test. With this process, you can concentrate on those sections of the program or functions of the machine which are still not performing properly.

3.2 The MotionBasic IDE operational modes

MotionBasic 6 and Xemo!Go The IDE provides two operational modes: so-called online and offline operation.



Remark

You don't toggle between the modes offline and online mode with the operational mode switch of the Xemo controller.
For the functionality of the operational mode switch, see the respective user manual of the Xemo controller.

MotionBasic offline operational mode

Offline operation

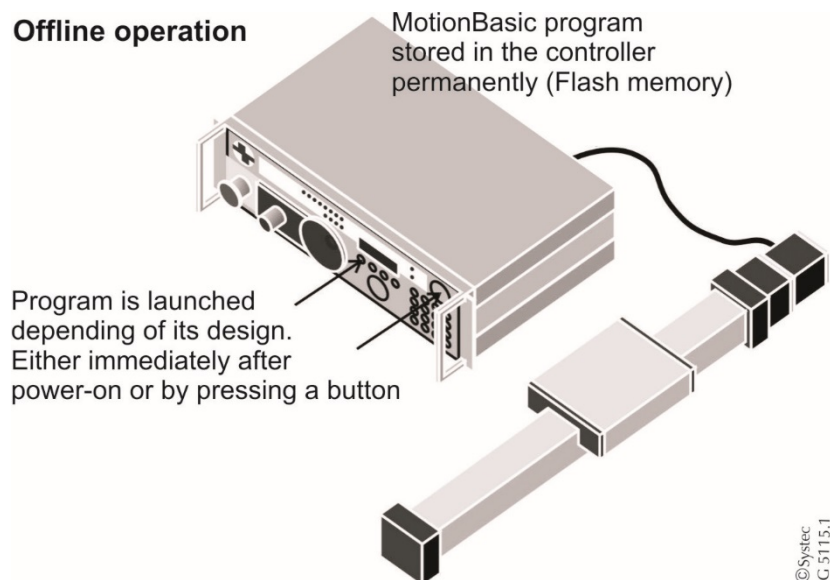


Fig. 2 Offline operational mode

“Offline operational mode” means that the generation of the program on the PC and the execution of the program in the controller are two completely separate processes. The first process, the generation of the program, must be concluded before the second process, the execution of the program, can succeed.

In the offline operating mode, a program is generated completely within the MotionBasic IDE on the PC. The completed program is then translated (compiled) into a language understood by the controller and transferred to the controller. There it is saved in either non-retentive (RAM) or permanent (Flash/ROM) memory. Now the program can be executed by the controller. Here, again, there are two possibilities. To search for errors in the so-called debug mode, the execution of the program can be controlled by the PC (The program is stored in the controller's RAM).

In contrast, a completed program is stored in the flash memory where it remains permanently, even after the controller has been switched off. Activation can be done by the controller itself; i.e. the controller no longer needs to be connected to a PC.



Remark

After the first programming or flash load, when the program is started for the first time, error 31, Invalid signature in EEPROM, may occur. After

clearing the error, the correct check-sum is generated and you can work normally.

MotionBasic online mode with Xemo!Go

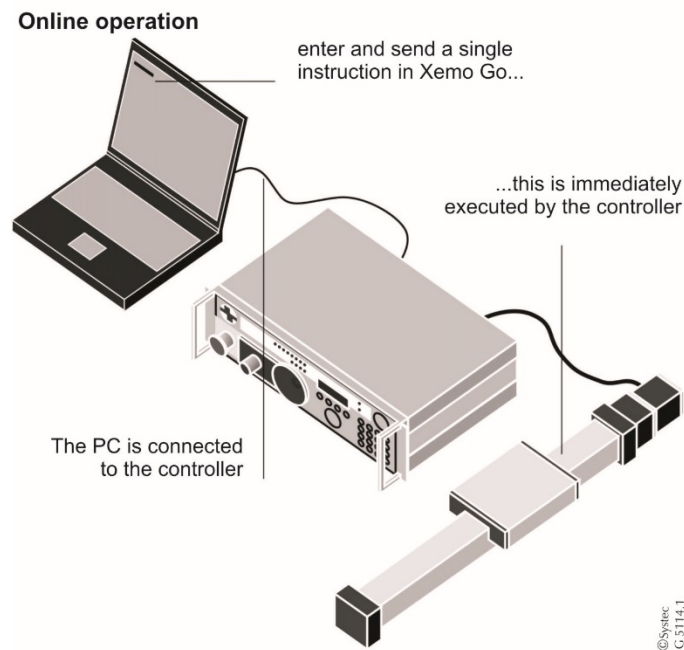


Fig. 3 MotionBasic: Online operational mode with Xemo!Go

In the online operational mode, you can communicate directly with the controller, not only with complete programs, but with individual MotionBasic instructions as well. This mode is intended for test purposes and initialization.

The application program Xemo!Go has been provided for the online mode. Xemo!Go can be called up directly from MotionBasic, but also functions autonomously.

In Xemo!Go, you can directly enter individual MotionBasic instructions and transfer them to the controller. The controller then immediately executes those instructions. For certain selected instructions, such as running axes or configuring outputs, there are even special command buttons. Accordingly, you can execute these MotionBasic commands with a click of the mouse.

3.3 Offline programming

You write a MotionBasic program in the Integrated Development Environment IDE.

Once you have written the program, translate (compile) it into a language which can be read by the controller. Then store the compiled program in the controller. There are two ways to do this. You can store your program in the main memory (RAM), where it will remain stored as long as the controller remains switched on. After the controller is turned off, though,

that memory is erased. Programs stored in the main memory can be tested from the PC with utilities such as e.g. single-step functions (see Fig. 4).

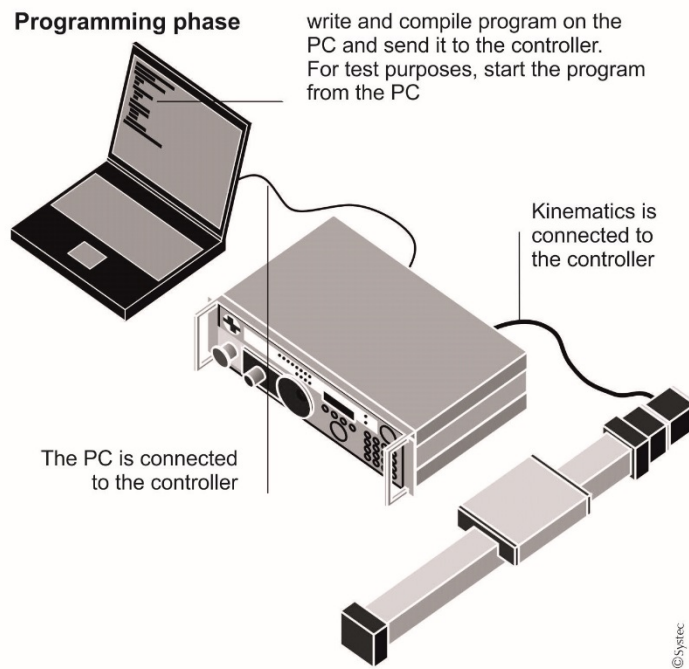


Fig. 4 Offline operation: During the programming phase, the PC is permanently connected to the controller

It is also possible to store programs in the controller's "Flash" memory where they will remain available even after it has been switched off and on. In that case, the PC can be detached from the controller. The program runs autonomously in the controller (see Fig. 5).

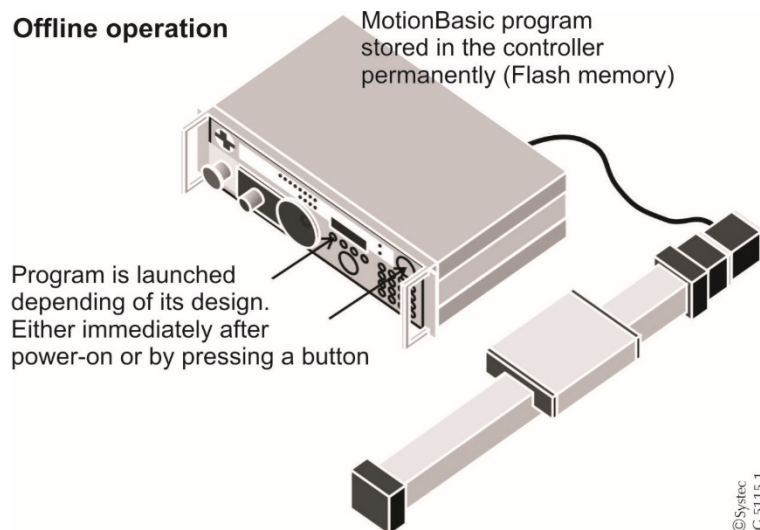


Fig. 5 Offline operation: The permanent connection to the PC is no longer needed

3.4 Online programming

In the previous chapter, we presented an example of an offline program. Offline means that a program is written on the PC and then stored in the controller, where it runs autonomously, i.e. without being connected to the PC:

MotionBasic permits a combination of online and offline programming. Online programming is the term used when commands are immediately executed via the programming interface.

Xemo!Go

A part of MotionBasic is the user interface Xemo!Go. There individual MotionBasic instructions can be entered into a command line and sent to the controller. The controller immediately executes these commands; they must not be embedded in a complete program. The syntax is identical to the syntax in offline programming, as described later in this manual. The command arguments may consist of compounded expressions. Variables, however, cannot be used.

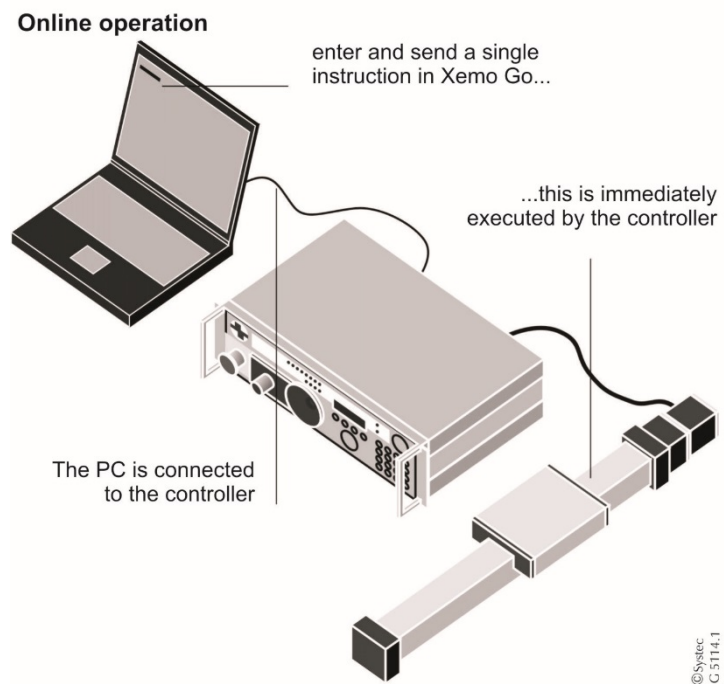


Fig. 6 Online operation: The PC is connected to the controller; individual commands are executed directly by the controller.

Online programming with Xemo!Go simplifies and shortens starting a machine up. The axes, inputs and outputs can be tested with your having to write a complete program.

4 Installation, starting up and ending

4.1 Installing MotionBasic 6

You will find the installation file suitable to your Xemo control,
MotionBasicIde_6.5.6.exe,
on your Systemec CD-ROM into the folder
Xemo\Xemo_Software,
or on your DriveSets CD in the folder
DriveSet_Software.

Installation file from the internet You can download the installation file as well from our website (<http://www.systemec.de/index.php?id=270&L=1>).

Difference of versions On the website you will find two different versions of the installation file. Reason: for Xemo controllers with integrated Ethernet interface, you need another program to update the Xemo firmware as for Xemo controllers without integrated Ethernet interface.

Xemo controllers with integrated Ethernet interface, Firmware 847, e.g. Xemo-Step, new generation of Xemo R and Xemo S:

XemoUpdate.exe

Xemo controllers without integrated Ethernet interface, Firmware 667, e.g. Xemo M, Xemo B:

ST10-Flasher.exe

Download IDE suitable to your Xemo controller.

Installation of MotionBasic 6 Start the execution of the installation via double click and follow the installation instructions. MotionBasic 6, Xemo!Go and the program to update the firmware will be installed.



Tip

The program XemoUpdate.exe can update not only the firmware of the Xemo controller, but also read and write the parameters of the Ethernet interface – via USB cable and with the operating mode switch set on "A".

4.2 Starting MotionBasic 6

Start the development environment with a double click on the symbol "MotionBasic 6.5.6" or via Start/Programs/Systemec/MotionBasic/MotionBasic6.5.6.
In case a newer program version is provided, the name will change accordingly.

4.3 Ending MotionBasic 6

Close the program by clicking on "Exit" in the "Files" menu.

5 The MotionBasic 6 work area

5.1 The application window

The following image shows the MotionBasic 6 application window

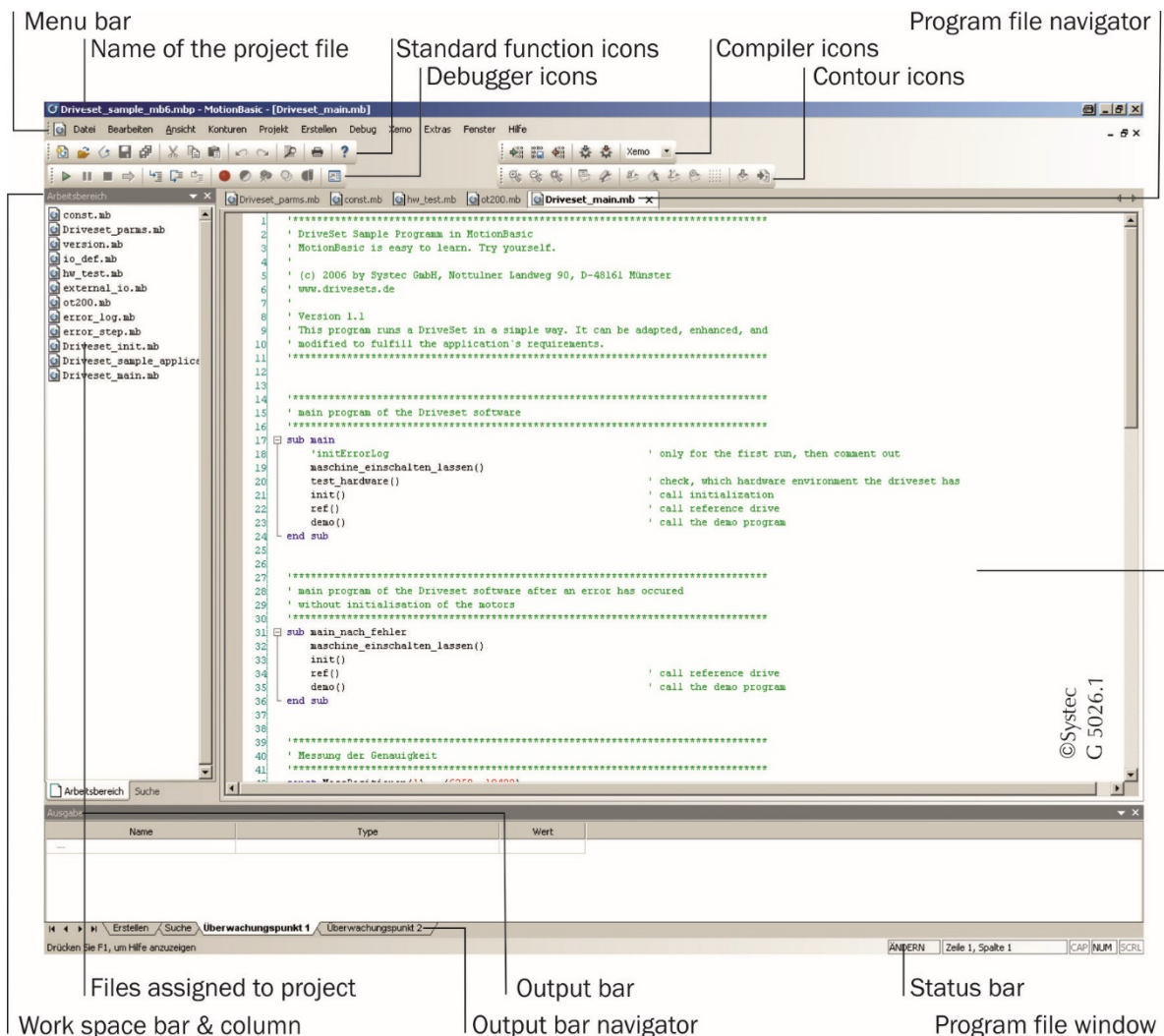


Fig. 7 MotionBasic application window

5.2 Areas and functions of the application window

- Menu bar** Area with pull-down menu options
- Name of the project file** MotionBasic differentiates between project and program files. At the top edge of the application window, the name of the project file presently open is displayed.
- Standard function icons** These icons encompass the standard Windows program functions such as copy, paste, cut, etc.
- Compiler icons** MotionBasic programs are translated (compiled) into a program code which can be read by machines. In this process, a so-called “Xemo code”

(XCode) is generated. Via these icons, you can e.g. call up the compilation, load the XCode into the controller's working memory or permanently file it in the flash memory.

Debugger icons	The debugger is used to search for programming errors. With it, you can control the execution of the program. There are, for example, various single-step functions for this purpose. You can likewise set breakpoints in order to halt the execution of the program at selected lines in the program.
Work space bar & column	The program files linked to specific MotionBasic project files are displayed in this column. By double-clicking on a file name, you can open any file not already opened. Further, you can enter variables from your program into a list. Then, when the program is being executed, the values which the variables assume are displayed.
Output bar	The output bar is divided into three subwindows. In the "Generated" window compiler commentaries are displayed when you translate (compile) a program. The two "Surveillance point" windows can be used to search for errors. You can enter variables from your program in a list here. Then, when the program is being executed, the values which the variables assume are displayed.
Output bar navigator	With the navigator, you can select one of the three windows "Generate," "Surveillance point 1" or "Surveillance point 2."
Program file window	The MotionBasic program files are displayed in this window. Here you can generate or modify programs. You can call up program files in this window independent of a specific project file.
Program file navigator	Use the navigator so select the file which you want to work on.
Status bar	This gives you information about the present status of the program. For example, if you are generating or modifying a program, the cursor position will be displayed in the field by its column and line location numbers.

