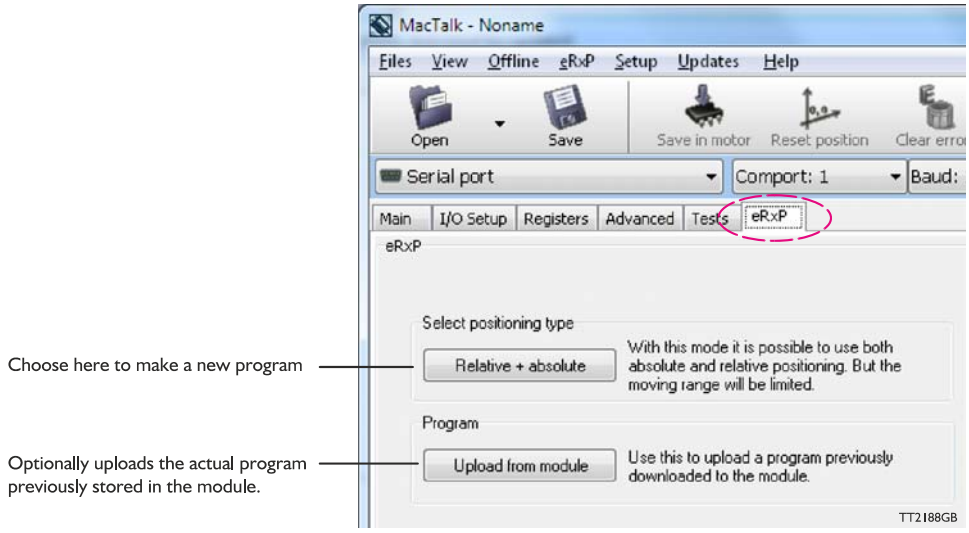
9.1 Getting started with programming

When using the MIS motors, almost any kind of program can be created using a set of user friendly icons.

Make the required choice on the eRxP Programming tab.
The name eRxP refers to the programmable module (R-module) from the MAC motor series. [e]mbedded [R]-module number [x] [P]rogramming

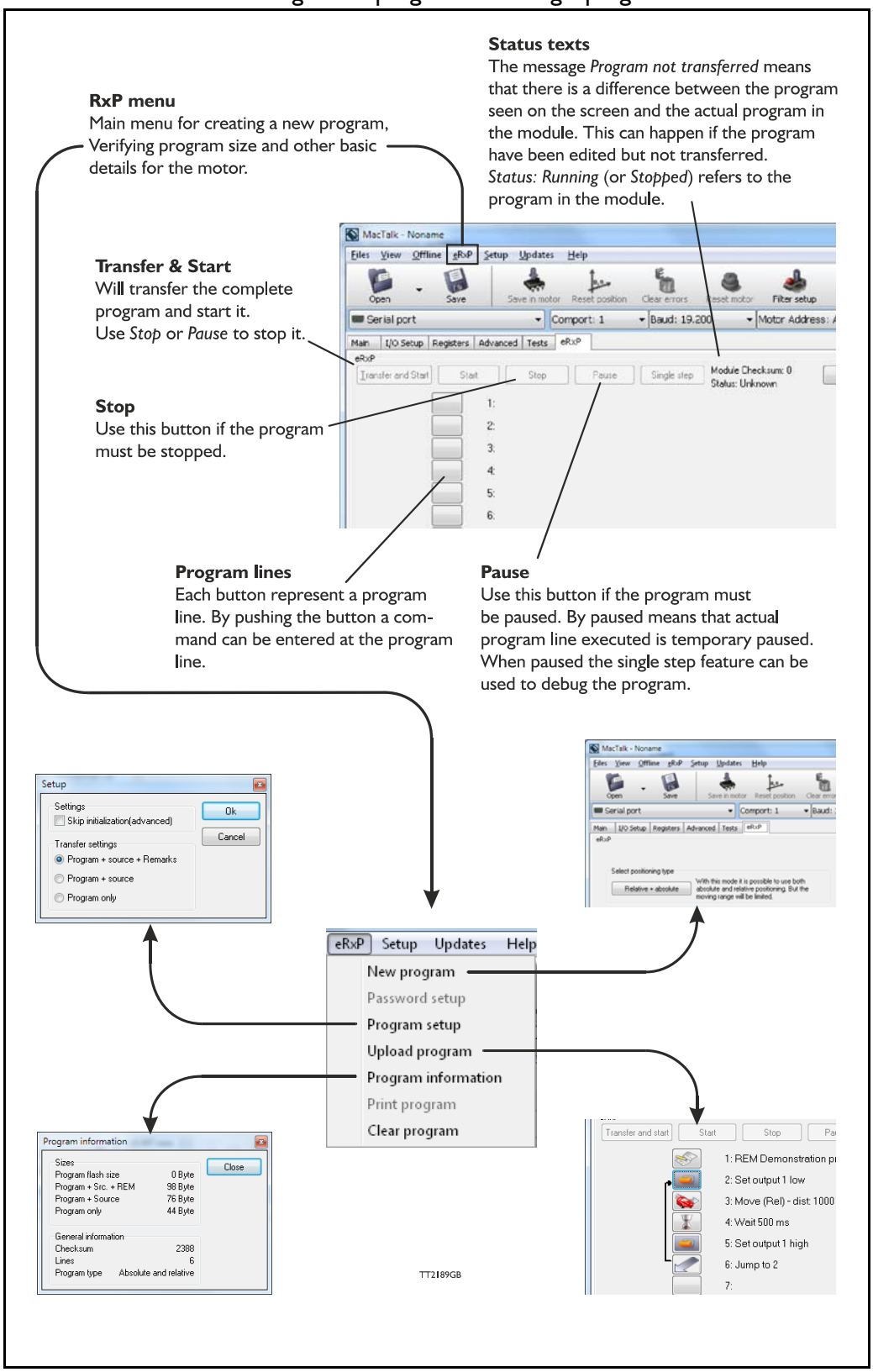


After making one of these 2 choices, the program window will be opened.

Please note: When a program is made and stored the motor will always startup in position mode. If this is not convinient insert a Mode = “passive” on first program line.