5.3 Editor functions

The editor of the IDE offers a range of convenient functionalities.

Fold procedure	By means of the minus square (see Fig. 8), you can fold your subfunction; by means of the plus square, you can unfold it (see Fig. 9). The area which is included by the subfunction is indicated by stylized brackets for better clarity.
-----------------------	--

In the folded condition (see Fig. 9), you only see the first line of the subfunction as headline.

```

220
221 |*****|
222 |'|
223 |*****|
224 | sub ref_abfrage
225 |   dim key as integer
226 |
227 |   cls()
228 |   printxy(1,1, "Driveset Demo")
229 |   printxy(1,4, "Ref      ")
230 |
231 |   do
232 |     key = keyread()
233 |     select case key
234 |       case _KEY_F1
235 |         exit do
236 |     end select
237 |   loop
238 |
239 | end sub
240
241 |*****|
242 |'|
243 | call the homing routines for all axes
244 |*****|
  
```

Fig. 8 Example of the editor functions syntax emphasis, line numbering and folding subfunctions.

```

219
220
221 |*****|
222 |'|
223 |*****|
224 | + sub ref_abfrage
240
241 |*****|
242 |'|
243 | call the homing routines for all axes
244 |*****|
  
```

Fig. 9 By using the plus square, you unfold your subfunction.

Line numbering

Line numbering offers you some good orientation within the individual files.

Syntax emphasis

Different language elements are highlighted through different text colours in the editor (see Fig. 8).

6 File organization in MotionBasic 6

6.1 Introduction

This introduction is provided for beginners who are not familiar with the concepts “project files” and “program files.” If you are already familiar with this kind of file organization, you can skip this and proceed to section 6.2.

6.1.1 File organization principles in MotionBasic

File organization in MotionBasic

There are two kinds of files used with MotionBasic; project files and program files.

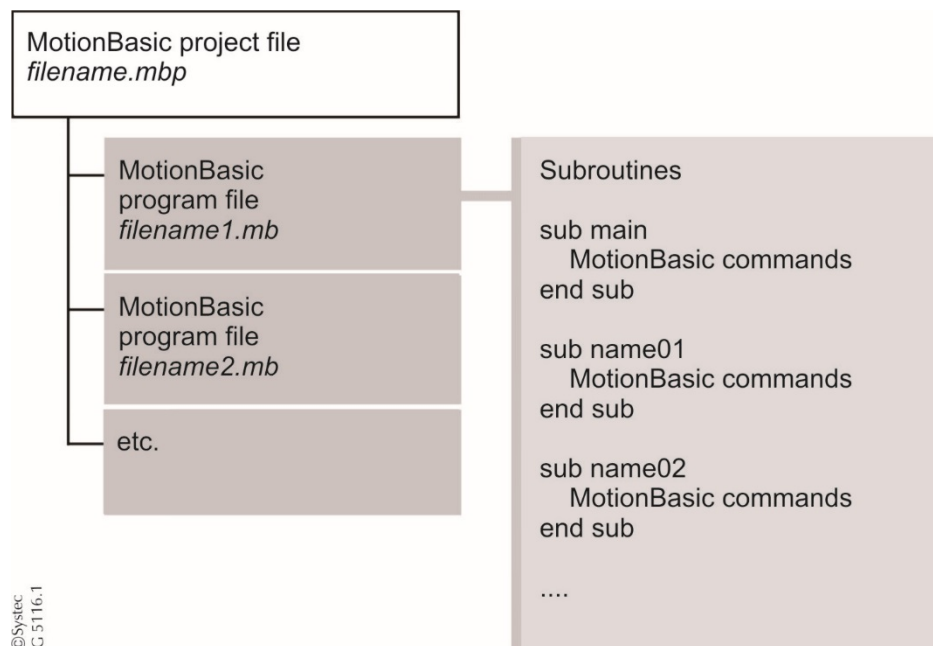


Fig. 10 File organization in MotionBasic

Project files

In MotionBasic menus, displays, etc. project files are simply marked with “project.” They contain no program code, but rather simply serve to organize program files. You can picture them as file folders which contain one or more files. You can recognize project files by the “.mbp” ending. In addition to their organizational functions, project files also fulfill an important technical function in regard to controllers. Before they are transferred to a controller, MotionBasic programs must be translated into a language which can be read by machines. The translation process itself is called compiling and it results in a so-called Xemo code, called an “XCode” for short. Compiling of MotionBasic program files is possible only when these are incorporated into a project.

Program files

You write your MotionBasic programs in program files, which, in MotionBasic, are simply called “files.” These program files comprise the program code for running axes, surveying inputs or managing outputs.

For certain machine control functions such as initialization or running axes, you can set up your own files; e.g. one file with the name “init.mb” and another called “runaxis.mb.”

You cannot simply compile single or multiple files by themselves and transfer them to a controller. They must first be assigned to a project. You need to create a project file even for a single program file.

Subprograms

A program file can contain a number of MotionBasic subprograms. However, one (and only one) of these must bear the name “main.” After being started, the controller searches for this “main” subprogram and executes it first. If there is no subprogram “main,” the compiler announces an error.

If you incorporate a number of program files into a project, the subprogram “main” must be included in one of these files. The title “main” can be used only once in a project.

6.1.2 Advantages of file organization

Advantages of file organization

Which advantage do you as the programmer gain from the file organization in MotionBasic?

This functionality is especially helpful in generating and maintaining extensive controller instructions. An example for clarification: suppose that a dual-axis system is used for inspection purposes, and there are three different kinds of inspection involved. Depending on the specific inspection, the axes follow different routes. System initialization, reference runs and security functions are the same for all inspections though.

In such a case, MotionBasic program files such as these would be appropriate:

Init.mb	(Initialization, reference run and security)
RunAxis1.mb	(Inspection route 1)
RunAxis2.mb	(Inspection route 2)
RunAxis3.mb	(Inspection route 3)

These program files would be organized in three projects

Project file	Program file
Investigation1.mbp	Init.mb., RunAxis1.mb
Investigation2.mbp	Init.mb., RunAxis2.mb
Investigation3.mbp	Init.mb., RunAxis3.mb

Even at the time when you are generating a program you can see one advantage of this file organization. Program segments which are needed in all projects only need to be programmed one time, and it is necessary

to generate only one file. Let us suppose that a change must be implemented in the security function: you need alter only one program file – the file `init.mb` – for this central assignment. Because all of the projects address “`init.mb`,” this modification is now valid for all three inspections.

Summary

In MotionBasic there are program files and project files. You write your MotionBasic program code in the program files. Such program files can be generated and modified individually, but they cannot be compiled and transferred to a controller.

These program files are organized in projects (project files). Because only projects can be compiled and transferred to a controller, program files must be assigned to a project. One file, though, can be used in a number of projects. A file which has been assigned to one or more projects can be modified without any of those projects being open.

6.1.3 Project files

Project files – simply called “projects” for short – are used to organize MotionBasic programs and their files. Projects do not contain program code, but instead reference program files (i.e. files containing program code) which have been assigned to them. In addition, project-related information is stored in them. Such information can include, for example, the number of open program files or the configuration of the debugger.

Generating several program files makes it possible for you to divide a complex application into subordinate tasks. You can then program these subordinate tasks individually and independent of one another. These individual program segments are then combined into a main file by the project file.

In addition to their organizational functions, projects also have a technical function as well: in MotionBasic, only projects can be compiled. Put more precisely, program files are compiled through the projects to which they have been assigned. Therefore, even if you have only one program file, you must generate a project for that single file.

Functions in regard to project files will be discussed in a subsequent section.

6.1.4 Program files

You write your MotionBasic programs in program files, simply referred to as “files” in MotionBasic. You can recognize these by their “.mb” suffix. A MotionBasic program file can contain one or more subprograms. It must, however, have one subprogram with the name “main.” The subprogram with the name “main” is the first part of the program which a controller locates and then executes. If you generate several subprogram

files for one project, only one of those files can contain the subprogram "main."

It is quite important that single program files cannot be compiled and transferred to a controller without being contained in a project. More about that in the next section.

Writing programs

The basic functions of the program editor correspond to those of a word processor. In the menu "Edit," you will find commands such as cut, copy, paste, etc. See section 7.10.3. for further descriptions.

The program editor recognizes MotionBasic instructions as well as value allocations and displays these in different colors.

With the help of a tabulator function, you can structure the program text with indentations. Further, you can modify font type and size in the dialog window "Format" which you can find under the menu "Extras / Options"

The file functions for program files are described in the following section.

6.1.5 Administration of project and program files – schematic display

Project and program files can be opened in various constellations. If you are unfamiliar with this kind of file organization, the possibilities which it offers may at first seem rather confusing. For that reason, we are providing a short synopsis in the form of a schematic display here.

Constellation 1

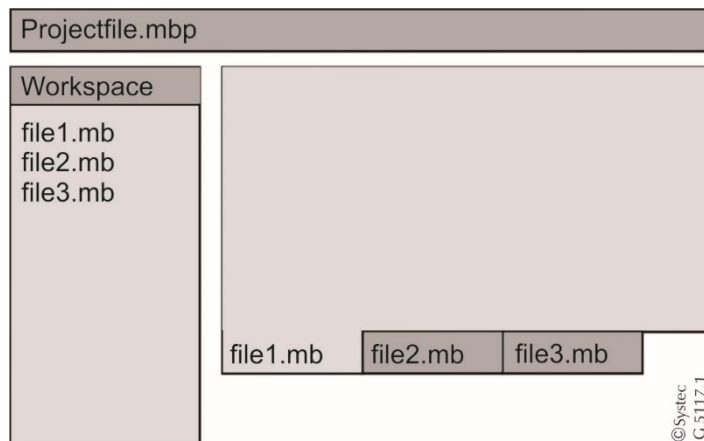


Fig. 11 Project and program files are open

Project file	Program files
Open	All program files belonging to this project are open (File1, File2, and File3)

Constellation 2

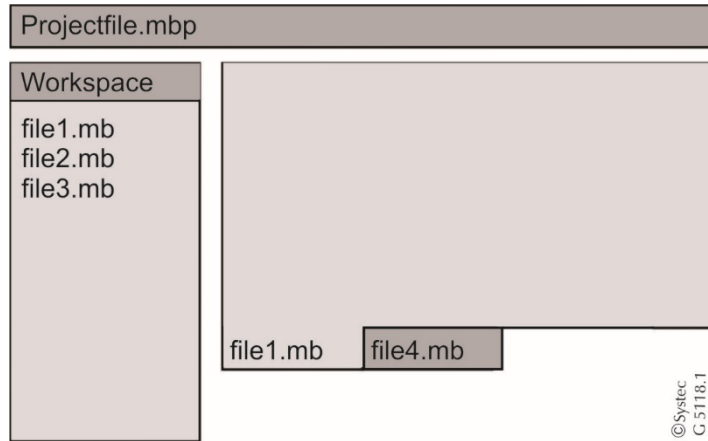


Fig. 12 Project and a file pertaining to the project as well as a non-project file are open

Project file	Program files
Open	One program file (File1) belonging to the project is open. Another file (File4) which does not belong to the project is also open.

Not all files which belong to a project must be displayed in the file window, i.e. be open. If you need to work on only one file of an extensive project, you can open this one specific file by itself. Likewise, more files can be open than those which belong to the project.

Constellation 3

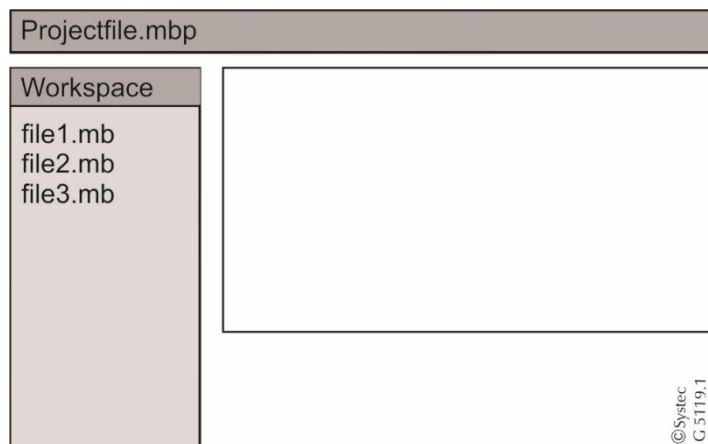


Fig. 13 Project files open, program files closed

Project file	Program files
Open	No program files belonging to the project are open.

Basically, this constellation does not make any sense. A file/The files can be opened through a menu or by a double click on the file name(s) in the "Workspace" window,

Constellation 4

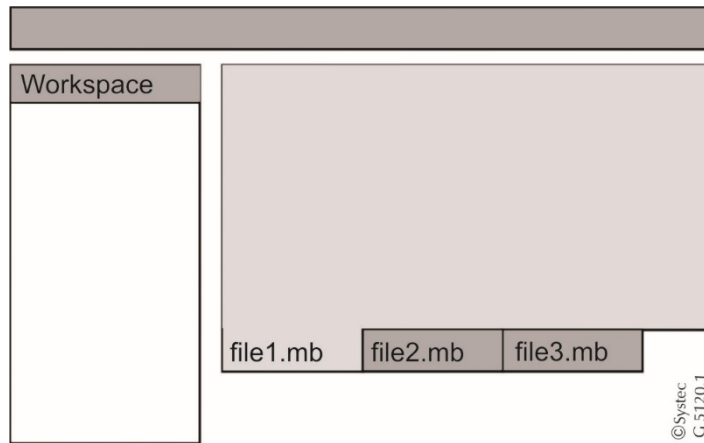


Fig. 14 Several program files open but no project; compilation cannot be effected.

Project file	Program files
Not open	One or more program files are open.

Program files can be edited without a project being open, but compilation is not possible then. Compilation can only be done if a project is open. This functionality only makes sense when you want to quickly and easily view something in a program file. You can open such a program file with a double click. In such a case, it will be opened as a single file without the associated project.

Project files can also be created and/or edited without being assigned to a project. If such files are to be compiled, though, they must be assigned to a project.

6.2 Generating and editing project files

Generate a new project file With the command “New file” from the “File” menu you can generate a new project file. After calling the order up, the following dialog window “MotionBasic – new project” appears.

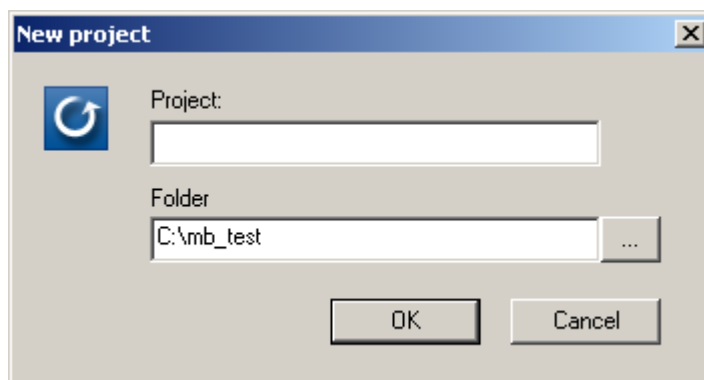


Fig. 15 Creating a new project

In this dialog window, you should enter the project name and select the directory where you wish to save the project file. After you have confirmed the entry, the project file will be saved immediately. Up to this

point, no program file has been assigned to it. This is the next step to be taken. Only one project at a time can be open.

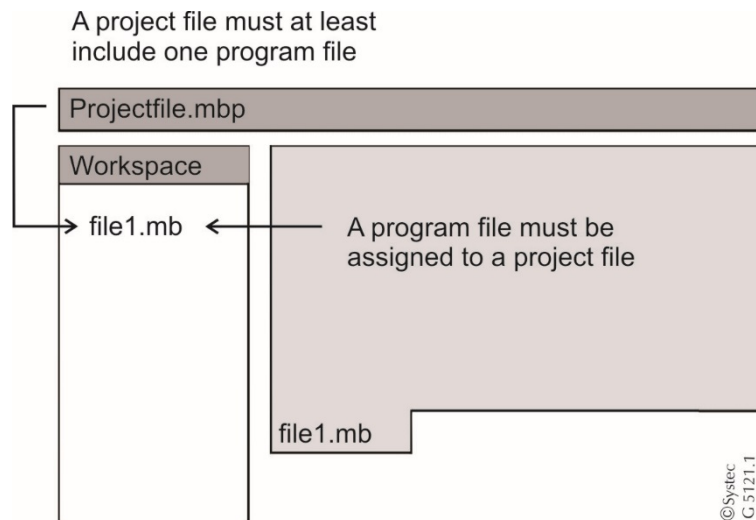


Fig. 16 Project file with at least one program file

Assigning a program file to a project

After you have first created a new project, no program file has been assigned to it yet. What is more, if you then open program files or create new ones via the menu "File," these will NOT automatically be assigned to the project. You need to specifically define their assignment. You do that with the command "Add to project" in the "File" menu. Here you have the choice between "New file", "Existing files", "New contour", or "Existing contour".

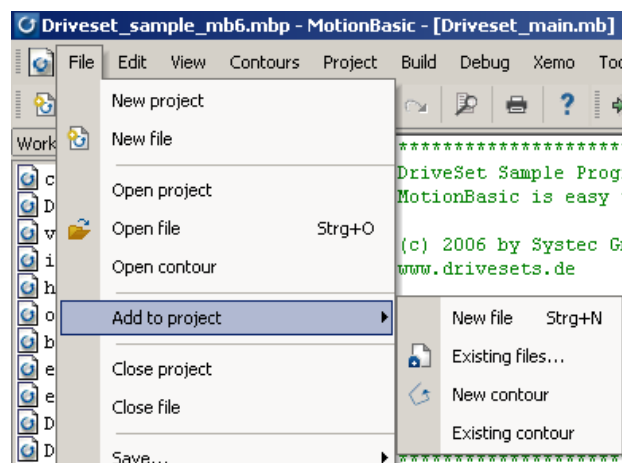


Fig. 17 Assigning a file to an existing project

Add to project / New file

With this command you generate a new program file which is immediately incorporated into the project just created/opened. You can recognize this because the program name immediately appears in the workspace window.