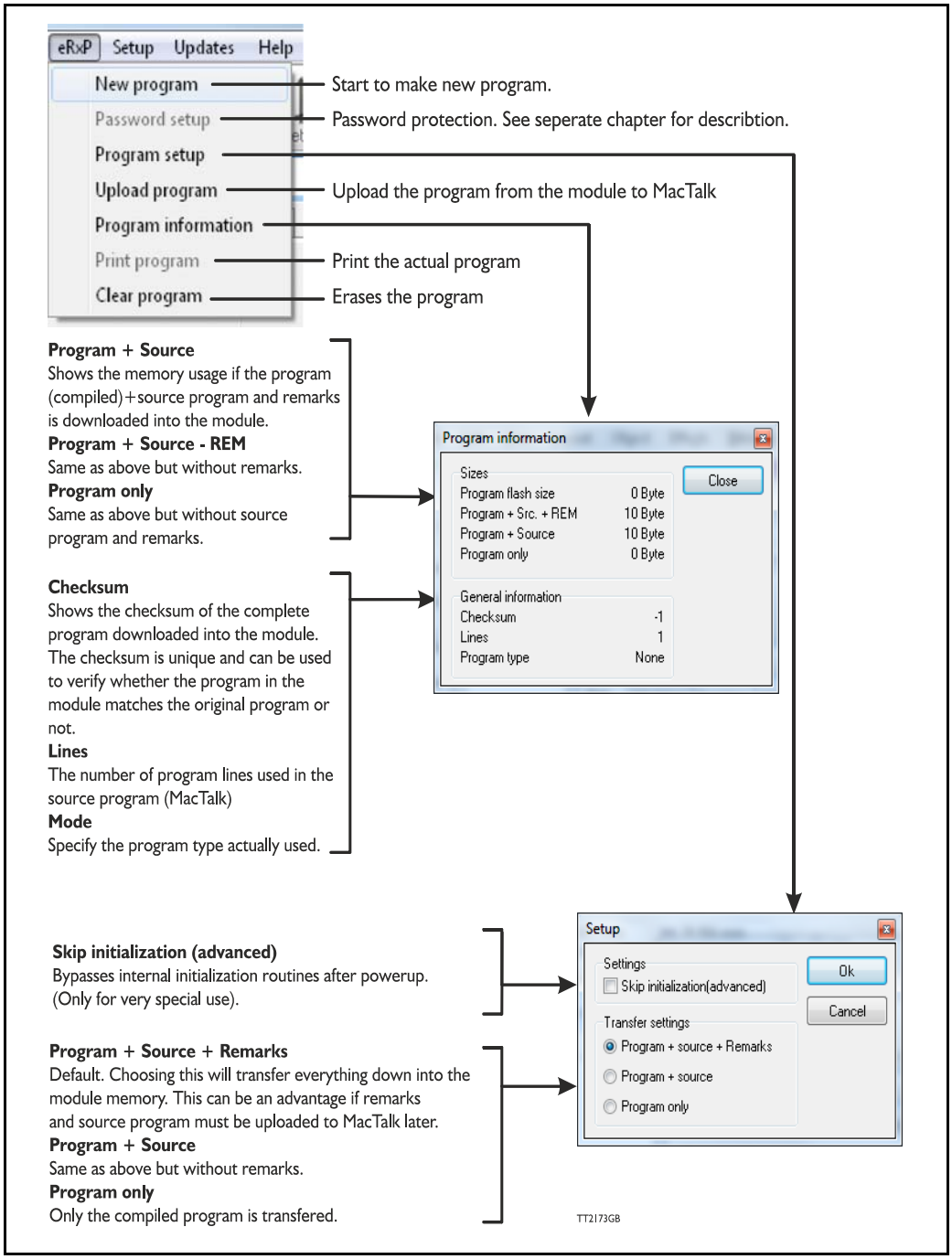
9.2 Programming Main window

The main window for creating a new program or editing a program is shown below:



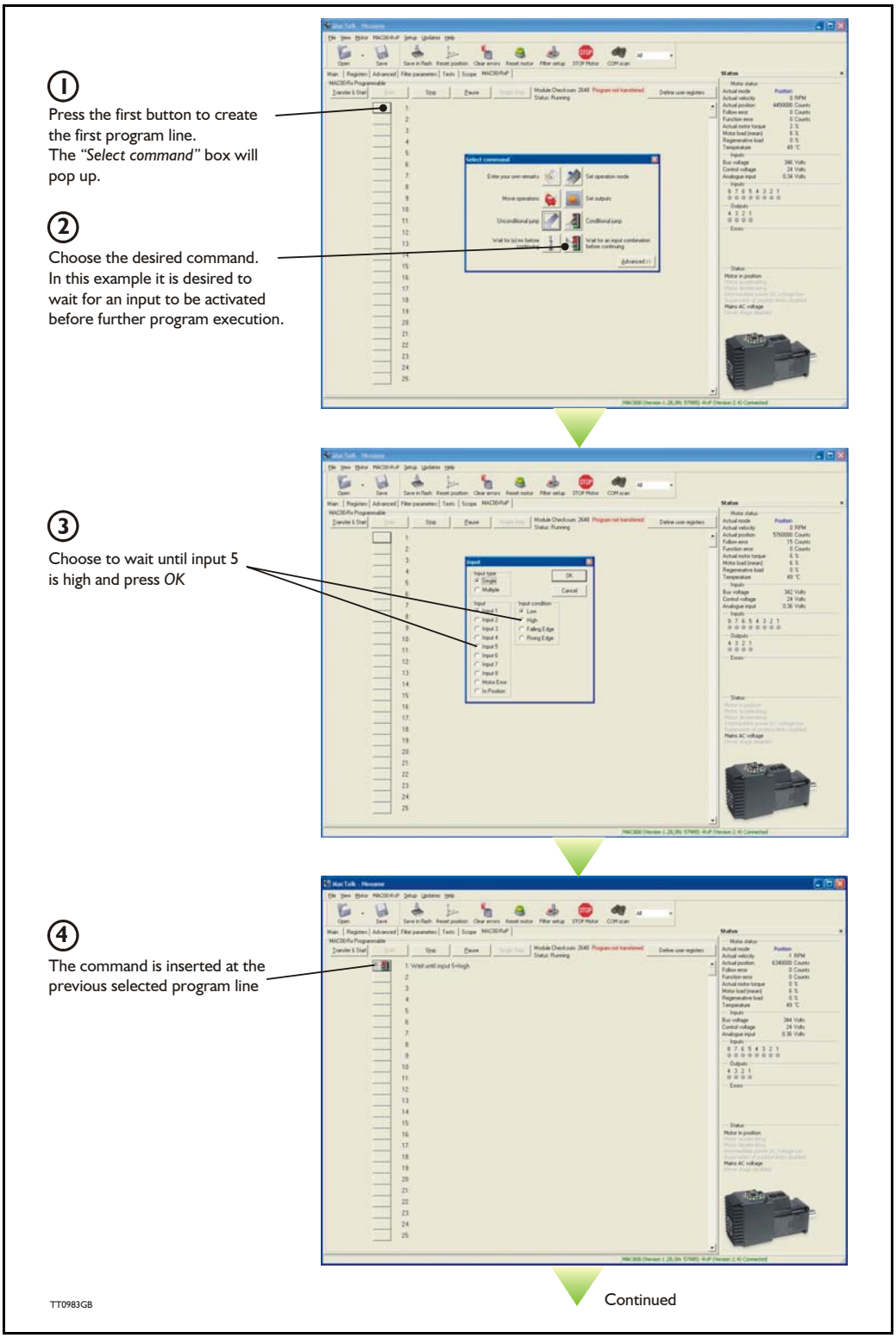
9.3 Programming menu

The menu found at the top of the main window gives access to the following options:



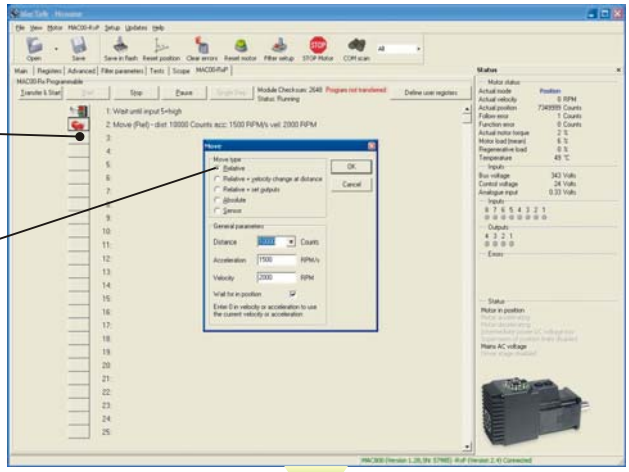
9.4 How to build a program

When choosing New program in the Programming menu or entering MacTalk for the first time, programming can be started.
Press the button at line 1 and a tool box will pop up.

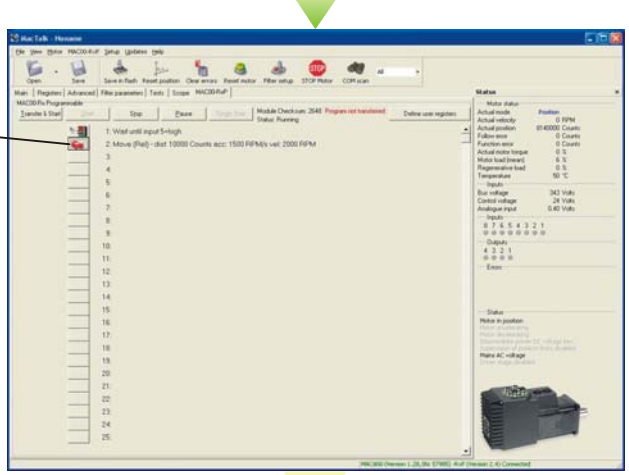


⑤
Press the second button to create the second program line

⑥
Choose the movement type needed.
Relative: Move x counts forward with reference to the actual position.
Absolute: Move to the x position with reference to the zero search position.