Add to project / Existing files

With this command you assign already existing – or, more precisely – previously saved program files to the newly opened project. A file which you have just generated with the "New file" command and which you

have not yet saved cannot be assigned to the project at this time. Assignment of a program file is not possible until that file has been saved. When a program file is added to a project, it is not important if the file is open or not.

Save / Save project

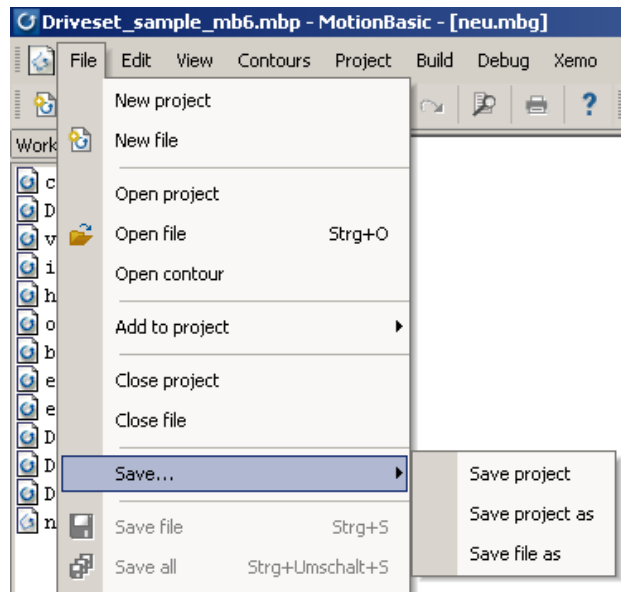


Fig. 18 Saving a project

You can save a project file with the command “Save / Save project” from the “File” menu. Information relevant to this project file is saved through this command as well. That includes the file name as well as project-related configurations. Those can be, for example, settings for the debug mode as well as breakpoints and previously surveyed variables.

Save as/Save project as

You can save your project using a different name if you select “Save project as” in the menu item “Save”. Please note that only the project file, .mbp, is saved but not the program files.

Open project

You open an already existing project with the command “Open project” from the “File” menu. The project will be opened in exactly the same condition as it was when you saved it. In the “Workspace” window, all files will be displayed which belong to the project. Whether or not these files are open depends upon their status when they were saved the previous time.

Deleting a program file from a project

If you wish to remove a program file from a project, select the proper file in the “Workspace” window by clicking on it with the mouse. You can then delete its allocation with the command “Delete” in the “Edit” menu. The program file is now no longer part of that project, but it does still exist as a file.

6.3 Generating and editing program files

Generating a new program file

You generate new program files with the command “New file” from the “File” menu. It is not necessary for a project to be open for this. You can also generate several new files.

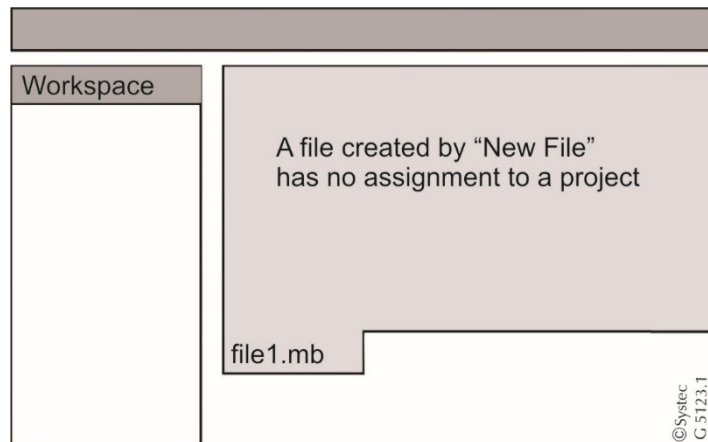


Fig. 19 A newly created file is not integrated into a project yet.

A program file newly generated by the command “New file” in the “File” menu is not immediately assigned to a project, regardless of whether a project is open or not.

Open existing program file

With the command “Open file” from the “File” menu, you can open previously saved program files. You can also open several files at once. A project does not have to be open for this. It does not matter if the existing files belong to another project (or projects).

Save program file

You can save your program files with the command “Save file” or the command “Save / Save file as” in the “File” menu. This command applies to a single file.

If you have more than one file open and have modified some or all of them, you do not have to save them individually. With the command “Save all” in the “File” menu, you can simultaneously save all open files.

7 Compilation

7.1 Compilation of the MotionBasic program

In order for the controller to receive an executable program, the MotionBasic program text must be compiled. That translates the MotionBasic instructions into a so-called "Xemo code" – "XCode" for short. You can run this compilation process off and on while you are generating the program. The compiler checks for various kinds of errors, including those in MotionBasic instructions (syntax errors), so this gives you running feedback on the accuracy of your programming.

Only project files can be compiled

For a program file to be compiled, it must be assigned to a project file. Individual program files cannot be compiled. As a result, it is necessary to create a project for even one single program file. You can find extensive information about this topic in section 6.1.

If you have not opened a project, do so now by executing the command "Open project" in the "File" menu. If you have not yet created a project, then do that with the command "New project" in the "File" menu. Assign your new program file to the project file with the command "Add to project / Existing files..." in the "Project" menu. You can find extensive information about this topic in section 6.1.3. "File organization in MotionBasic."

Compiling

Select the command "Compile" from the "Create" menu.

The compiler now translates all program files belonging to the open project. It does not matter here if the program files are open or closed. During this process, the results of the compilation process will be displayed in the "Output window" of the "Create" dialog window. Both the compiled files and any errors discovered will be listed.



Remark

Please note that the MB compiler is a two-pass compiler and therefore also accepts variables and functions which are to be declared at a later point, but that the preprocessor has only one pass.

So constants which are to be used in the preprocessor must have been declared previously. If this rule is not obeyed, odd effects may occur whose causes can be difficult to determine. Therefore, it is strongly recommended that constants for the preprocessor are placed in a central file at the start.

The preprocessor processes the files in the order in which they appear in the project window. The context menu of the project window also includes an item for rearranging the files so that you can move a file subsequently to the start and thus change the order of processing.

Compiler error messages

During the translation process, the compiler checks the program text for syntax errors. If it discovers an error, the compilation process is immediately interrupted and the error displayed in the "Create" window. At the same time, a dialog window with the same text message will appear. An arrow in the left column of the file window will point to the program line

in which the error has been discovered. After the error message has been confirmed, MotionBasic will return straight to the line containing the error in the open program file. You can now edit the program text and eliminate the error. Afterward, start a new compilation. Compilations always run up to the first error message.

You can find extensive information about possible errors in section 9.1.

Successful compilation

When the compilation has been successfully concluded, the message “Done” will appear at the end of the list in the “Create” window. Your complete MotionBasic program has now been translated into the XCode which can be read by machines. How do you continue with this XCode now?

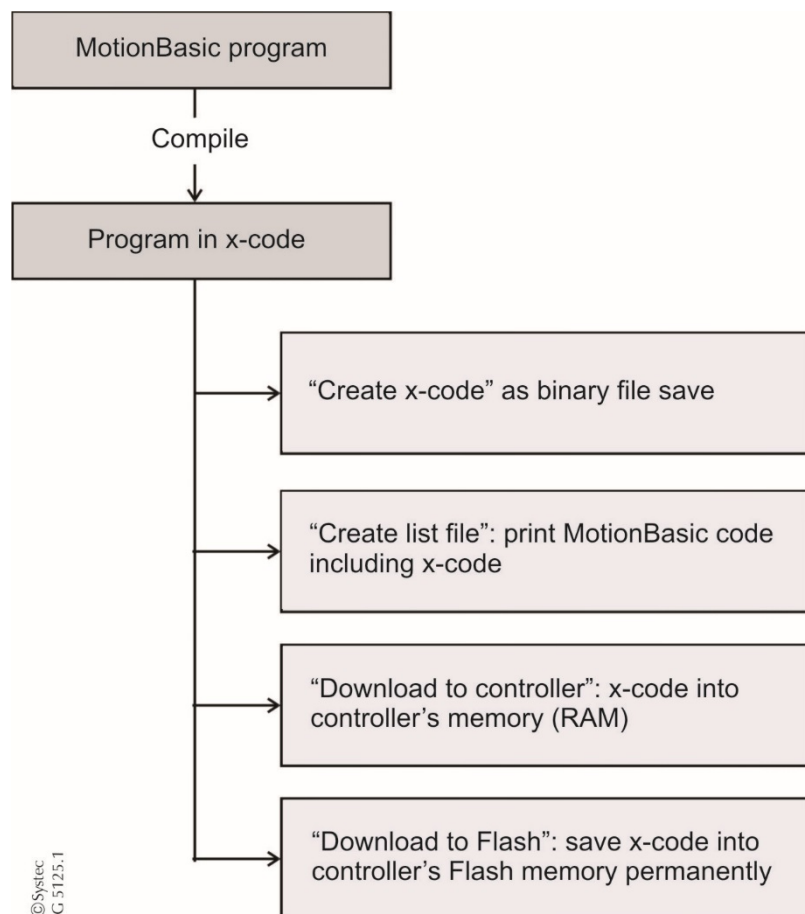


Fig. 20 Compilation of MotionBasic programs

It makes sense to transfer the XCode to the controller. There are two alternative ways to do this. You can transfer the XCode into the controller’s working memory (transient) or flash memory (permanent). In both cases, the PC and the controller must be connected by the interface cable. Configuration of the interface parameters will be described in Chap. 8.1.

7.2 Using the XCode

- Loading XCode into the controller** This is done through the command “Download in RAM” in the “Create” menu. The XCode is temporarily stored in the controller’s memory. It will be deleted as soon as you turn the controller off. This storage mode is very important when searching for errors in the program: debugging is possible only when you load the XCode into the controller. A program which has been stored in the controller’s working memory can only be executed from the PC.
- Loading the program into flash memory** The XCode is transferred to the controller and stored there permanently through the command “Download in ROM” in the “Create” menu. The PC no longer needs to be connected to the controller, as the controller itself can execute a program stored in its flash memory. The program is started by switching the controller off and on. Of course, the program can only run independently when it has been programmed accordingly.
- Create an XCode file** XCode can be written into an external file with the command “XCode / Create XCode file” in the “Create” menu. After this command has been called up, a dialog window appears in which you can determine the name and file location for the XCode file. An XCode file is a non-readable binary file. It makes it possible for you to pass on compiled MotionBasic programs which exclude the recipient from access to the source code. With the command “Download XCode file into ROM,” in the “Xemo” menu, these files can be transferred from the PC to the controller and stored there.
- Create a list file** Through the command “XCode / Display XCode” in the “Create” menu, a file is created from a MotionBasic project which lists all the MotionBasic instructions included in it. For each instruction, the specific XCode is displayed together with its specific line number. This file is useful when a Xemo controller running independently of a PC announces an error. The error message contains the line number of the XCode error. With help of the list file, you can trace down the location of the instruction with the error.

8 Connecting the PC with the controller

8.1 Setting the interface parameters

In the menu Extras/Options, please select the "Communications" tab.

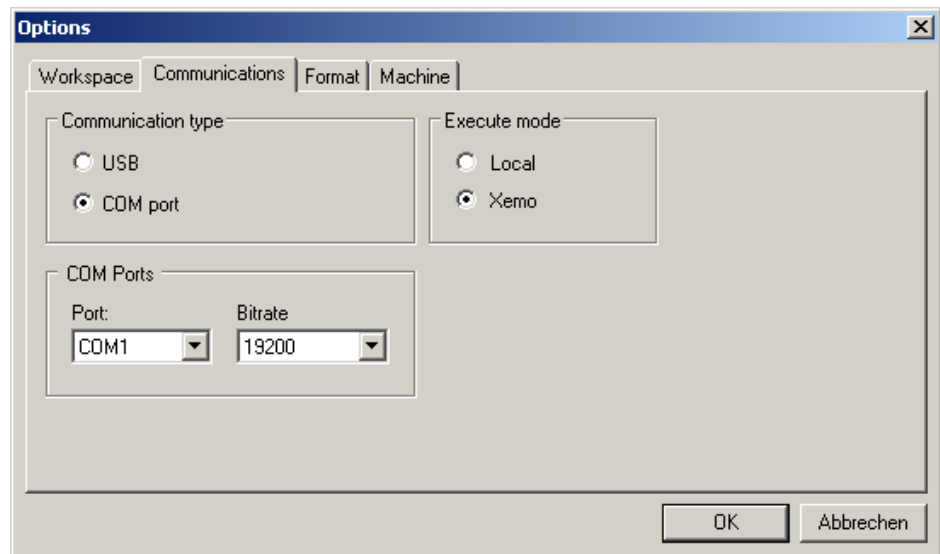


Fig. 21 MotionBasic: setting the interface parameters

In the dialog window, select the interface used on your PC. You need to set the bit rate at 19200 for a serial interface.

Type of communication

In the dialog box, select the interface used on your PC.

USB

Do you want to talk the Xemo control via USB, select USB.

Ethernet

Also adjust USB for Xemo controllers with integrated Ethernet interface.

In addition you need to assign the value 3 (Ethernet as the desired interface) to parameter 1407 and the IP address of the Xemo controller to parameter 1409 in your MotionBasic program. Comfortable, make this assignment via ComDevice before the main program call sub main, see example. After compiling the MotionBasic IDE then "knows", about which interface and to which Xemo controller the program should be sent.

Example

```
ComDevice = TCP "192.168.1.204"

'Main program
sub main
...
end sub
```

'MotionBasic code



Tip

If you are even unsure what IP address has your Xemo controller: with the program XemoUpdate.exe, you can read and write the IP address (and other parameters for the Ethernet interface) – via USB cable and with the operating mode switch set on "A".



Remark

For Xemo controllers with integrated Ethernet interface, the Setup parameter 1407 must be set in any case. This is comfortable about the program XemoUpdate.exe – via USB cable and with the operating mode switch set on "A".
 Select the desired interface through ComDevice in the tab Setup, see Fig. 22. Then click on "Schreiben" (write).

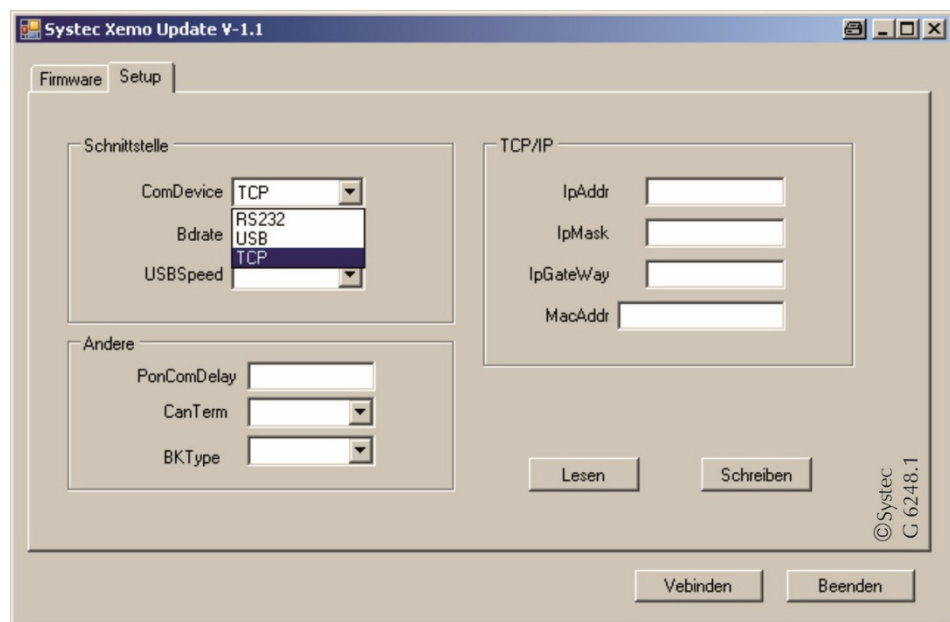


Fig. 22 XemoUpdate.exe: Tab Setup to adjust the interface at the Xemo controller

RS232 – COM Ports

For the RS - 232 interface, select COM port. In the "COM Ports" section, set the corresponding interface on 19.200 baud rate.

Execute mode

In the "Execute mode" select Xemo. With this setting, you can transfer programs to the Xemo controller. In the "Local" mode, no data is transferred to the controller. It serves simulation purposes on the PC.

8.2 Program start and run

Hardware requirements

Your PC must be connected to the controller via either the serial or a USB interface. The interface parameters must be set as described in the previous section.

If the Xemo controller is to receive a program from the PC, the operation switch must be set to either "0" or "F." In position "F," the controller is ready for operation once it has been turned on: a program can be started from the PC. In position "0," the controller automatically starts a program stored in its flash memory after being turned on.



Fig. 23 Operation switch on the Xemo controller at position "F".

The controller can be controlled from the PC in the operation switch positions 0-2 and "F" (online, mixed operation, Xemo!Go).

Software requirements

Your MotionBasic program must have been compiled without errors before it can be transferred to the controller. Select "Compile" in the "Create" menu. Transfer the compiled program to the controller with the command "Download in ROM" in the "Create" menu.

Your program is now temporarily stored in the controller. You can only start it from the PC. In order to load a program into the controller, any program running at the moment must first be stopped. If that is not done, MotionBasic will display an error message and require that the program be stopped.

This section describes the command for starting the program and the possibilities of interrupting the program during execution. In the following section "Error search," we will show you the possibilities of step-by-step control. If you are a beginner or even an inexperienced user, you should read the next section and then execute your program step-by-step.

Execute (F5)

Select the menu command "Execute" from the "Debug" menu. The program will start. The text message "Execute" appears at the left in the status bar. If the program has been modified previously, it will be compiled again prior to starting. The program will continue running until it stops itself – if, of course, it has been properly programmed.

End (Shift + F5)

With "End" in the menu "Debug," the program will be completely stopped. The debugger is no longer active then. A new program run (re-start) is only possibly from the very beginning of the program – i.e. at the "main" routine.

Stop

With "Stop" in the menu "Debug," a program is stopped during its execution. It stops at the program line active at that moment, but is not actually ended. Active motor motions, though, are broken off. In the status bar, the text message "Stopped" is displayed.

The execution of the program can be continued from the place where it was interrupted. That is possible with a step-by-step command such as "Process step" or "Single step." Part of the program can be executed with the command "To the cursor" or to its end by clicking on "Execute." These commands are described in detail in the following section.