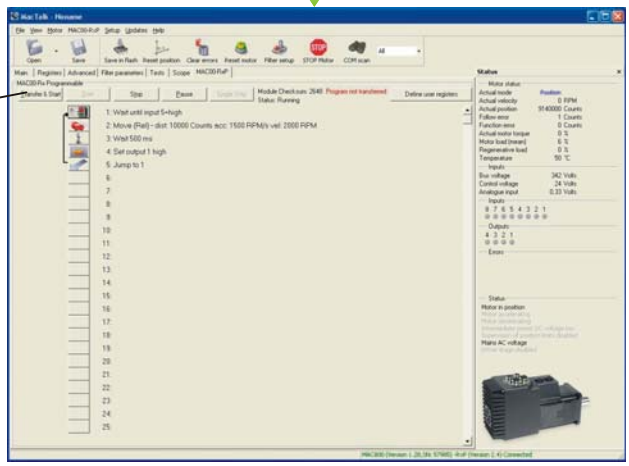


⑦
The relative move command just entered is converted into a program line.



⑧
Multiple program lines are entered by the user forming the last part of the program.

⑨
Now the program is finished.
Press the *"Transfer & Start"* button.
Now the program will be transfered and stored permanently in the module.
The program will be executed immidiately



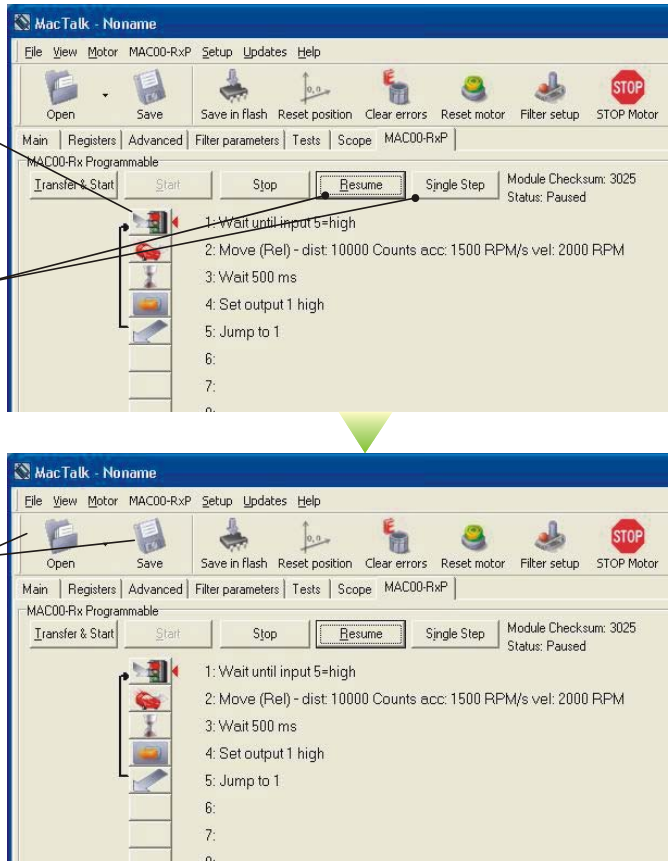
TT0984GB

Continued

⑩
Now the program is running continuously. The actual program line which is executed is shown by the small red arrow.

⑪
By choosing the "Pause" button, the program is paused. After it is paused, it is possible to single step through each program line which can be a useful feature to debug the program since the action in each line can be closely observed.

⑫
When the program is finished, it can be saved on the harddisc or floppy disc. Please be aware that when saving the program it is the complete program including the overall setup of the motor such as servofilter, I/O setup etc. Everything is stored in a file with the extension *.MAC*. Later it can be opened and restored in the motor.



TT0985GB

9.5 General programming hints

When programming and saving programs the following hints may be useful to ensure that the program behaves as expected.

1. When transferring the program to the motor, it is saved permanently in memory and the program will be executed each time the motor is switched on.
2. Before beginning to program, ensure that the basic parameters for controlling acceleration, torque, safety limits, etc. are set to proper values. When saving the program to the PC, all of these basic parameter settings will be saved together with the program as a complete motor setup package.
3. A program line can be edited by double-clicking on the command text.
4. When the cursor is placed on top of the command icon, an edit menu will be shown by right-clicking.

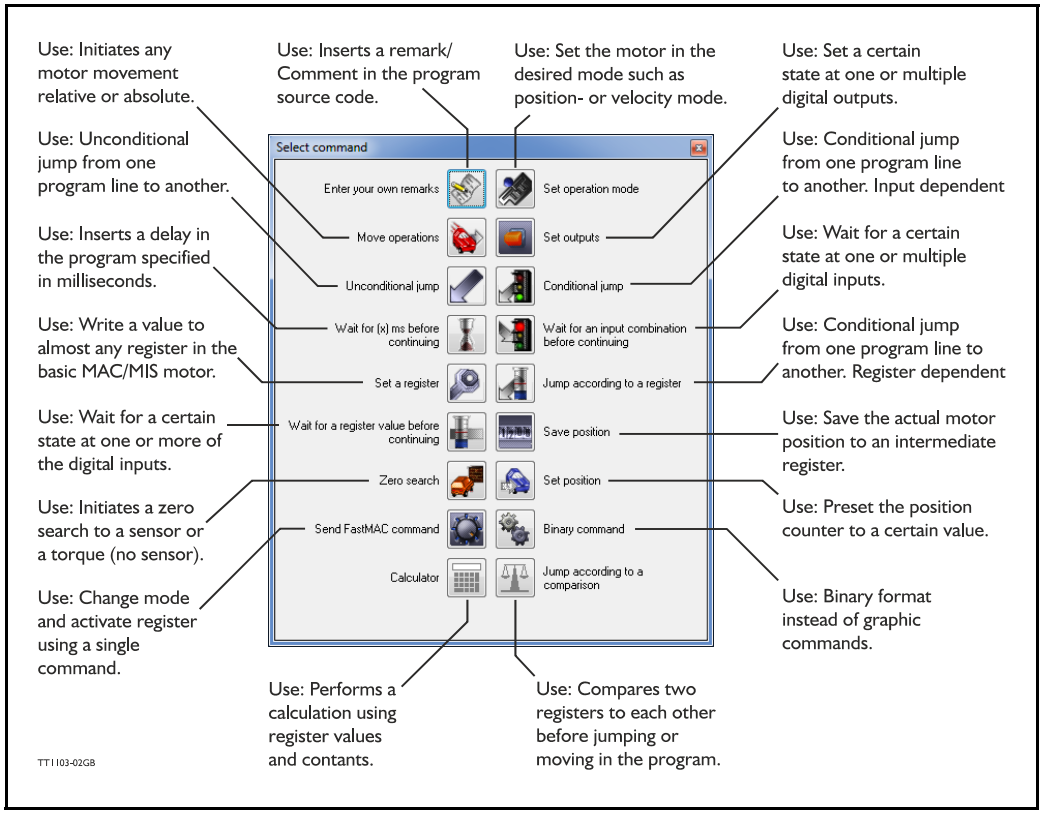
9.6 Command toolbox description

The toolbox used for programming covers 18 different command types.

The basic idea of the commands is to provide easy access to the most common functions of the motor. Some functions may seem to be missing at first glance, but the buttons “Set register in the QuickStep motor” or “Wait for a register value before continuing” give direct access to all the 50 registers in the basic QuickStep motor, such as the gear ratio or the actual torque register.

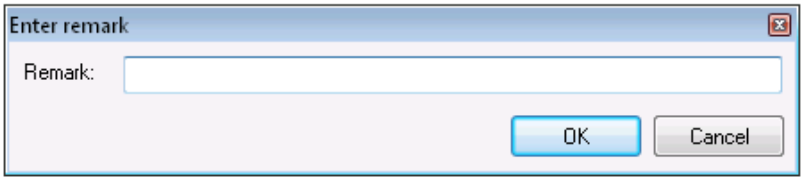
In total, this gives a very powerful programming tool since >95% of a typical program can be built using the simple command icons, while the remaining 5% is typically achieved by accessing the basic motor registers directly.

The following gives a short description of all 18 command icons.

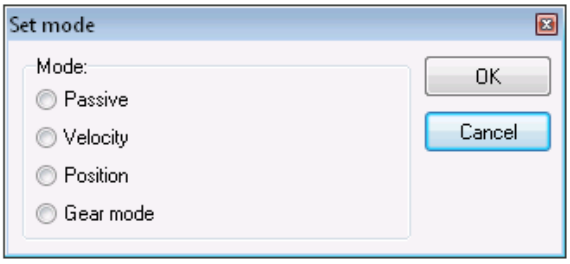


9.7 Graphic programming command reference

9.7.1 Enter your own remarks

Icon:	
Dialogue:	
Function:	Inserts a remark/comment in the source code. The program line will not do anything, but can make the source code easier to read. This can be very important if other programmers have to review or work on the code, or if the program is only worked on infrequently.

9.7.2 Set operation mode

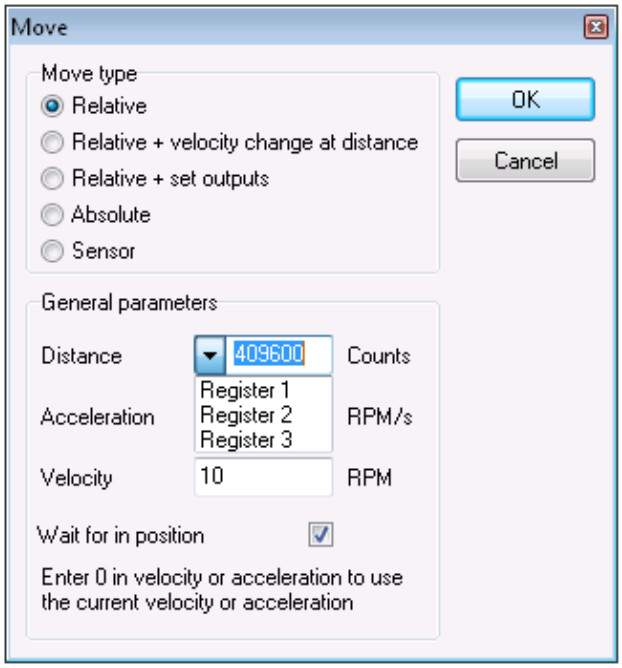
Icon:	
Dialogue:	
Function:	Sets the operating mode of the motor. When the program encounters a program line with this command, the motor's operating mode will be set to the specified mode. This allows you to use different operating modes in different parts of the program. For a detailed description of the individual operating modes, refer to section 1.3.1., *Basic modes/functions in the QuickStep motor*, page 12.

9.7.3 Move operations

Icon:	
Function:	The Move commands are very flexible, with five different operating modes. Each mode is described in its own section below.

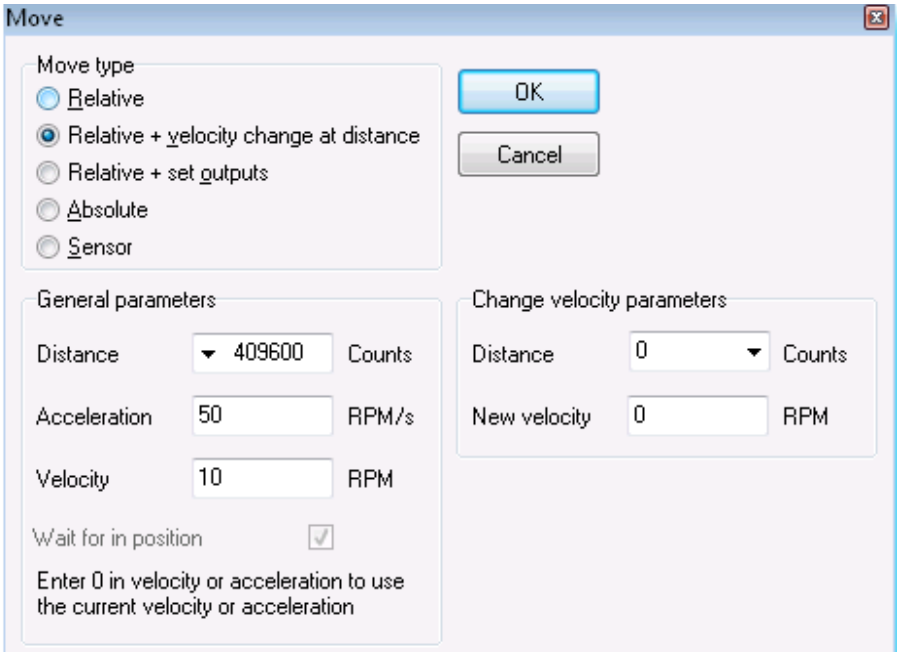
9.7 Graphic programming command reference

9.7.4 Move (Relative)

Icon:	
Dialogue:	
Function:	Performs a movement relative to the current position. The distance moved is measured in encoder counts, and can either be entered directly or taken from three registers in the user memory area. For further information on using these memory registers, refer to the sections on the 'Save position' and 'Set position' commands. Note that if you specify a velocity, motor register no. 5 (V_SOLL) will be overwritten with this velocity value. Also, if you specify an acceleration, motor register no. 6 (A_SOLL) will be overwritten with the acceleration value specified. Register no. 49 (P1) is always overwritten by this command. If the 'Wait for in position' option is checked, the program will wait until the motor has finished the movement, before proceeding to the next program line. If this option is not checked, the program will start the movement, then immediately start executing the next command. The motor will finish the movement on its own, unless given other instructions by the program.

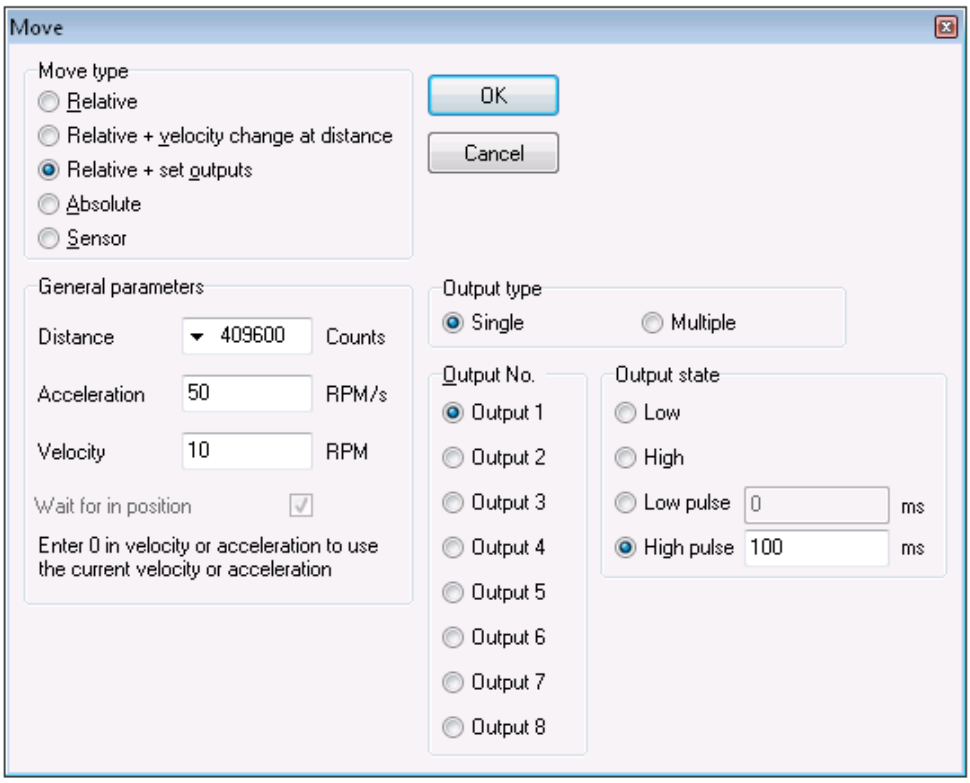
9.7 Graphic programming command reference

9.7.5 Move (Relative + velocity change at a distance)

Icon:	
Dialogue:	
Function:	Performs a relative movement, and changes velocity at a specified distance before reaching the new position. The distances are measured in encoder counts and can either be entered directly, or taken from three memory registers in the RxP module. For further information on using these memory registers, refer to the sections on the 'Save position' and 'Set position' commands. Note that motor register no. 5 (V_SOLL) will always be overwritten with the value specified in the 'New velocity' field. Also, if you specify an acceleration, motor register no. 6 (A_SOLL) will be overwritten with the acceleration value specified. Register no. 49 (P1) is always overwritten by this command. This command always waits until the movement is finished, before proceeding to the next line in the program. In case a fatal (system) error happens such as temperature error the program excecution stay at this command line until the motor is reset. Avoid selecting the "Wait for in position" flag and use a loop after the move command which is looking at "in position flag" and "error"