- Re-start**
(Icon support text
“Re-start”) A program which has been stopped can be re-started with “Re-start” in the “Debug” menu. All variables are returned to their original status, execution of the program starts at the beginning.
- Additional functions** For testing a program, the development environment offers additional possibilities of intervening in the execution of a program. You will find a detailed description in the next chapter.

9 Error search with the IDE

MotionBasic's integrated development environment (IDE) provides a variety of possible ways to develop programs. Among those are project administration, the modular structure of programs, and very fast compilation. However, any program can contain errors (bugs) which prevent it from running as desired.

Debugging

"Debugging" means the systematic search for and correction of program errors. The IDE puts at your disposal a group of tools which are described below.

9.1 Kinds of errors

There are three basic kinds of errors in programs: syntax, runtime and programming errors.

Syntax errors

Syntax errors occur when MotionBasic's syntax is not observed during programming. A program file cannot be compiled if it contains incorrect or wrong instructions. If the compiler encounters a false instruction during compilation, it will break the process off. The line with the false instruction is indicated by an arrow in the left column of the file window. Further, a window with a text message about the error appears. That same message will also be displayed in the "Create" window of the output window.

Common causes of syntax errors are mistakes in spelling or the omission of commas in instructions as well as references to non-defined variables. In procedure or functional call-ups, false kinds of variables or an incorrect number of parameters are often communicated.

After an error has been eliminated, a new compilation run can be started. If all syntax errors have been eliminated and the program file is compiled without any errors, the program can then be executed. At this stage, you can then search the program for runtime and logical errors.

Runtime errors

Runtime errors occur if an unacceptable action is implemented during execution of the program. In such a case, the program itself contains valid instructions and, accordingly, compilation is carried out without error. Instead, the errors will occur during the execution of the program. Such errors include, for example, exceeding the limits of an area in an array, or definition of a trajectory path which is larger than the physical dimensions of the axis. If MotionBasic discovers such an error, it issues the following message:

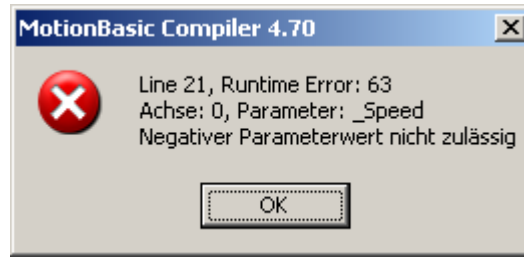


Fig. 24 Example of a runtime error, error 63 in this case

Error message: runtime error

After the error message has been acknowledged, MotionBasic goes directly to the program line containing the error. In addition, this line is marked by a blue arrow displayed in the left column of the file window.

Programming errors

Programming errors are fundamental errors in a program’s structure, usually caused by a mistake in the programmer’s thinking. All instructions are correct in their syntax, all values are within the permissible limits, and, accordingly, the program is compiled without problems. What the program does, though, is not what the programmer intended: variables assume false or unexpected values, axis movements are not executed as desired, program instructions are not properly completed.

Such errors are often quite hard to find as they cannot be discovered automatically. Accordingly, the integrated debugger provides a variety of tools to help you track down such errors. By monitoring instructions and variables as well as through direct intervention in the execution of the program, you can isolate and eventually resolve program errors.

9.2 Methods of error search

Sometimes when a program does not function as you want it to, the errors are immediately obvious and you can easily and quickly correct them. Errors are, however, often hidden and quite difficult to find, especially when sections of the program interact with one another. In such cases, it is often useful to interrupt the execution of the program at a particular point and then continue with its execution step-by-step, all the while monitoring the values of variables and expressions. This controlled program execution enables a systematic search for errors. In this way, an error can be isolated and then successfully eliminated.

The following sections describe the various possibilities made available by the IDE’s debugger.

Monitoring program execution

The most important function of the debugger lies in its monitoring of the execution of a program. If you execute the program step-by-step and observe exactly how each instruction is carried out, you can determine more easily which section of your program is causing problems. The debugger provides four tools for this method:

- Execution of individual instructions
- Entry into subprograms
- Execution of a program up to a specific point
- Returning a program to its original status

Monitoring variables As you execute a program line-by-line, you can monitor values and/or changes in the values of variables. For this purpose, MotionBasic provides two so-called monitor windows in the output bar. There you can enter into a list those variables which have already been defined in a program. During execution of the program, the list will then display any changes in the values which the variables take on.

Simulation of the Xemo controller's user interface

Xemo R controllers can be equipped with an optional display and control panel. The Xemo B also includes a display and control panel. Both operational units have the same functionality, contain almost exactly the same keys and have similar displays, but they do differ in their configurations. The Xemo R's display has four lines of 20 characters each and four function keys beneath the display. The Xemo B's display has four lines of 40 characters each and six function keys beneath the display.

For the Xemo S controllers, Systemec offers a separate control unit (OT300) which looks exactly like that of the Xemo B's control panel.

The Xemo control panel is depicted in MotionBasic. That means that the control units can be simulated. You can call up the control panel through "Extras / Xemo-panel."



Fig. 25 Simulation of the Xemo controller's interface

All of the user interface's function, e.g. entries via keys, text shown on the display, or activation of the LEDs, can be simulated. The controller does not have to be connected to the PC, but the setting "local" must be selected for the interface (Extras/Options/Interfaces)

9.3 Monitored program execution

Single step (F8)

When you debug a program, a program line is the smallest unit which you can execute. If you include a number of instructions in one line, though, these are not executed individually, but rather in one step together.

All debugger operations, the execution of individual instructions, the entry into a subprogram, and the interruption of a program relate to lines. The debugger shows through a yellow arrow in the left column of the file window which line it will process next.

The step-by-step execution of the program is possible through the command "Procedure step" in the "Debug" menu. With "Single step," MotionBasic branches off into subprograms or functions and executes these step-by-step as well.

Procedure Step without branching off into subprograms

"Procedure step" and "Single step" differ in the way in which they manage subprograms and/or function call-ups. Whereas "Single step" jumps into subprograms, "Procedure step" treats them as individual instructions and continues with the next line after a subprogram has been executed. If you are fully certain that a subprogram functions properly, you do not have to go through it with single steps. You can use "Procedure step" and avoid branching off into the subprogram.

Incidentally, you can alternate the use of "Procedure step" and "Single step." As long as no subprogram or function call-up is pending, both commands have the same effect. Let us assume that you encounter a number of subprogram call-ups during the step-by-step execution of your program but do not want to examine all of them. Then use "Procedure step" in that case and use "Single step" for those which you wish to examine.

If you change program text with "Single step" or "Procedure step" during the execution of a program, a warning will appear:

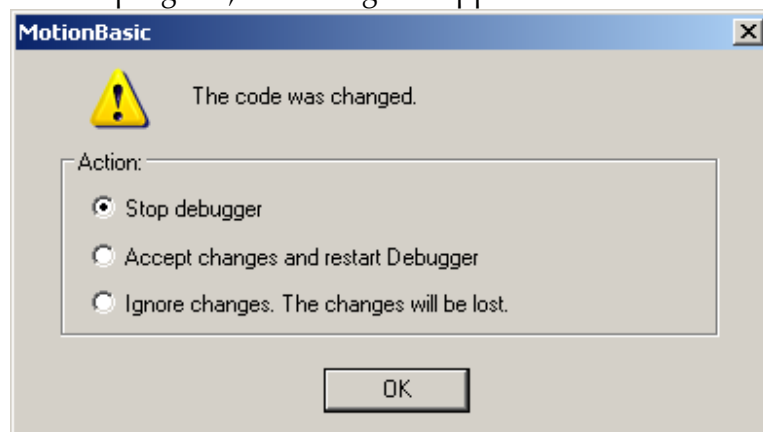


Fig. 26 Warning and enquiry if the program was changed during the execution

Jump back (Switch + F11)

Before a change to a program is implemented, you must stop the program. Afterward you must re-start it again for testing.

With “Jump back” in the “Debug” menu, a subprogram which you have branched into will be executed as normal from the present program line until the end of the subprogram. Then MotionBasic leaves the subprogram and waits in the program line after the subprogram call-up.

Interrupting the execution of a program Up to cursor

There are two ways to run a program with the debugger up to a specific point and stop it there.

The easiest way is simply to place the cursor at the line at which the program is supposed to stop. Afterward, click on “Up to cursor” from within the “Debug” menu. At first the program will run normally up to the instruction at which the cursor is standing. There it will stop. Now you can check the variables and either let the program continue running or execute it step-by-step.

Breakpoints

The second possibility of stopping a program at a certain point consists of setting a breakpoint at the chosen line. When the program is running, it will be stopped at the instruction marked with the breakpoint. Breakpoints are more flexible in their use because you can set a number of breakpoints at different places throughout a program. You will find a description of breakpoints in section 9.5.

9.4 Monitoring variables

If you debug a program step by step during its execution and want to keep an eye on the variables while doing so, you can call up two so-called “Watchpoint” tab cards. There you can enter the variables which you wish to monitor in a table. The values of all monitored variables are then displayed at all times during the execution of the program.

Opening the “Output” window

You can find the Watchpoint tab cards in the “Output” window. If that window is not opened, you can call it up in the main menu under “View / Output.”

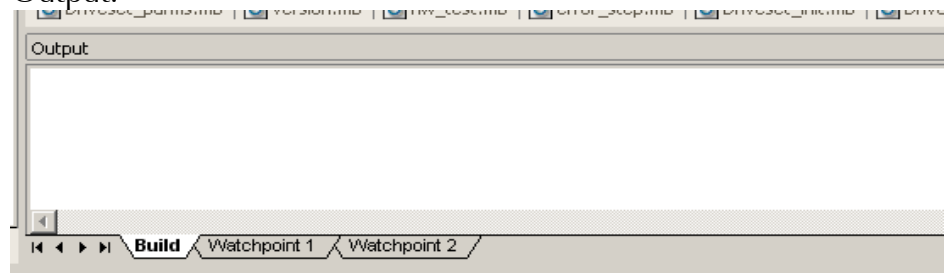


Fig. 27 Output bar, monitoring variables in the “Watchpoint” windows

Adding variables

Select either the “Watchpoint 1” or “Watchpoint 2” window.

Enter the name of a variable in the table by clicking on a blank space in the “Name” column. That marks the “Name” and the neighboring “Value” fields. A click on the right mouse key opens a context menu.

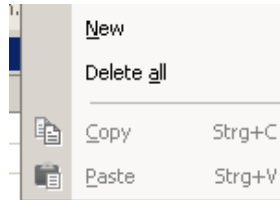


Fig. 28 Menu window for entering a variable

Select “New” An entry field will open in the first table area or, if variable entries already exist, in the first free table area. Now you can enter the name of the variable which you wish to monitor during execution of the program. Confirm your entry by hitting the return button or by clicking on the mouse outside the entry field. The variable is now included in the list of the expressions to be monitored.



Remark

Please keep in mind that variables, which you have declared by Defin or Defout, are not yet supported.

Deleting variables

Place the cursor on the variable which you wish to delete. When you click the right mouse key, the variable will be marked and an option window will open. Select “Delete” there.

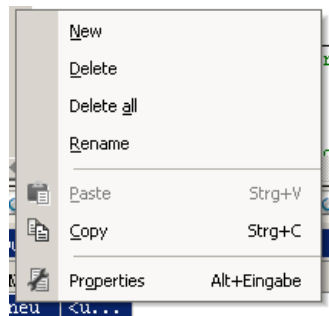


Fig. 29 Context menu with existing variable entries

Renaming variables

Place the cursor over the variable which you wish to rename. When you click on the right mouse key, an option window will open. Select “Rename” there. The variable name will be highlighted and can now be changed.

Deleting all variables

Place the cursor over one of the table fields in the “Watchpoint” window and click on the right mouse key. In the option window which appears, select “Delete all.”

9.5 Breakpoints

To aid in searching for errors, MotionBasic provides you with the possibility of setting breakpoints in program lines. Execution of the program will then be halted at those lines.

Setting individual breakpoints

To set a breakpoint, first place the cursor in the program line at which the execution of the program is to be halted. Then select the command

“Reverse breakpoint” in the “Debug” menu. At the left border of the window at the level of the program line, a red dot will be displayed. That shows you that a breakpoint has been assigned to a program line. You can also call up the option window for setting breakpoints by clicking the right mouse key.

You can set breakpoints at various places in the program text. Likewise, you can distribute breakpoints among various program files which you are working with.

Modifying individual breakpoints

You can modify breakpoints by enabling and disabling them. Once you have set breakpoints and wish to run a program without interruption, you do not have to delete your breakpoints, but can simply disable them. Then, when you are ready, you can simply enable them again. The positions of the individual breakpoints are maintained. This function speeds up debugging, as enabling and disabling breakpoints is faster than setting and removing them.

You can also modify a breakpoint by placing the cursor on it. Then select “Enable/disable” from the “Debug / Breakpoints” menu. The disabled breakpoint will now be displayed as a red dot with a white center instead of a solid red dot. You can re-enable a breakpoint by executing the same function a second time.

You can also call up an options window for modifications by clicking the right mouse key.

Deleting individual breakpoints

To delete a breakpoint, place the cursor in the specific program line. Select “Reverse breakpoint” in the “Debug” menu.

Display and edit all breakpoints

You can search for a breakpoint in the program by scrolling through the program text and placing the cursor at the proper place.

A more comfortable alternative, though, consists of calling up a list of all breakpoints. You do this by clicking on “Edit” in the “Debug / Breakpoints” menu. Breakpoints can be enabled and/or disabled in this list and directly targeted and removed as well.

This function is only active when breakpoints have already been set. All breakpoints existing in the project are displayed in the list. By displaying the file name of the program file and the line number, it shows you where breakpoints exist.

This display is especially helpful with larger programs, as you can access all breakpoints directly from the central list.

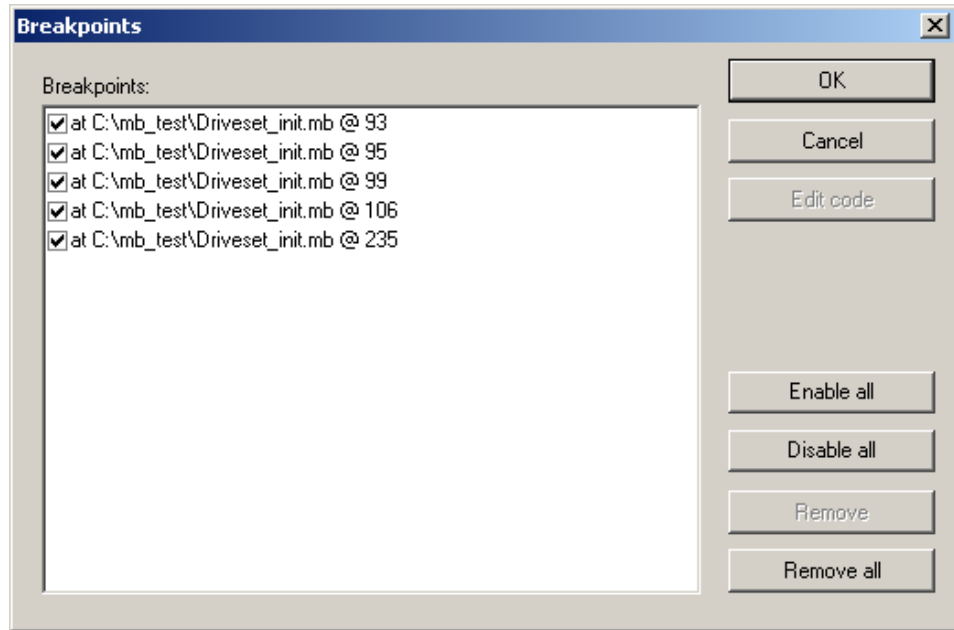


Fig. 30 List of all breakpoints

Enabling / disabling breakpoints

The following functions are available:

The checkmark box at the head of the line serves to enable and/or disable each specific breakpoint. A box with a checkmark shows that a breakpoint is enabled. You make changes by clicking on the box.

Edit code

Once you have marked a breakpoint in the list, you can jump directly to the program line containing that breakpoint by clicking on the “Edit code” button.

Enable all

All breakpoints are enabled

Disable all

All breakpoints are disabled

Remove

A marked breakpoint is deleted from the list

Remove all

All breakpoints in the project are deleted

10 Menus of the IDE

10.1 Overview

There are eleven pull-down menus available in the menu bar.

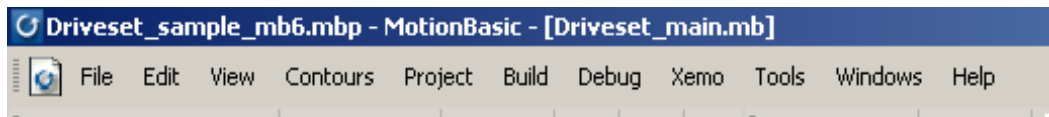


Fig. 31 The menu bar

Descriptions of the individual functions, see the following chapters.

Menu	Chapter	from page
File	10.2	45
Edit	10.3	49
View	10.4	50
Contours	10.5	52
Project	10.6	59
Build	10.7	60
Debug	10.8	61
Xemo	10.9	62
Tools	10.10	65
Windows	10.11	68
Help	10.12	69

10.2 “File” menu

New project

Create a new project file. After you call up this command, a dialog window appears in which you enter the name of the project file and the place it is to be stored. At least one program file must be assigned to a project file. Assigning a program file is described in the “Add to project” menu section below.

New file

Create a new MotionBasic program file. This command can be executed whether or not a project is open. A new program file is NOT assigned to a project, even when one is open. Assignment must be carried out explicitly via the “Add to project” menu.

Open project

Opens a project file. A dialog area appears in which you can determine the kind of file. The default comprises MotionBasic project files with the suffix “*.mbp”. In this case, only project files are displayed.

Only one project can be open at a time.

Open file

Open an existing MotionBasic program file (.mb), a MotionBasic contour (.mbg) or a file of a different format (*.*)

Program files can be opened regardless of whether any project file or the corresponding project file is open.

Files in a format differing from the MotionBasic format: The IDE calls up the program associated with the file. This program opens the file in a separate window.