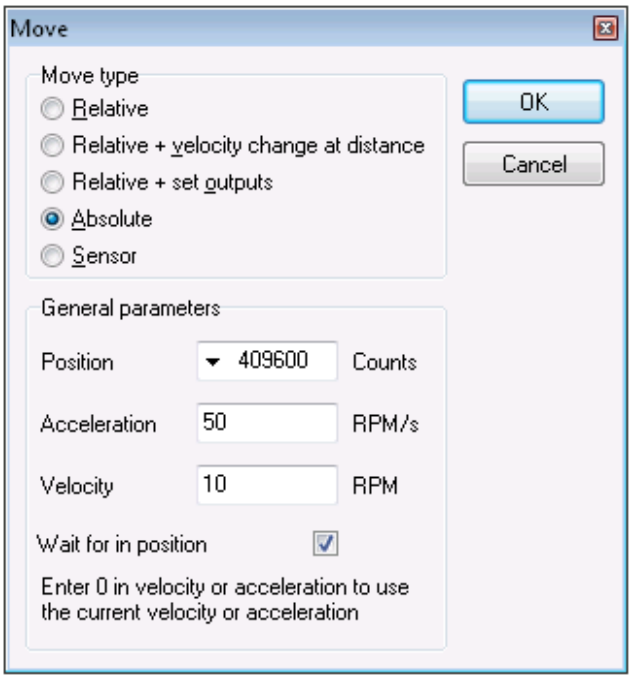
9.7 Graphic programming command reference

9.7.6 Move (Relative + set outputs)

Icon:	
Dialogue:	
Function:	Performs a movement relative to the current position, and sets one or more outputs when the operation is completed. The distance moved is given in encoder counts and can either be entered directly, or can be taken from one of three memory registers in the user memory area. For further information on using these memory registers, refer to the sections on the 'Save position' and 'Set position' commands. Note that if you specify a velocity, motor register no. 5 (V_SOLL) will be overwritten with this velocity value. Also, if you specify an acceleration, motor register no. 6 (A_SOLL) will be overwritten with the acceleration value specified. Register no. 49 (P1) is always overwritten by this command. This command always waits until the movement is finished, before proceeding to the next line in the program.

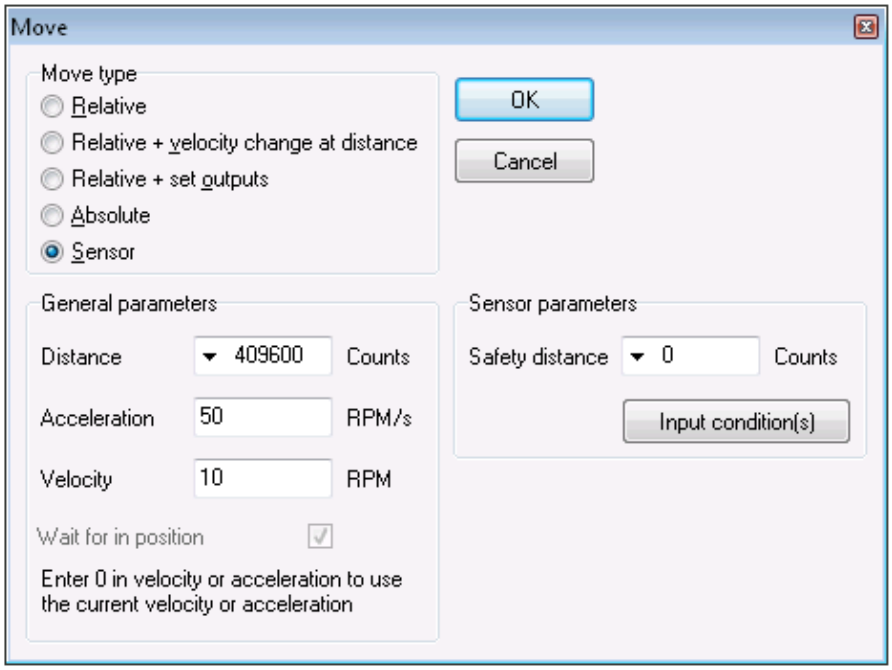
9.7 Graphic programming command reference

9.7.7 Move (Absolute)

Icon:	
Dialogue:	
Function:	Moves to an absolute, non-relative position. The position is given in encoder counts and can either be entered directly, or can be taken from one of three memory registers in the user memory area. For further information on using these memory registers, refer to the sections on the 'Save position' and 'Set position' commands. Note that if you specify a velocity, motor register no. 5 (V_SOLL) will be overwritten with this velocity value. Also, if you specify an acceleration, motor register no. 6 (A_SOLL) will be overwritten with the acceleration value specified. If the 'Wait for in position' option is checked, the program will wait until the motor has finished the movement before proceeding to the next program line. If this option is not checked, the program will start the movement, then immediately start executing the next command. The motor will finish the movement on its own, unless given other instructions by the program.

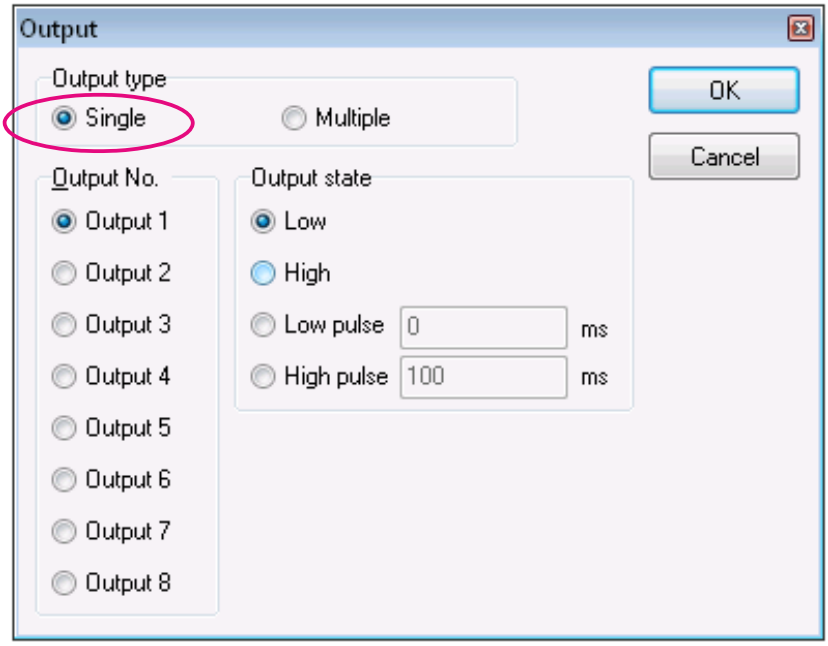
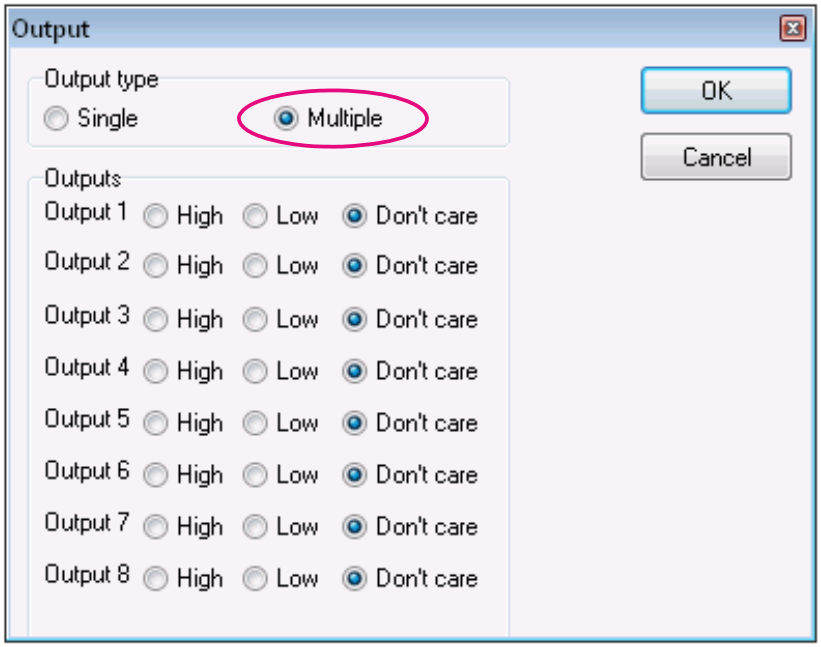
9.7 Graphic programming command reference

9.7.8 Move (Sensor)

Icon:	
Dialogue:	
Function:	Performs a movement in the direction specified until an input condition is satisfied. The motor then moves the distance specified before stopping. The motor will not move farther than the Safety distance specified, regardless of whether the input condition is satisfied. The distances are measured in encoder counts and can either be entered directly, or taken from three memory registers in the user memory area. For further information on using these memory registers, refer to the sections on the 'Save position' and 'Set position' commands. Note that if you specify a velocity, motor register no. 5 (V_SOLL) will be overwritten with this velocity value. Also, if you specify an acceleration, motor register no. 6 (A_SOLL) will be overwritten with the acceleration value specified. Register no. 49 (P1) is always overwritten by this command. This command always waits until the movement is finished before proceeding to the next line in the program.

9.7 Graphic programming command reference

9.7.9 Set outputs

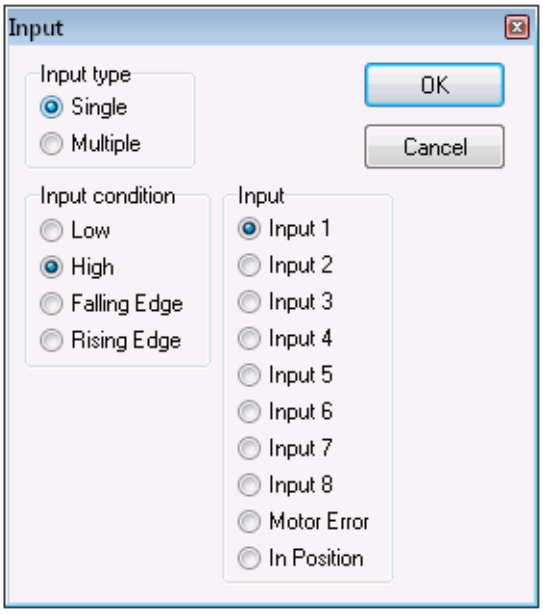
Icon:	
Dialogue:	 
Function:	Sets one or more outputs. When setting a single output, you can set it to high, low, or you can specify the length (in milliseconds) of a pulse to send out on that output. When setting multiple outputs, you can specify whether to set each output high, low, or leave it in its current state.

9.7 Graphic programming command reference

9.7.10 Unconditional jump

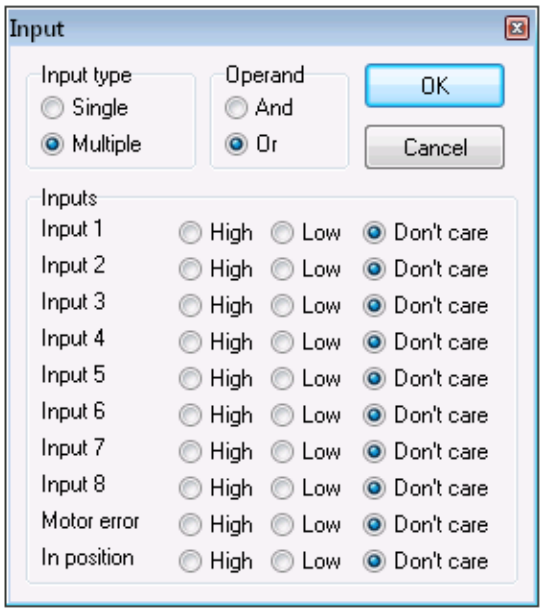
Icon:	
Dialogue:	None. After selecting this command, the mouse cursor changes. The next program line that you click on will become the destination for the jump.
Function:	Jumps to another line in the program.

9.7.11 Conditional jump (single input)

Icon:	
Dialogue:	
Function:	Tests for an input condition before either jumping to another line in the program or moving on to the next line in the program. If the condition is met, the command jumps to the specified program line. If the condition is not met, the program proceeds to execute the next line in the program. When 'Input type' is set to 'Single', the command can test a single input for one of four possible conditions: the input is low, the input is high, the input has transitioned to low (Falling Edge), or the input has transitioned to high (Rising Edge). If transitions are tested for, the transition must have taken place during the last 30 microseconds. After pressing the OK button, the dialogue will disappear, and the mouse cursor will change. The next program line that you click on will then become the destination of the jump command.

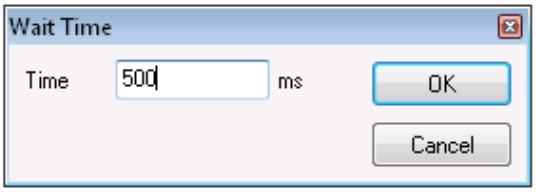
9.7 Graphic programming command reference

9.7.12 Conditional jump (multiple inputs)

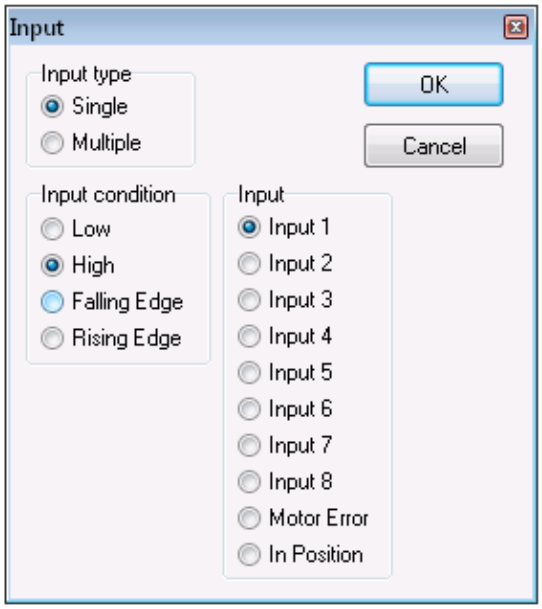
Icon:	
Dialogue:	
Function:	Tests for an input condition before either jumping to another line in the program or moving on to the next line in the program. If the condition is met, the command jumps to the specified program line. If the condition is not met, the program proceeds to execute the next line in the program. When 'Input type' is set to 'Multiple', multiple inputs can be tested for being either high or low. The 'Operand' setting determines whether one or all of the inputs must meet their test criterion. If set to 'And', all inputs must match their test settings. If set to 'Or', only one input need match its test setting. Inputs that are set to 'Don't care' are not tested. After pressing the OK button, the dialogue will disappear, and the mouse cursor will change. The next program line that you click on will then become the destination of the jump command.

9.7 Graphic programming command reference

9.7.13 Wait for (x) ms before continuing

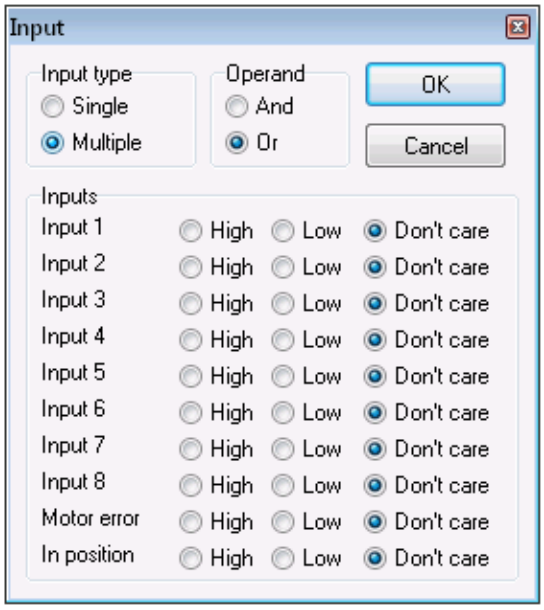
Icon:	
Dialogue:	
Function:	Causes the program to pause for a number of milliseconds before continuing. The maximum pause that can be specified is 32767 milliseconds. The minimum pause that can be specified is 0 milliseconds. Note that this command overwrites Timer 1 in the RxP module's memory.

9.7.14 Wait for an input combination before continuing (single input)

Icon:	
Dialogue:	
Function:	Waits for a specified input condition to occur. The next line in the program will not be executed until the input condition has been met. If 'Input type' is set to 'Single', the command will wait for one of four things to happen on the specified input: that the input tests as high, that the input tests as low, that the input transitions from high to low (Falling Edge), or that the input transitions from low to high (Rising Edge). The input is tested with 30 microsecond intervals.