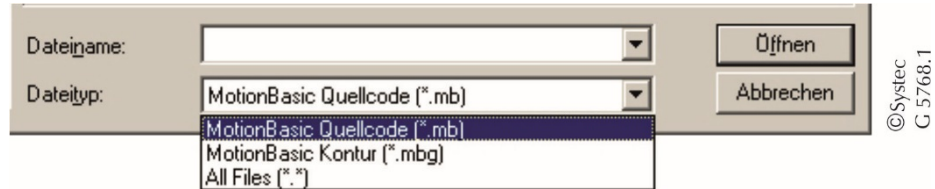


Fig. 32 Selection of the file format to be opened

Add to project

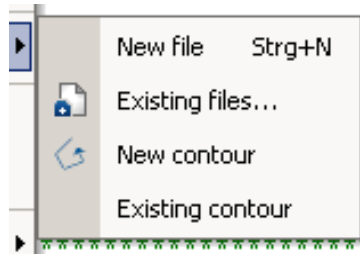


Fig. 33 Subitem "Add to project"

New file

Create a new file and assign it to the project

Existing files

Here you can select an existing MotionBasic file (or files) and assign it (them) to the project

New contour

With this menu command, you can convert a "DXF" file into a MotionBasic graphic file and assign it to the project. The name of the new "mbg" file must be assigned by the user.

Existing contour

An existing MotionBasic graphic file (*.mbg) can be incorporated into the project.

Close project

An open project file is closed

Close file

A program file is closed.

If the program file belongs to a project and the project is open too, the file window is closed. The program file continues to be assigned to the

project, though, which is shown by the display of the file name in the workspace bar.

Save

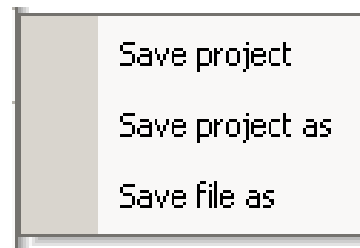


Fig. 34 Subitem "Save"

Save project

Saves a project file. The project file must have at least one program file assigned to it (see above under "New project").

If you execute changes in program files of an open project, the program files are also saved by the command "Save project" as well.

Save project as

An open project file can be saved under a different name

Save file as

A MotionBasic program file can be saved under a different name.

If a program file which is assigned to a project is saved under a different name, this new name will be displayed in the file window immediately. The file does NOT, however, automatically belong to any project, even if one is open at the moment. This is shown by the fact that the new file name does not appear in the workspace bar. If you wish to assign the file, you must do that via the "Add to project" menu.

Save file

Save a MotionBasic program file

Saving is done independently of a project file

Save all

All open MotionBasic program files are saved

Page setup

This and the following commands refer to the printing of a MotionBasic file.

Set the paper format and page borders

Print preview

Previews the program file to be printed

Print

Print a MotionBasic program file

Print setup

Paper format, configure the specific characteristics of the printer connected to the computer

Recent projects

List of the most recently opened projects. The number of the projects to be listed can be set under “Extras / options / workspace” menu.

Recent files

List of the most recently opened files. The number of the files to be listed can be set under “Extras / options / workspace” menu.

Exit

Exit MotionBasic 6.

10.3 “Edit” menu

Undo

Refers to the program editor. The most recent change can be reversed.

Redo

Can re-implement the action(s) just reversed with “Undo.”

Cut

A selected section of text in the program editor can be cut and stored in a temporary memory.

Copy

A selected section of text in the program editor can be stored in a temporary memory.

Paste

A section of cut or saved text can be inserted at the cursor’s location

Delete

Delete refers to several areas of the development environment.

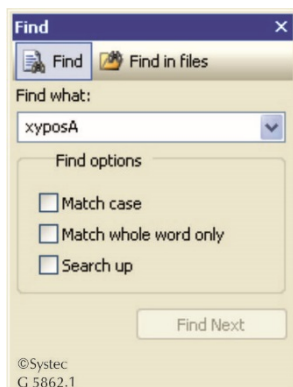
Selected sections of text in the program editor are deleted.

Selected variables in the “Watchpoint” windows are deleted from the list.

Program files which have been assigned to a project can be deleted from the project file. Select the file which you wish to remove from the project and select “Delete.” That deletes the connection between the program and project files.

Select all

Refers to the program editor and selects the entire text of the program file active at the moment.



Find (Ctrl F)

Opens a window with search function for the program files on the left. It offers various options which enable you to narrow your search. The search direction – to the end or the beginning of the document – as well as the restriction to whole words and case sensitivity can be set.

Find next (F3)

Renewed search for the character, sequence of characters and/or words defined under “Find.” The search is carried out to the end of the document.

Find previous (Shift F3)

As under “Find next” but in the direction of the beginning of the document.

Replace (Ctrl H)

Individual characters, character sequences and/or words can be located and exchanged for others.

Find in files (Ctrl Shift F)

Search function for all program files of the current project. Various options as with the ordinary search. The results window (tab: search) shows a list of the file paths and lines containing the search term. By double clicking on the line with the located file, you can open the corresponding file and the cursor is positioned at the location point.

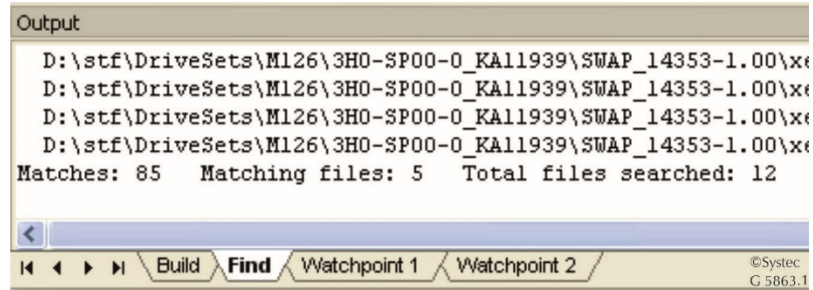


Fig. 35 Example of the results of a search conducted in all files of the project

Go to

You can jump to a specific line number. The line numbers are displayed in the status bar.

Properties

Characteristics of the active program file. This includes the file name and date of the last change. You can add a commentary to the file in this window.

10.4 “View” menu

In the “View” menu, you can select which MotionBasic windows and symbol bars you wish to have displayed.

MotionBasic 6 contains the following windows:

Output (Output bar)

Lower right window. Contains the subwindows “Output”, “Watchpoint 1” and “Watchpoint 2”.

Status bar

Lower right line in the MotionBasic application window. In various windows, the line and column numbers, for example, are displayed, as well as MotionBasic status descriptions such as “Edit”, “Execute” or “Stopped”.

Workspace

Left window with a listing of the program files which belong to the open project file.

Find

Left window with search function within the current file and the whole project.



Remark

The window of the search function can be changed from a window docked to the workspace into a freely movable window by dragging with the mouse.

Standard toolbar



Fig. 36 Display of the toolbar with symbols for the standard Windows functions

Build toolbar

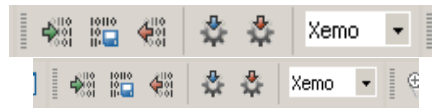


Fig. 37 Display of the toolbar for coding, downloading into ROM or RAM

Debug toolbar



Fig. 38 Display of the toolbar for debugging

Graphics toolbar



Fig. 39 Display of the toolbar for the graphic functions

Customize

Here you can implement your personal customization of the MotionBasic applications



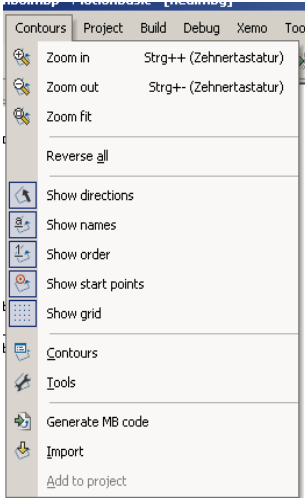
Tip

You can change the ordering of a toolbar by seizing and moving the bar with the mouse at the left (red marking in Fig. 40).



Fig. 40 Point of action for changing the position of a toolbar; the example given here is the toolbar "Debug"

10.5 "Contours" menu



In the "Contours" menu, all functions are enabled which you need to edit and transpose your contours into MotionBasic code.

The menu commands are only active when the file presently displayed is a contour file with the suffix "mbg" (MotionBasic graphic).

Zoom in

The contour display is enlarged

Zoom out

The contour display is reduced

Zoom fit

Returns to the contour's initial size

Reverse all

The trajectory directions of all contours are reversed

Show directions

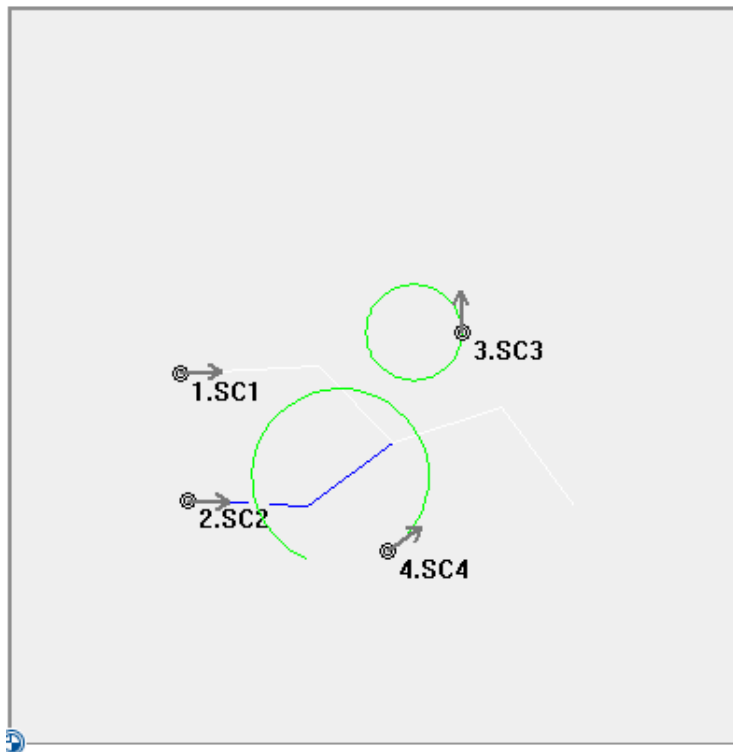


Fig. 41 Display of contours WITH directional arrows

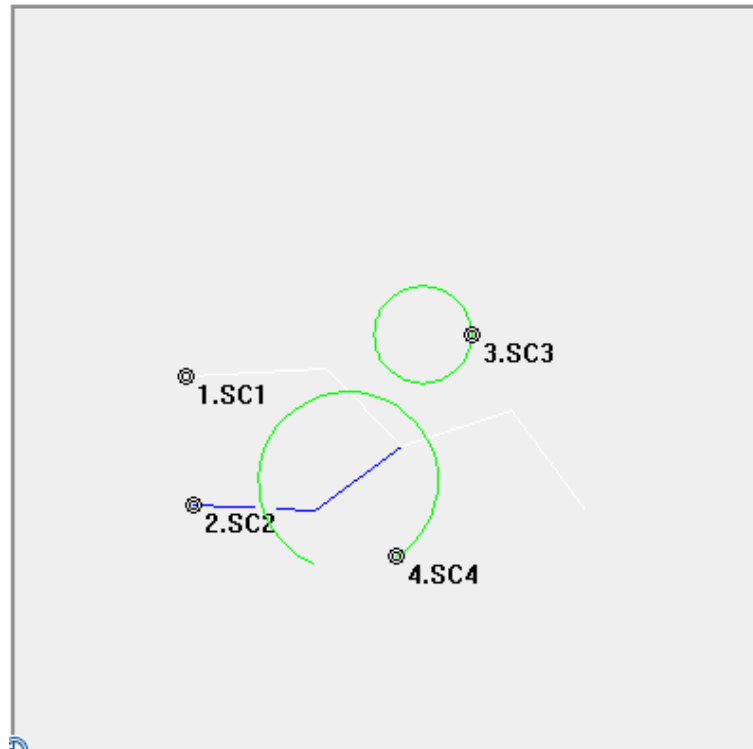


Fig. 42 Display of contours WITHOUT directional arrows

Show names

With this menu command, the display of the names of individual contour sections can be activated and/or deactivated.

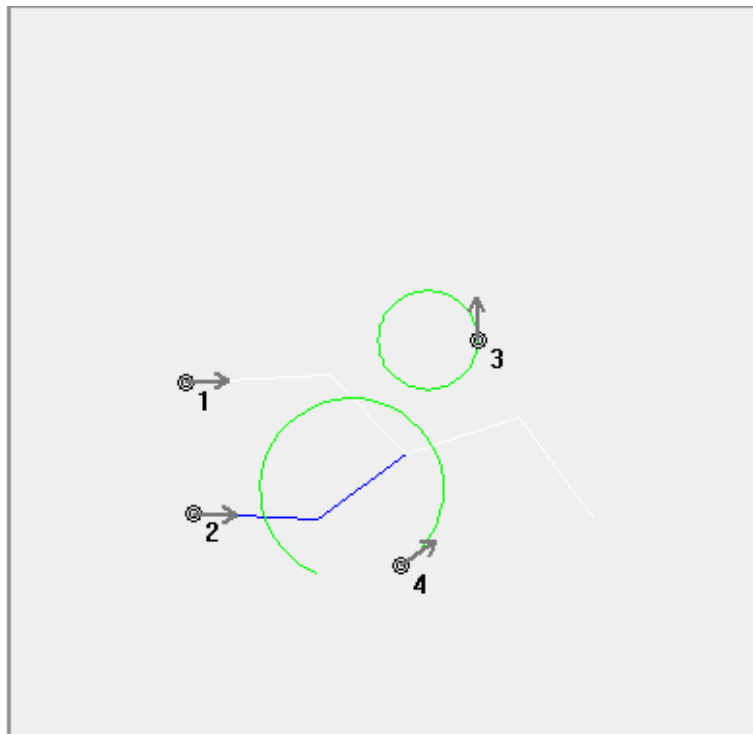


Fig. 43 Display of names is deactivated

Show order

With this option, the order of the individual contours can be activated and/or deactivated.

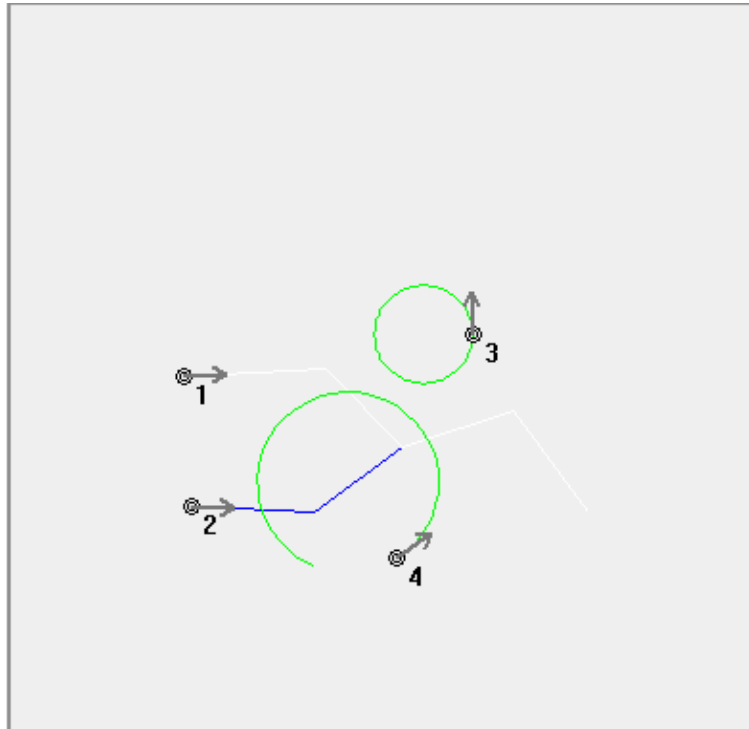


Fig. 44 This example shows the order in which the contours are executed

Show grid

Here the display of grid dots behind the graphic can be activated and/or deactivated. These grid dots help orientation.

Editing functions

Contours

With this menu command you can initiate a detailed modification of a contour:

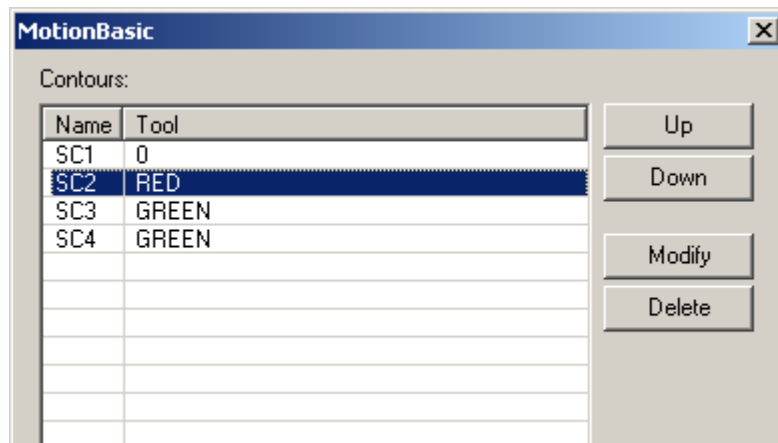


Fig. 45 Window for rearranging the order of the individual contours

If you mark a single contour section, you can move this section in the order with the "Up" and/or "Down" keys as you wish.

With “Delete,” this contour section can be deleted.

With “Modify,” you access the contour’s properties (see Fig. 46).

You can move the contour’s start point, change the kind of tool selected and, under “Extras,” influence further options of the various tools. (See “Contour” menu, “Tools”).