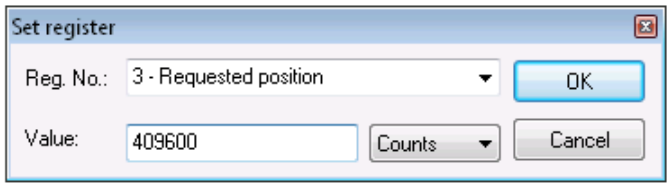
9.7 Graphic programming command reference

9.7.15 Wait for an input combination before continuing (multiple inputs)

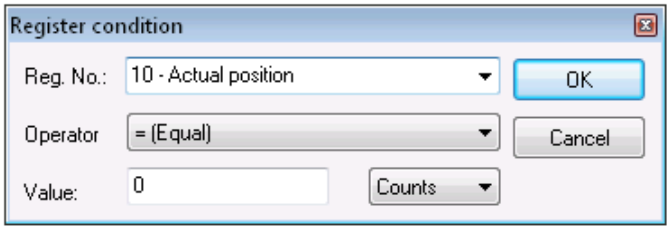
Icon:	
Dialogue:	
Function:	Waits for a specified input condition to occur. The next line in the program will not be executed until the input condition has been met. If 'Input type' is set to 'Multiple', multiple inputs can be tested for being either high or low. The 'Operand' setting determines whether one or all of the inputs must meet their test criterion. If set to 'And', all inputs must match their test settings. If set to 'Or', only one input need match its test setting. Inputs that are set to 'Don't care' are not tested. The inputs are tested with 30 microsecond intervals.

9.7 Graphic programming command reference

9.7.16 Set a register in the MIS motor

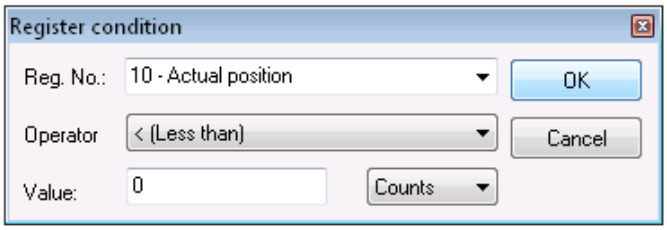
Icon:	
Dialogue:	
Function:	Sets a register in the motor to a specified value. The register is selected from a list of known, user-accessible registers. The value can either be entered as native motor units or it can be entered as generic engineering units. The dialogue above provides an example: register no. 3 (P_SOLL, or Requested position, depending on your preference) can either be set to an integer number of encoder counts, or it can be set to a non-integer number of revolutions.

9.7.17 Jump according to a register in the MAC motor

Icon:	
Dialogue:	
Function:	Tests a register in the motor against a specified value before either jumping to another line in the program or moving on to the next line in the program. If the condition is met, the command jumps to the specified program line. If the condition is not met, the program proceeds to execute the next line in the program. The value can either be entered as native motor units, or it can be entered as generic engineering units. The dialogue above provides an example: register no. 10 (P_IST, or Actual position, depending on your preference) must be equal to 0 revolutions if the jump is to be executed. The position that the register is tested against can be specified as an integer number of encoder counts or can be specified as a non-integer number of revolutions. After pressing the OK button, the dialogue will disappear and the mouse cursor will change. The next program line that you click on will then become the destination of the jump command.

9.7 Graphic programming command reference

9.7.18 Wait for a register value before continuing

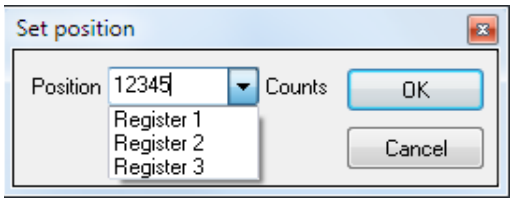
Icon:	
Dialogue:	
Function:	Tests a register in the motor against a specified value and waits until the specified condition is met. The value can either be entered as native motor units or can be entered as generic engineering units. The dialogue above provides an example: register no. 10 (P_IST, or Actual position, depending on your preference) must be less than 0 revolutions, before the program will continue. The position that the register is tested against can be specified as an integer number of encoder counts, or can be specified as a non-integer number of revolutions.

9.7.19 Save position

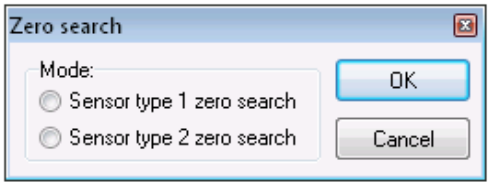
Icon:	
Dialogue:	
Function:	Saves the current position from register no. 10 (P_IST) to one of three locations in the user memory area. The saved position(s) can then be used whenever a position or distance is needed in a move command.

9.7 Graphic programming command reference

9.7.20 Set position

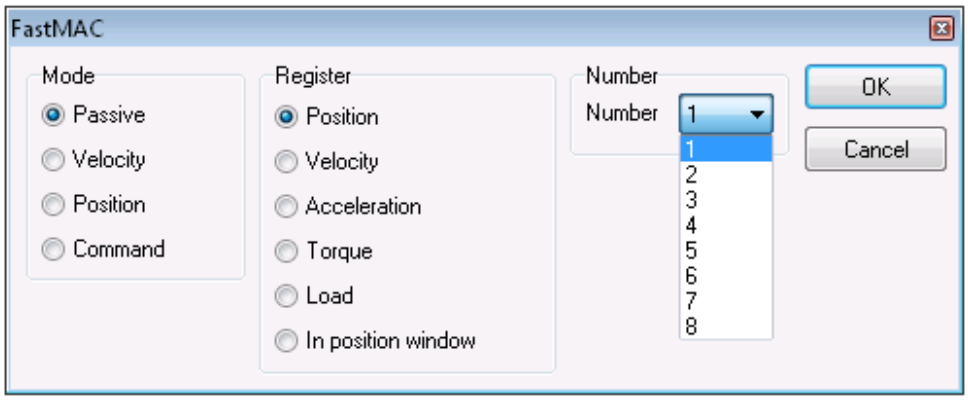
Icon:	
Dialogue:	
Function:	Change the "Actual postion" (P_IST register 10) value to a new value or the value in one of three position values stored in the user memory area (register 1, 2 or 3). This is the reverse of the 'Save position' command.

9.7.21 Zero search

Icon:	
Dialogue:	
Function:	Initiates a zero search. The program waits until the zero search has completed before proceeding to the next command. For a detailed description of how to set up a zero search, refer to *Zero search modes*, page 163

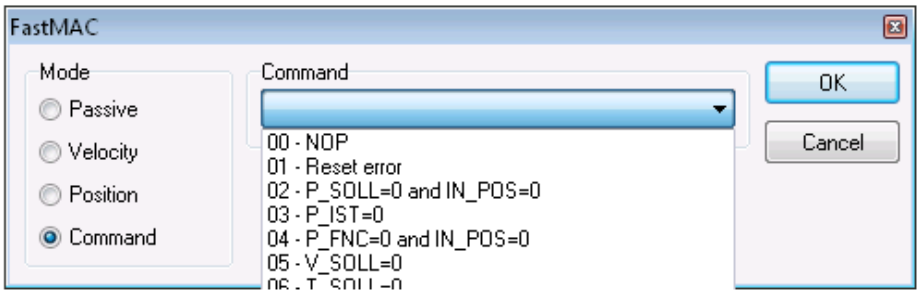
9.7 Graphic programming command reference

9.7.22 Send FastMAC command (change mode and activate register)

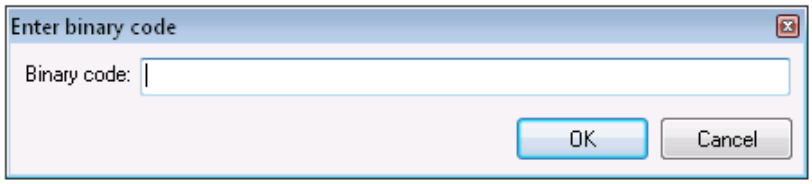
Icon:	
Dialogue:	
Function:	FastMAC commands are also sometimes referred to as FlexMAC commands. The advantage of these commands is a very low communication overhead. FastMAC/FlexMAC commands are described in detail in section 4.5.7 of the MAC user manual, JVL publication no. LB0047-20GB (V2.0 or newer). However, a brief summary is in order. If 'Mode' is set to 'Passive', 'Velocity', or 'Position', the motor will switch to that mode. Also, one of the passive motor registers will be activated, in the sense that its value will be written to the corresponding active motor register, which actually controls motor behaviour. In the example above, the value in register no. 65 (V1) will be written to register no. 5 (V_SOLL). Move operations will then take place at that velocity.