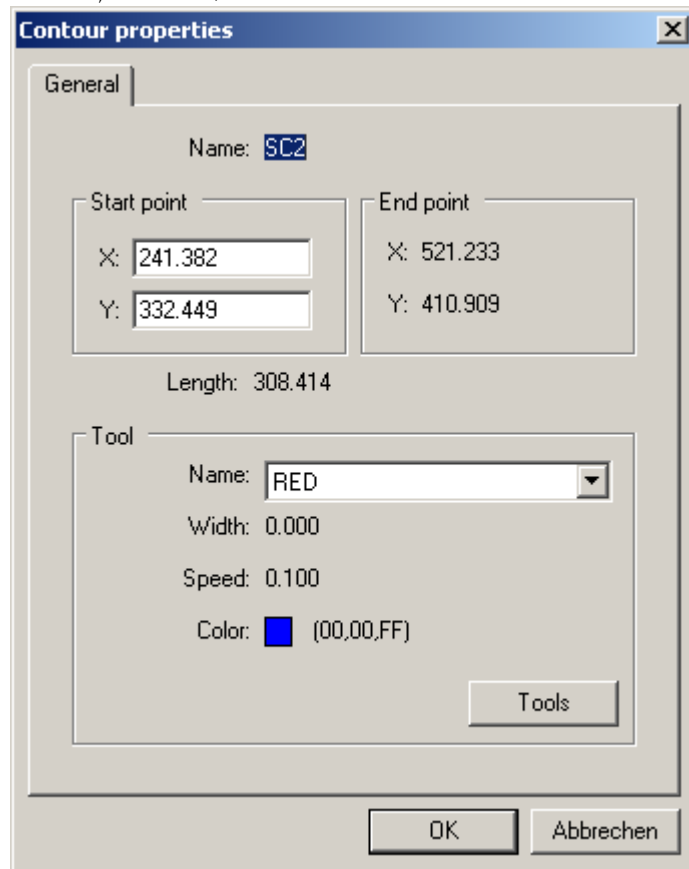


Fig. 46 Mask for modifying a contour

You can move the contour’s start point, change the tool selected and influence further options of the various tools. (“Contour properties” menu, “Tools”).

Tools

With this menu command, you can modify the tools which belong to the contour (see Fig. 47).

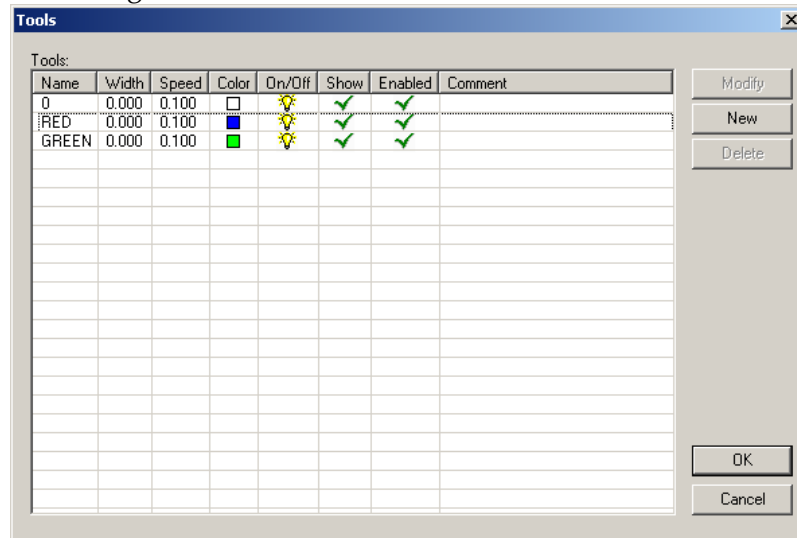


Fig. 47 Overview showing the status of the tools assigned to the contours

In this window, you can activate/deactivate the illustration of all contour sections of the tools displayed by clicking on the respective light bulb.



Remark

If you remove the checkmark for a kind of tool in the “Enabled” column, the relevant tool is deactivated. This means that a contour section for which this tool was selected will not be executed. Consequently, upon generation of the MotionBasic code, no code is generated for this specific contour section.

You can create a new tool with the button “New.” An existing tool can be deleted with the “Delete” button, and you can make changes to a tool with “Modify” (see Fig. 48).

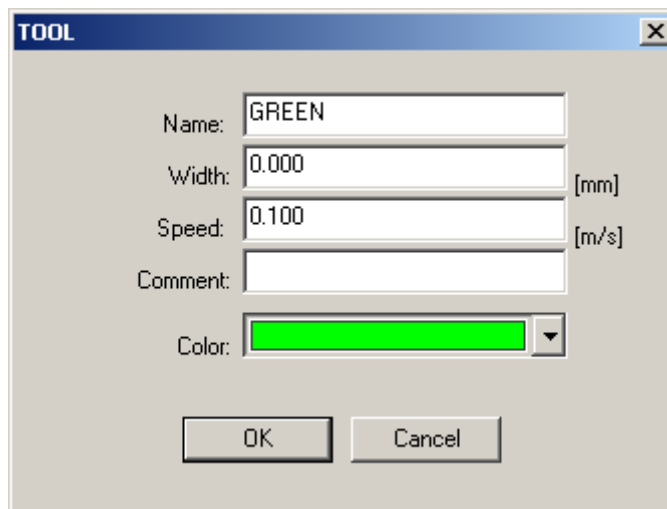


Fig. 48 Mask for modifying a tool

Here you can select the display color for this tool. In addition, you can modify the tool's name, width and speed for the selected tool.

Example

If you have a milling machine, for example, you could assign various diameters of the milling tools to the contours. You could also assign tools so that one contour is for rough-machining and another for finishing.

Integration into an MB project

Generate MB code

With "Generate MB code," you can generate the MotionBasic code which this contour will then follow in the Xemo.

The advice on p. 56 is to be noted.

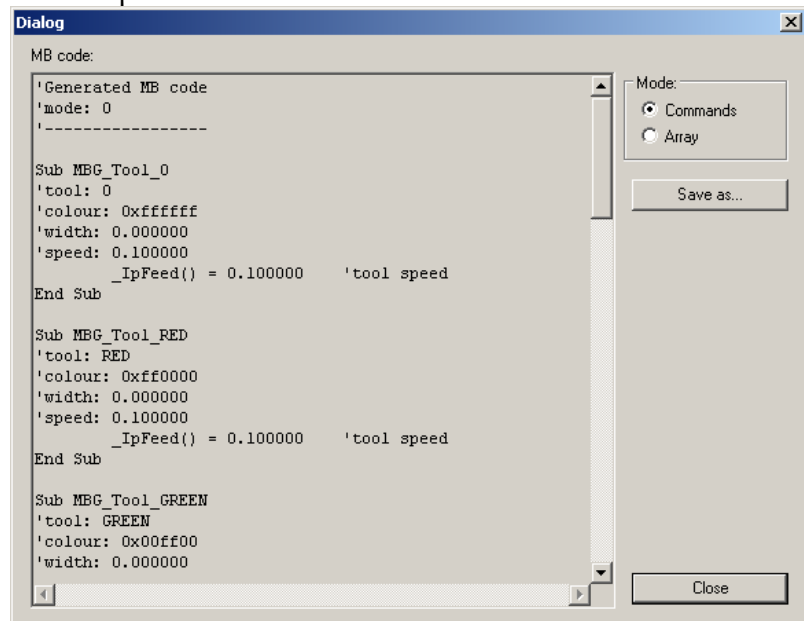


Fig. 49 Generation of MB code from the contours

The code can take the form of a motion command (lin, circle, arc, ...) or in the form of an array which the programmer can then use in the rest of his program code. By switching from "Mode" to "Array," you generate the code in the form of an array.

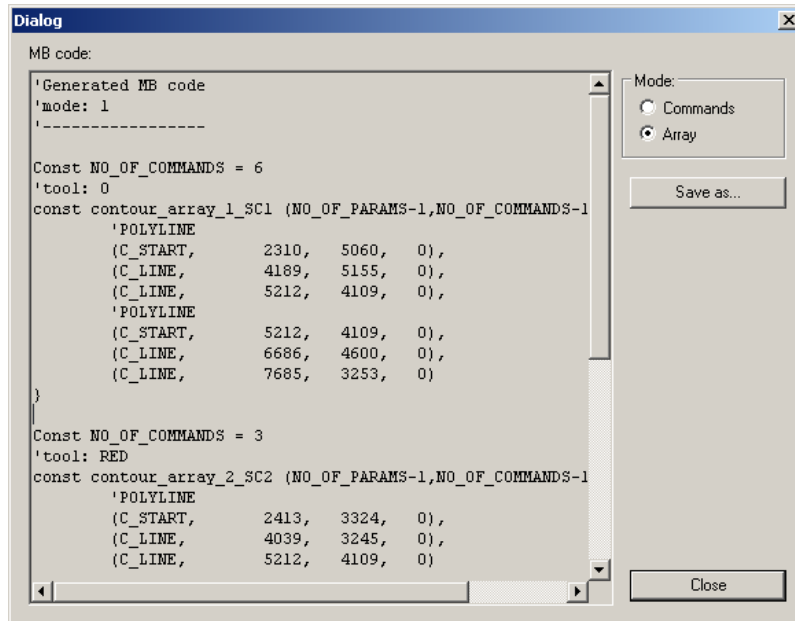


Fig. 50 The same coding as in Fig. 58: Generation in arrays

If you call up “Save as”, then you can save the displayed code in the MotionBasic file with the suffix “.mb”.

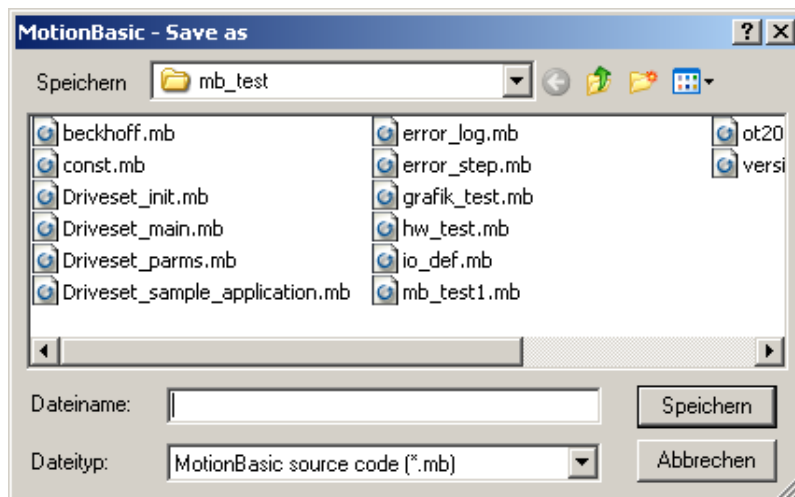
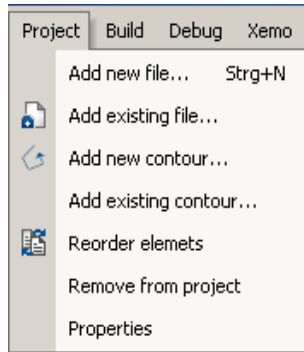


Fig. 51 Saving the generated code

This MotionBasic file will be added to the open project.

10.6 “Project” menu



Add new file

A new program file can be opened and immediately incorporated into the project file open at the moment.

Add existing file

Existing program files can be assigned to project files open at that moment.

Add new contour

The same function as in the “File / Add to project / New contour” menu.

Add existing contour

The same function as in the “File / Add to project / Existing contour” menu.

Reorder files

A window appears in which you can reorder the files. As an alternative, mark a file in the file list, click on the right mouse key and select the function from the context menu.

Remove from project

If a file is marked in the workspace, then it can be excluded from the project with this command.

Properties

Properties of the active project file. Entry of the project name and the file directory. In a blank space, you can add a commentary to the project.

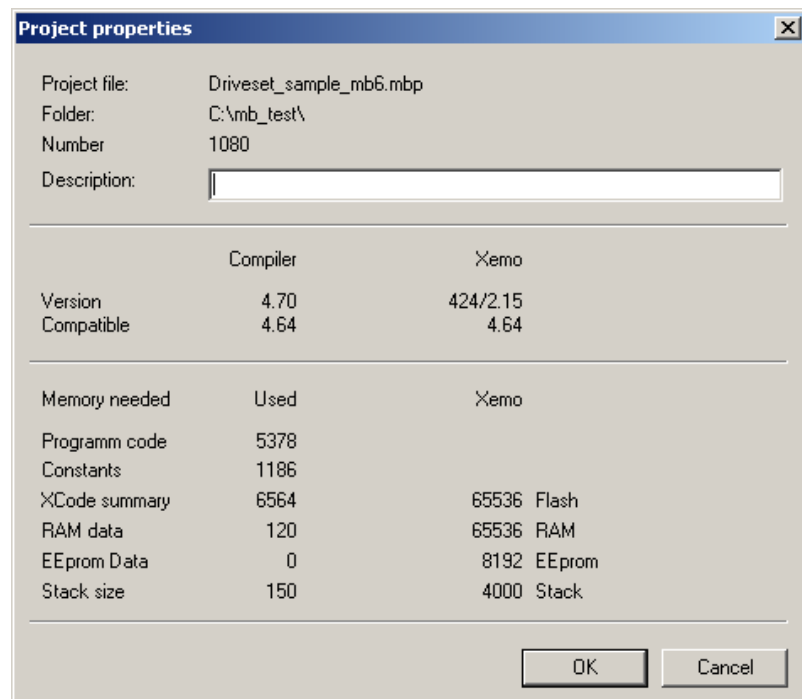
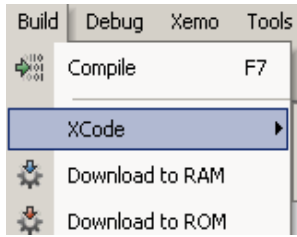


Fig. 52 Properties of the open project

If the project has been compiled since the last change, this will display how much storage memory the program, the RAM variables, the EEPROM variables and the stack need.

If a controller is connected, the amount of storage space available in the Xemo will also be displayed here.

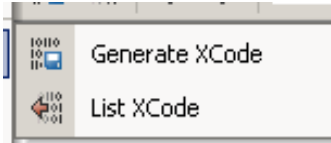
10.7 “Build” menu



Compile

A MotionBasic project is translated into instructions (Xemo-Code = XCode) which can be read by machines and thus by the controller.

XCode



Generate XCode

The instructions (XCode = Xemo-Code) which are generated during compilation of a MotionBasic program can be written into an external file with this command. After this command is called up, a context window appears in which you can determine the file name and storage location for the XCode file.

An XCode file is a non-readable binary file. It enables the transmission of translated MotionBasic programs in which the recipient has no access to the source code.

With the command “Download to ROM” from the “Xemo” menu, these files can be transferred to and stored in the controller.

List XCode

“List XCode” generates a file from a MotionBasic project. This file lists all MotionBasic instructions contained in the project. The specific XCode of each instruction is displayed with its corresponding line number.

This file is useful when a Xemo controller running independently of a PC announces an error. Together with the error message, the line number of the XCode is displayed in which the error has occurred. With help of the list, you can track down the MotionBasic instruction in which the error has occurred.

Download to RAM

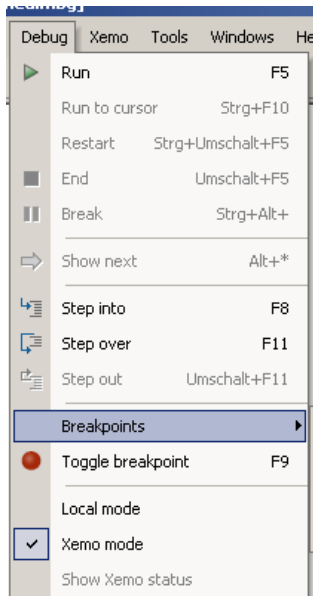
The MotionBasic program is loaded into the controller’s working memory (RAM). There it is stored temporarily. (When the controller is turned off, the program is erased.) The program can be started from the connected PC. In this operational mode, debugging the program is possible.

For this function, the Xemo controller’s operation switch must be turned to either position “F” or position “0”.

Download to ROM

A MotionBasic program is stored permanently in the Xemo controller. If the controller’s operation switch is set to position “0”, the program will start automatically when the controller is turned on provided, of course, it has been programmed to do so.

10.8 “Debug” menu



Run (F5)

You start a MotionBasic program with “Run.” If you have already compiled and transferred it to the controller, it will start immediately. If not, then “Run” will initiate the compilation and transfer.

The complete program will be run. If breakpoints have been set, the program will be executed up to each one of these and stopped in sequence. Continuation is done with “Run” or “Step into.”

Run to cursor

The program is run until it reaches the place where the cursor was set.

Restart

The program will be started again.

End

Execution of the program is stopped

Break

If a program is running, it can be interrupted by “Break.” The position in the program is indicated by an arrow at the left edge of the window. A program paused in this way can then be restarted with “Run” or in “Step into”.

Show next

With this command, you can display the next instruction due to be executed. This is helpful when you have several files open and want to check on something. Select “Show next” and the cursor will jump to the file which contains the instruction.

Step into (F8)

With “Step into” you execute a program step-by-step. You can work your way forward program line by program line and observe the program’s execution.

Step over (F11)

Basically, the program is carried out in the same way as in the “Step into” mode. However, the process does not branch off into a called-up subprogram (procedure or function), but instead the entire subprogram is processed in sequence.

Step out (Shift-F11)

Jump back into the called-up program from a subprogram.

Breakpoints

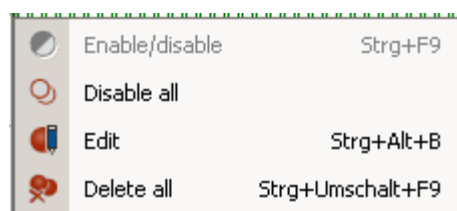


Fig. 53 Submenu for the breakpoints

Enable/disable

Activates and/or deactivates a set breakpoint. If you wish to avoid the program stopping at a breakpoint, you can disable that breakpoint. It remains set at the same position, but has no effect.

Disable all

Disables all set breakpoints.

Edit

A list of all set breakpoints is displayed. These can be edited here (deleted, enabled, disabled).

Delete all

All set breakpoints are deleted.

Reverse breakpoints (F9)

Set a breakpoint, delete a set breakpoint. Place the cursor at the corresponding line in the program and select "Reverse breakpoints."

Local operating mode

If this command is set, the work is being carried out locally. In that case, a program will not be downloaded to and started in the Xemo, but instead will remain in and be started from the PC. In this mode, you can run tests without actually needing a Xemo.

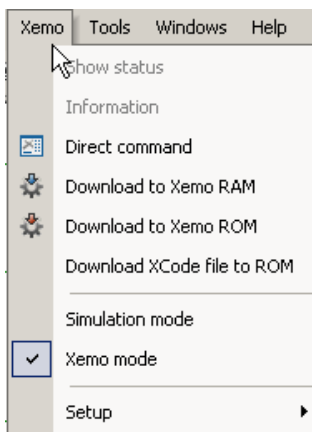
Xemo operating mode

If this command is set, work is carried out in the Xemo and not locally.

Xemo status

Not yet implemented.

10.9 "Xemo" menu



Show status

Not yet implemented

Information

Not yet implemented

Direct command

With this menu command, you can start the "Xemo!Go" program. Here you can work directly with the controller to test how the individual commands affect the Xemo.

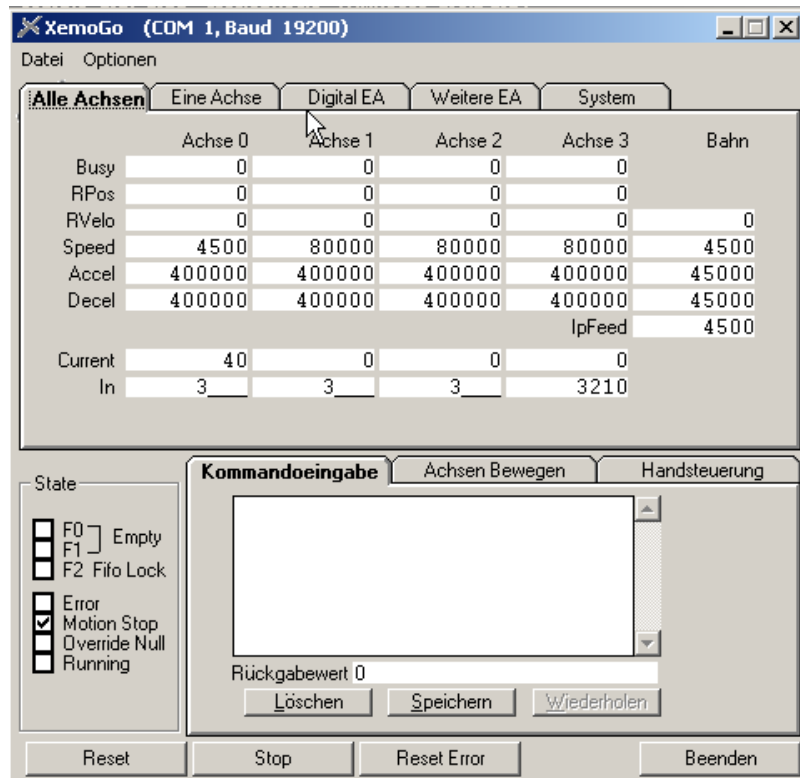


Fig. 54 Xemo!Go user surface

Download to Xemo RAM

Please see menu "Build / Download to RAM", p. 60

Download to Xemo ROM

Please see menu "Build / Download to ROM", p. 60

Download XCode file to ROM

Download a previously generated XCode file to the Xemo ROM (see "Build / XCode / Generate XCode file")

Simulation mode

This menu command is active when the local operating mode is being used.

Xemo mode

This menu command is active when the operating mode is NOT set to local.

Setup

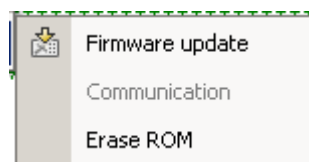


Fig. 55 Options in the submenu "Setup"

Firmware update

With this command, you can overwrite the Xemo's operating system. The following window then appears:

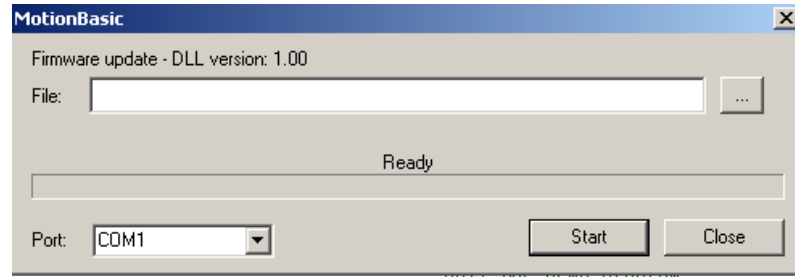


Fig. 56 Selection of the new firmware



Remark

You should use this program command only after consulting with SYSTEMEC, as it could destroy the remote control of your operating system.



Remark

For Xemo controllers with integrated Ethernet interface, you run a firmware update using the program XemoUpdate.exe. It is with the appropriate version of the MotionBasic IDE, see Cape. 5.1, automatically installed on your computer. You can find this program in Windows/Start/Systemec/MotionBasic 6.5.6 or in the applications folder on C: /, where you have installed MotionBasic.

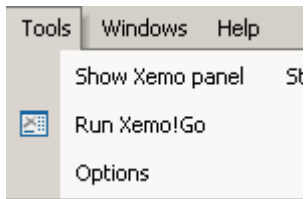
Communication

Not yet implemented

Erase ROM

The permanent storage memory of the Xemo controller is erased.

10.10 “Tools” menu



Via “Show Xemo panel” in the “Tools” menu, you can call up an operating panel simulation or Xemo!Go and select presettings via “Options”.

Xemo panel



Fig. 57 Virtual display for testing the programming; in the upper left corner, you can choose between the display for Xemo R and Xemo B.

All of the user interface functions, i.e. input via the keys, show text on the display, or switch the LEDs can be simulated. The controller must not be connected, but the “local” mode must be selected. (Extras / Options / Interface)

Xemo!Go

Calls up the diagnostic tool Xemo!Go. For this purpose, the controller must be connected to your computer and be turned on.

Fig. 54 shows the Xemo!Go user interface (Chap. 10.9), see Chap. 11, p. 71, for an introduction and the Xemo!Go manual [SYSTEMEC775] for a detailed description.

Options
“Workspace” option

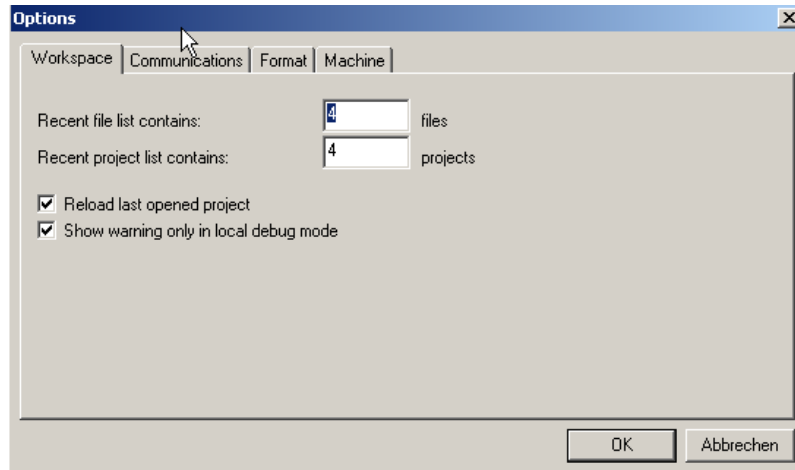


Fig. 58 Settings of the workspace

The list of the recently opened project and/or program files can be viewed in the “File” menu. In the dialog field “Workspace,” you can enter the number of files which you wish to have displayed.

“Communications” option

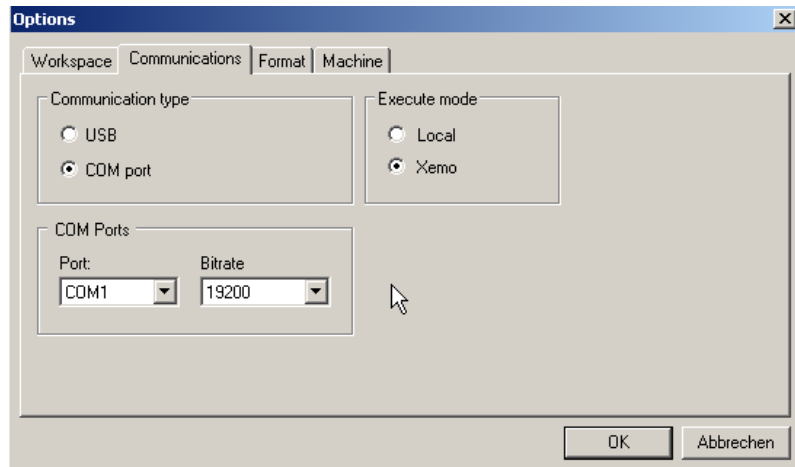


Fig. 59 Selection and setting of the chosen communication type

In the dialog field “Communications,” you can click on the type of communications (i.e. interface) to which the Xemo controller is to be attached. The baud rate for the RS232 interface is defaulted at 19,200 bauds. You should also set this value on your PC. Via the selection in “Execute mode,” you can determine whether you wish to test a Motion-Basic project locally on the PC – “Local” setting – or whether you wish to make a connection to the controller (PC and controller must first be connected) – “Xemo” setting.

Also see chap. 8.1, p. 33.

“Format” option

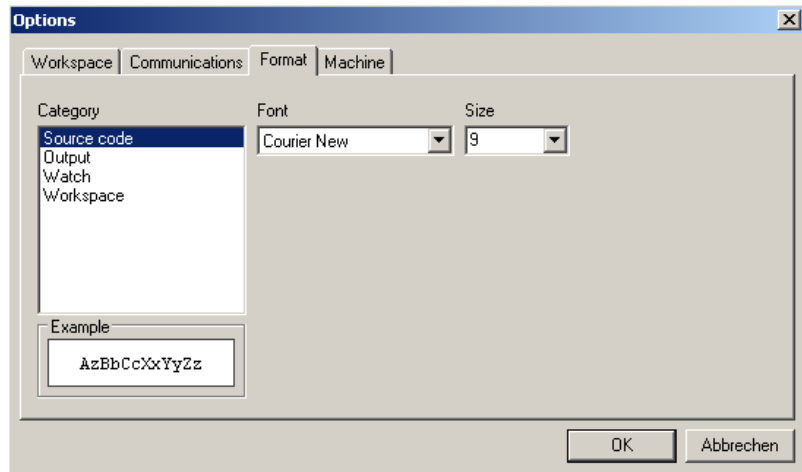


Fig. 60 Sheet for setting the font

With the “Format” option, you can set the font kind and size for the individual MotionBasic windows.

“Machine” option

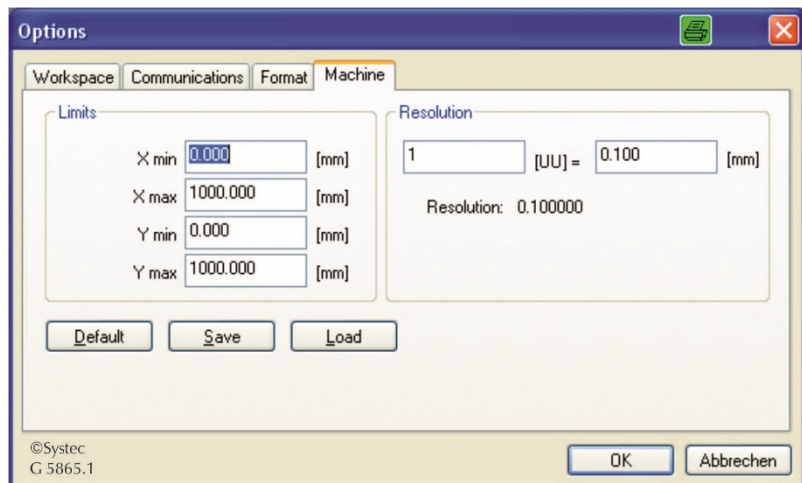
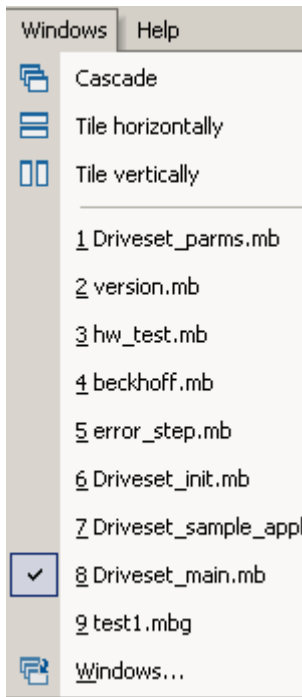


Fig. 61 Sheet regarding the size of the working plane and the resolution of your machine

Via the “machine” option, you can specify the size of the working plane, “limits”, and the resolution which you accept for your system (also see [SYSTEC717]).

10.11 “Windows” menu



With the menu “Windows“, you can determine how program files are located on the monitor. In addition, all open program files are listed.

Cascade

The windows of the individual program files are displayed staggered behind one another.

Tile horizontally

The windows of the individual programs are placed one above the other.

Tile vertically

The windows of the individual programs are placed one next to the other.

Windows...

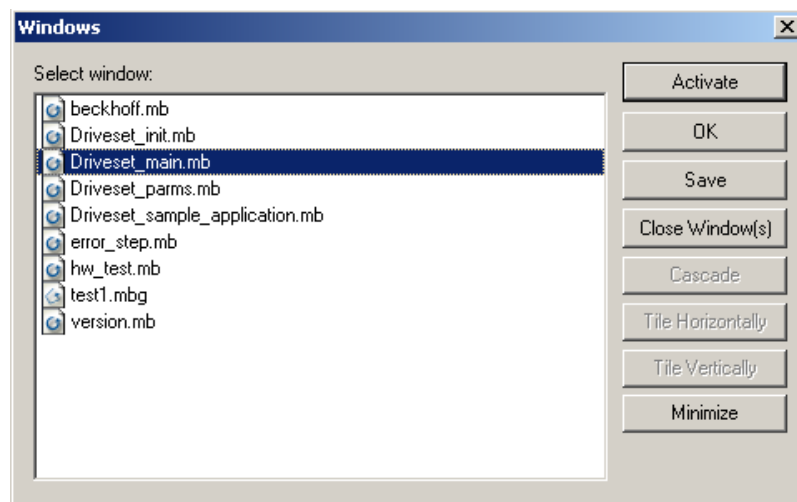
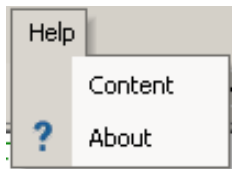


Fig. 62 Overview of all open windows

A dialog window opens with a list of all open program files. The files can be selected and saved and/or closed.

For these functions, you can mark one or more files at a time.

10.12 “Help” menu



Content

Here you will find online help for MotionBasic 6. You can target specific questions and problems.

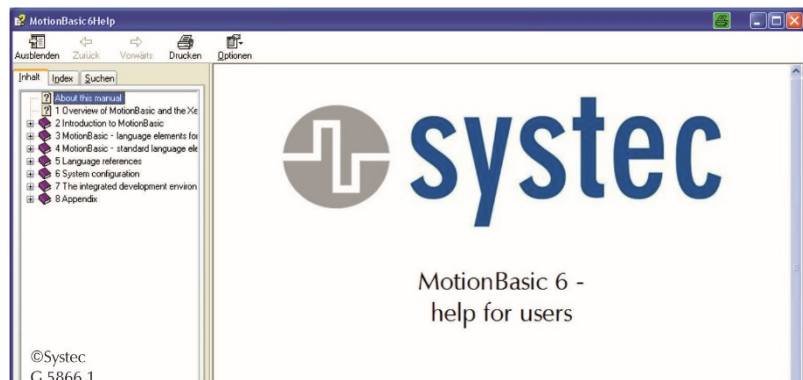


Fig. 63 Welcome page of the MotionBasic help

You can also call this help up by marking text in your code and then hitting the function key “F1” to call up online help.

If you have marked “main,” for example, and then hit “F1,” you will see the following window on your screen.

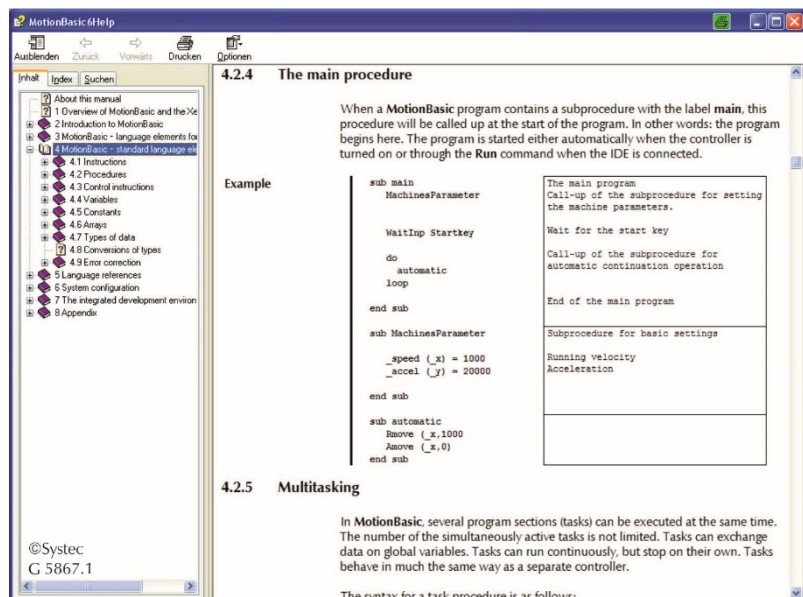


Fig. 64 Call-up of a help entry regarding the marked word by pressing “F1”, here the example of the word “main”

About

By clicking on this command, you get a display containing the present software numbers of your IDE, compiler, XemoDLL and Xemo!Go.

Whenever contacting Systemec about problems, please always have these numbers available. In case of errors, it is even better if you copy this window and send it Systemec via Email.



Fig. 65 Display of the versioning

11 Online operation with Xemo!Go

Manual operation with Xemo!Go

In offline operation, you write your MotionBasic programs on the PC, transfer these to the Xemo controller and then start the controller up. In online operation with Xemo!Go, you enter individual commands, which are then instantly carried out.

Likewise, you can change parameters, such as e.g. the velocity of an axis, and immediately check the effect. For test purposes, initializing or familiarization, this mode of operation is very helpful. However, not all MotionBasic commandos are available during online operation.

Xemo!Go and MotionBasic

Xemo!Go can be called up directly from within MotionBasic by a click on the Xemo!Go icon. It does, however, function independently, so it does not matter if MotionBasic is active or not. To call Xemo!Go up, though, a controller must be connected to the PC. Without a controller, calling up the program makes no sense: an error message is displayed and Xemo!Go cannot be opened.

11.1 Xemo!Go's application window

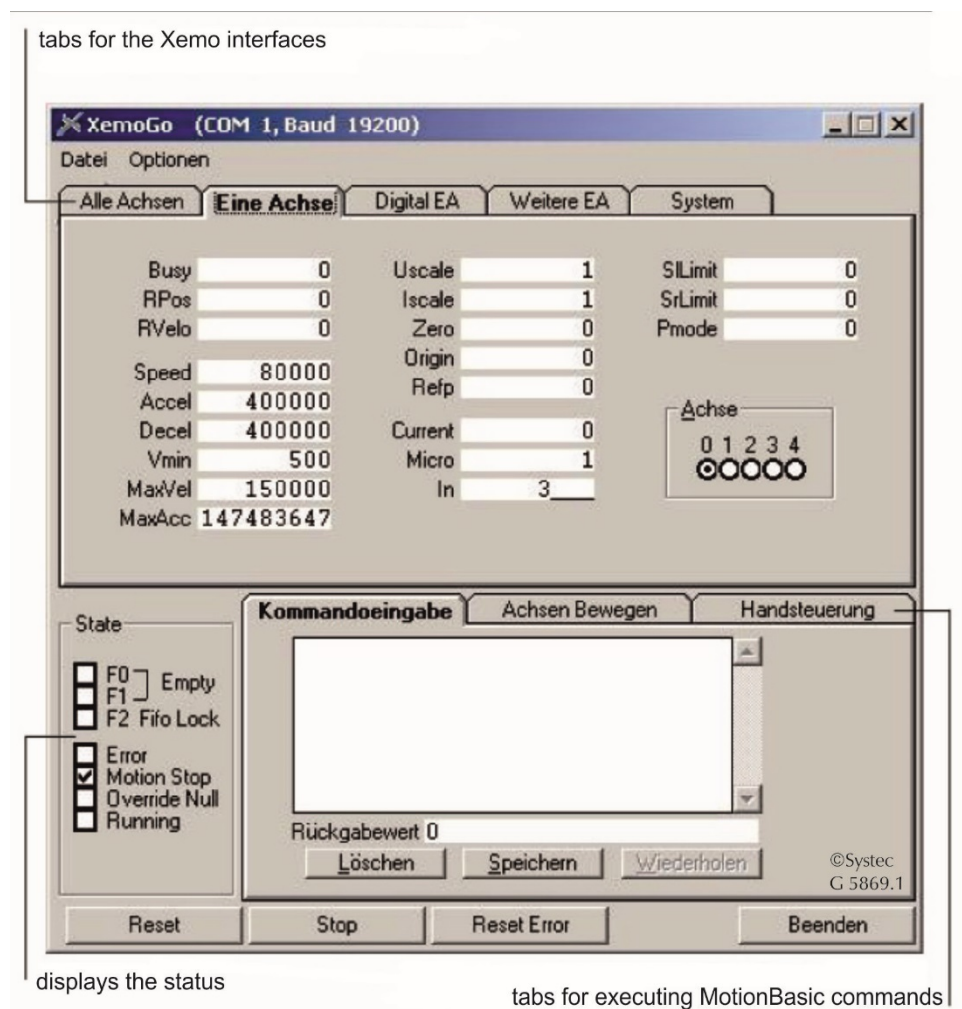


Fig. 66 Xemo!Go's application window

Xemo!Go’s application window is divided into five tabbed index cards at the top. Each index card has a specific functional area of the controller assigned to it.

The individual functional areas are subdivided into axes, digital inputs and outputs, system data and expansion options.

When you call up Xemo!Go, the controller’s status is read into the computer and displayed in the application window. If the controller is in operation and programs are being executed at that moment, the data displayed are those which were valid at the point in time when you called up Xemo!Go.

11.2 Digital inputs and outputs

Digital inputs and outputs in Xemo!Go

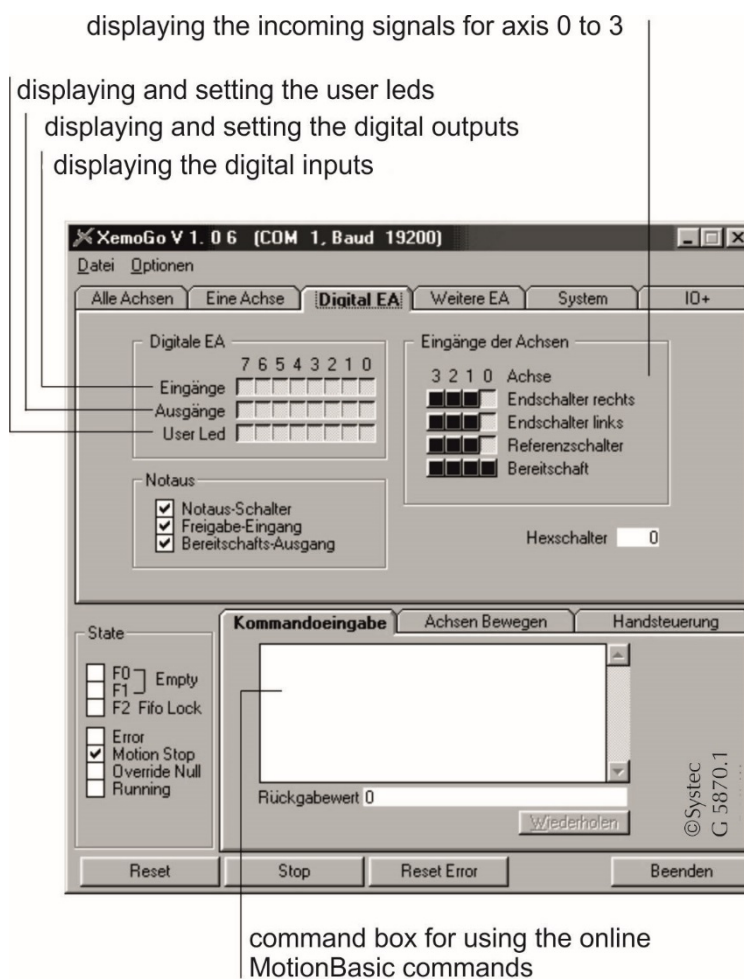


Fig. 67 Index card – digital inputs and outputs

Digital EA

In the upper left area of the “Digital EA” index card, you will see the field “Digital EA” with eight check-boxes for the inputs and outputs as well as the user LEDs. The numbering of those inputs and outputs corresponds to the controller’s socket boards X17 and X18.

You can now set an output by clicking on a check-box with the mouse. You can tell that the output has been set when the LED of that specific output lights up.

The display of a check-box for an input is inverted as soon as an input signal is present.

You can turn the user LEDs on and off by clicking on their check-boxes just as you do with those of the outputs.

Axis inputs

In the area “Axis inputs”, the digital inputs of the individual axes are displayed. You can recognize there if contact with end- or reference switches has been made. Likewise, the ready signal of each specific axis is shown.

11.3 Axis settings and running commands

Running an axis with Xemo!Go

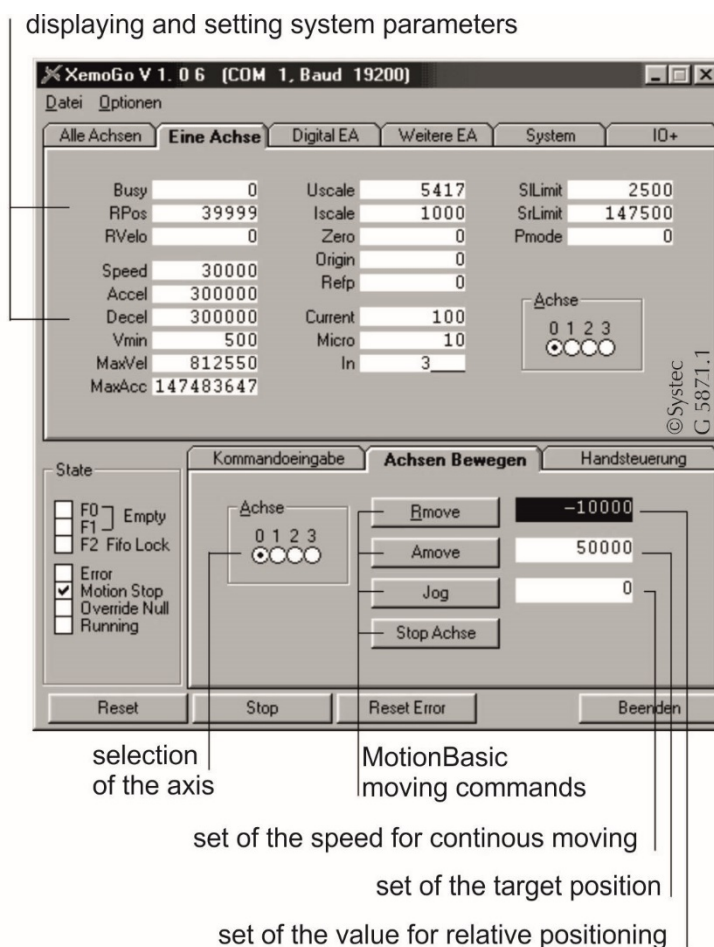


Fig. 68 Index card “One axis”

System parameters

In the upper area of the index card “One axis,” you can see the values of the system parameters which have been allocated to an axis.

Xemo!Go reads these values from the controller.

If you have initialized the controller through your MotionBasic program, you will see in Xemo!Go the value allocations executed by the program.

If the controller is not initialized through a MotionBasic program, the default settings are used for the system parameters. These are displayed in Xemo!Go.

You can change the system parameters in Xemo!Go simply by overwriting the values in the individual fields.

Running commands Three index cards are located in the lowest area of the Xemo!Go window: Command entry, Move axes, and Hand control. Buttons for the MotionBasic process commands Rmove, Amove and Jog have been placed on the index card "Move axes."

Rmove means relative trajectory, Amove for absolute positioning and Jog for continuous motion of an axis. You do not have to enter these commands; just insert the proper values.

In the blank space to the right of the button Rmove, enter the value for a relative trajectory. After you click on the Rmove button, the axis will move from its present position along exactly this trajectory.

In the blank space to the right of the "Amove" button, you enter the value for absolute positioning. After you click on the "Amove" button, the axis will move directly to that specific position.

In the field belonging to the button "Jog," you enter the velocity. After you click on the "Jog" button, the axis will move in continuous operation at this velocity. You can stop this motion by clicking on "Stop."

12 Appendix

12.1 Compatibility between MotionBasic 5 & MotionBasic 6

This chapter is intended for users who have used MotionBasic 5 – the version for the DOS operating system – to generate programs and/or who still use it today.

MotionBasic program files are compatible and, consequently, will run under MotionBasic 5 and/or 6. The project files, however, are not fully compatible.

With MotionBasic 5, you cannot call up project files generated with MotionBasic 6, although you can access the individual program files.

With MotionBasic 6, you can open MotionBasic 5 project files which can then be converted. During the conversion process, you will see an inquiry window through which you can confirm the conversion.

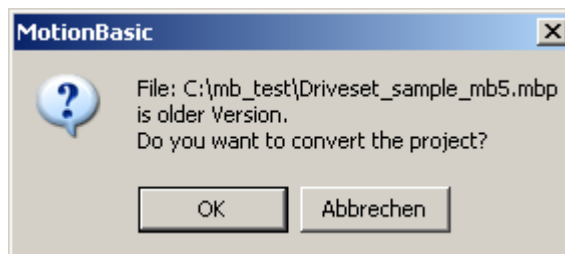


Fig. 69 When opening, enquiry if the project is to be converted

If you wish to leave the project file in its original version, simply save the converted file under a different name.

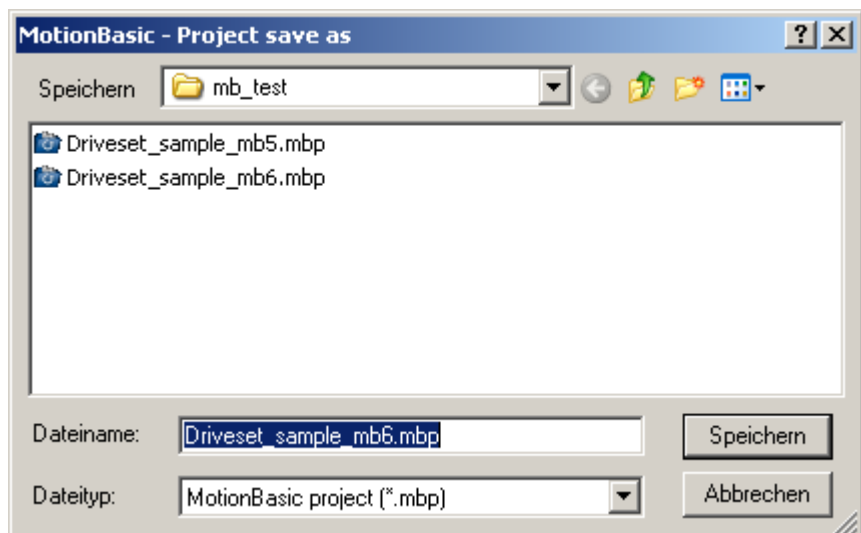


Fig. 70 Trick used to retain an old version of your project

12.2 Bibliography

- [SYSTEC591] Xemo DLL User Manual, Doc-No. 591-11
© Systemec GmbH, Münster, 2017
- [SYSTEC625] Xemo R/S Equipment Manual, Doc-No. 625-11
© Systemec GmbH, Münster, 2017
- [SYSTEC717] MotionBasic 6 Programming manual, Doc-No. 717-11
© Systemec GmbH, Münster, 2017
- [SYSTEC735] Xemo M Equipment Manual, Doc-No. 735-11
© Systemec GmbH, Münster, 2017
- [SYSTEC764] Xemo B-Panelsteuerung Gerätehandbuch, Doc-No. 764-12
© Systemec GmbH, Münster, 2017
- [SYSTEC767] LabView-Funktionsbibliothek, Doc-No. 767-22
© Systemec GmbH, Münster, 2017
- [SYSTEC772] Technologieoptionen, Doc-No. 772-12
© Systemec GmbH, Münster, 2017
- [SYSTEC775] Xemo!Go User Manual, Doc-No. 775-11
© Systemec GmbH, Münster, 2017
- [SYSTEC826] Structured troubleshooting, Doc-No. 826-41
© Systemec GmbH, Münster 2016
- [SYSTEC836] OT300 Gerätehandbuch, Doc-No. 836-12
© Systemec GmbH, Münster 2015
- [SYSTEC858] Xemo-Step Gerätehandbuch, Doc-No. 858-12
© Systemec GmbH, Münster, 2016

You will find these manuals in your manual folder, on your Systemec CD or also downloadable on www.systemec.de/service/downloads/?L=1 .

13 Index

About	69	Zoom in.....	52
Add		Zoom out.....	52
existing contour	59	Copy	49
existing file.....	59	CPU time.....	11
new contour	59	Customize application window.....	51
new file	59	Cut	49
Axis inputs.....	73	Data exchange.....	11
Baud rate	34, 66	Debug	61
Break.....	61	debugger	10
Breakpoints	42	Delete	49
Delete all	62	Digital EA.....	72
Disable all.....	62	DIN 66025	9
Edit.....	62	Direct command	62
Enable/disable.....	62	DLL	10
Reverse.....	62	Download in RAM.....	63
Build	60	Download in ROM.....	63
C 10		Download to RAM.....	60
Close		Download to ROM.....	60
file	46	End.....	61
project.....	46	Erase ROM	64
CNC.....	7	Error correction	
Communication	64	Error 31	14
Communications	33, 66	Ethernet.....	33
compile	14	Example.....	33
Compile	60	Existing	
Preprocessor	30	contour.....	46
Two-pass compiler.....	30	file	46
compiler.....	10	Exit	48
Compiler-Info.....	69	File	
Content (Help)	69	close	46
Contour		existing	46
existing.....	46	menu	45
new	46	new	45, 46
Contours	52	Properties	50
Edit.....	54	reorder.....	59
Generate MB code	57	Find.....	49, 51
Reverse all	52	in files.....	50
Show		next	49
directions.....	52	previous.....	49
grid	54	Firmware Update	63
names.....	53	Flash	15
order.....	54	Format.....	67
Tools.....	56	functions bibliography	10
Zoom fit	52	G-Code	9

Go to	50	files.....	48
HPGL code	10	projects.....	48
IDE.....	7	Remove from project.....	59
Information	62	Reorder file	59
ISO code.....	10	Replace	49
LabVIEW.....	10	Restart.....	61
Library.....	10	RS232	34
Linux.....	10	Run	61
Local operating mode	62, 63	to cursor	61
Menu	45	Running commands	74
Mode	See Operational mode switch	Runtime errors	37
Monitoring variables.....	41	Save	
Watchpoint	41	all47	
Multitasking.....	11	file	47
New		file as	47
contour	46	project.....	47
file.....	45, 46	project as.....	47
project.....	45	Select all.....	49
Offline		Setup.....	63
operation.....	10	Show	
programming.....	14	status Xemo	62
Online		Show next	61
commands	11	Status bar	50
programming.....	16	Step into (F8).....	61
Open		Step out (Shift F11).....	61
file.....	46	Step over (F11).....	61
project.....	45	Subprocedures	11
Operating mode		Fold	19
local	62, 63	Syntax errors	37
Xemo.....	62, 63	System parameters	73
Operational mode switch.....	13, 17, 34, 60	task.....	11
Options.....	66	Toolbar	
Output bar	50	Build.....	51
Page setup	47	Debug	51
Paste	49	Graphics	51
PLC.....	7, 9	Standard	51
functionality	11	Tools	65
Print.....	47	translate	14
preview	47	Unix	10
setup	47	USB.....	33
program editor	10	VBA.....	10
Project		Visual C.....	10
close.....	46	VisualBasic	10
new	45	VisualBasic for Applications.....	10
open.....	45	Window	
Properties.....	59	Cascade.....	68
RAM.....	14	Tile	
Recent		horizontally.....	68

vertically	68	Xemo operating mode	62, 63
Workspace	50	Xemo panel	65
options	66	Xemo status	62
XCode	60	Xemo!Go	10, 65, 71
Download to ROM	63	Application window	71
generate	60	Digital inputs and outputs	72
list	60	Xemo-DLL	10
Xemo	9		