9.7 Graphic programming command reference

9.7.23 Send FastMAC command (macro command)

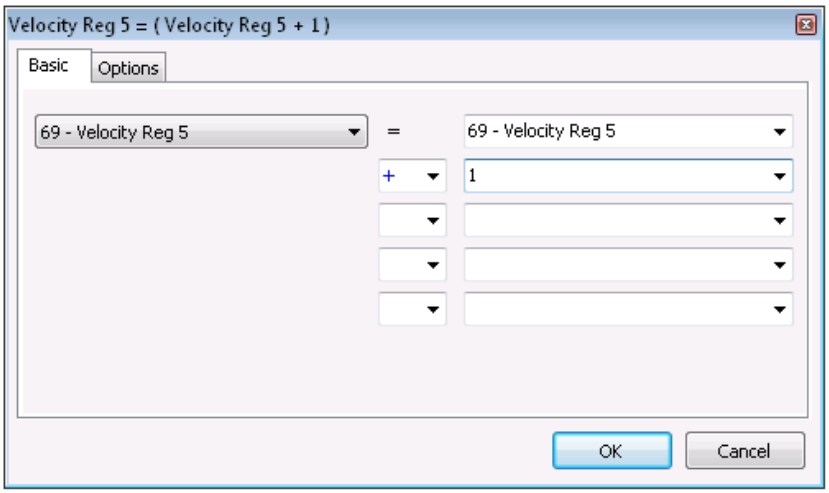
Icon:	
Dialogue:	
Function:	If 'Mode' is set to 'Command', the motor does not necessarily change mode but it can be commanded to carry out a series of predetermined operations. Describing all of the FastMAC commands is beyond the scope of this section but for example, using a single command it is possible to activate four different sets of registers, each controlling position, velocity, acceleration, torque, load factor, and in-position window. FastMAC/FlexMAC commands are described in detail in section 4.5.7 of the MAC user manual, JVL publication no. LB0047-20GB (V2.0 or newer). However, a brief summary is in order.

9.7.24 Binary command

Icon:	
Dialogue:	
Function:	MacTalk programs are sent to the motor in a compact, binary format, which is then interpreted by the motor's firmware. The existing set of graphic commands covers most situations, but when special needs arise, anything that can be done with programs can be done with a binary command. If special needs arise that are not covered by the other commands, contact JVL for assistance.

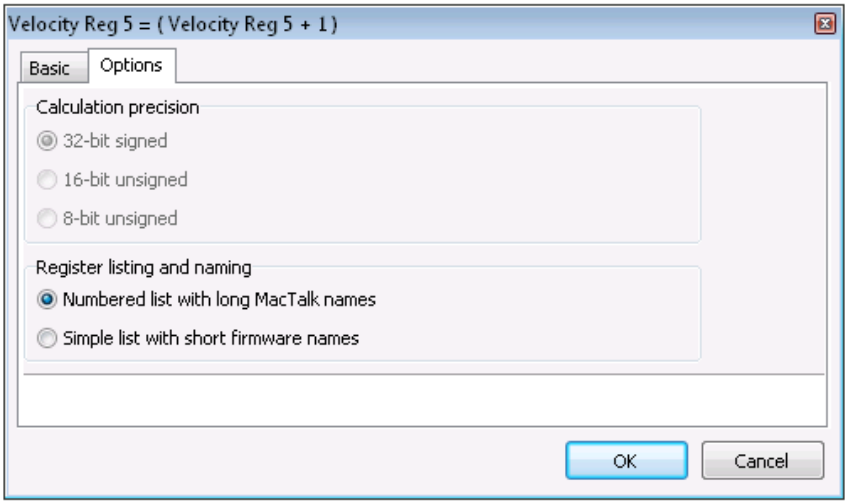
9.7 Graphic programming command reference

9.7.25 Calculator (basic)

Icon:	
Dialogue:	
Function:	Performs a calculation using register values, constants, and the four basic arithmetic operations: +, -, * and /. The result is stored in a register. Arithmetic operations take place in the order that they are specified. Operands/arguments can be either integer constants or registers. The caption of the dialogue box shows the resulting expression in traditional infix format. It is continuously updated as you type in the expression. Note that if you write a value to a register using this command, that value is always measured in native motor units. Conversion from generic engineering units is only supported for the commands 'Set a register', 'Jump according to a register', and 'Wait for a register value before continuing'. If you make a calculation please be aware that most of the registers in the MIS motors operate with integers so its often needed to multiply before doing a division in order to become a precise result. Also be aware that all calculations are done in 32 bit format which gives the possibility to operate with values from -2^{31} to 2^{31}. If the result of a calculation gives a higher value than 2^{31} it will therefore becomes negative and similar if a calculationresult becomes lowe than -2^{31} the result becomes positive.

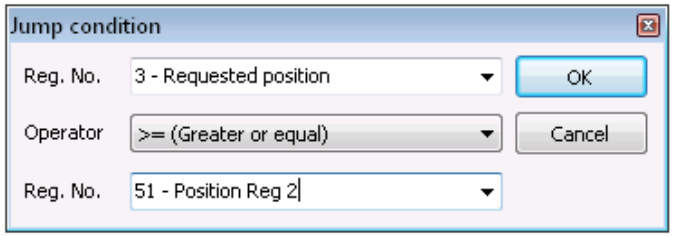
9.7 Graphic programming command reference

9.7.26 Calculator (options)

Icon:	
Dialogue:	
Function:	The options tab contains various settings that affect the operation of the Calculator command. 'Calculation precision' is currently preset to 32-bit precision and cannot be changed. This is not an error, and should not be reported. 'Register listing and naming' provides an alternative method of entering data into the dialogue by selecting 'Simple list with short firmware names'. Instead of selecting, for example, '3 – Requested position' to access register no. 3, you can simply type 'P_SOLL'. If you wish to enter a constant, you simply enter the digits – the dialogue will not mistake the constant for a register number. If you are in doubt about a register name, look at the expression in the caption of the dialogue box. A recognized register name will appear in the expression. An unrecognizable register name will appear as a zero. You can switch between the two methods of data entry at any time.

9.7 Graphic programming command reference

9.7.27 Jump according to a comparison

Icon:	
Dialogue:	
Function:	Compares two registers with each other before either jumping to another line in the program or moving on to the next line in the program. If the condition is met, the command jumps to the specified program line. If the condition is not met, the program proceeds to execute the next line in the program. Any two registers can be compared with each other but the command does not do anything beyond comparing the registers numerical values measured in native motor units. To ensure that comparisons are meaningful, it is preferable to compare registers that hold the same type of information in the same binary format. In the example above, two position registers are compared. Both hold position information, both are 32-bit wide, and both measure position in encoder counts. Such a comparison will always yield meaningful, predictable results. For other types of registers, see the relevant register sections.

9.8 Command timing

Each command has a certain execution time. The specified execution time in the following table is the maximum execution time if not using CANopen, serial communication and the motor is disabled. The actual execution may be faster.

Icon	Name	Execution time [µs]
	Remarks	0
	Set operation mode	60
	Move relative (no velocity, no acceleration)[1]	90
	Move relative + set velocity (no acceleration)[1]	150
	Move relative + set velocity + set acceleration[1]	210
	Move absolute (no velocity, no acceleration)[1]	60
	Move absolute + set velocity (no acceleration)[1]	120
	Move absolute + set velocity + set acceleration[1]	180
	Set single output (high/low)	30
	Set multiple outputs	30*number of outputs
	Unconditional jump	30
	Conditional jump (inputs)	60
	Set a register	60
	Conditional jump (register)	120
	Save position	60
	Set position	90
	Send fastMAC command	30
	Binary command	30

1) The time for all move commands is shown without waiting for in position

9.9 More about program timing

The firmware is structured so that one program instruction is executed for each pass of the main loop, which takes approximately 30 microseconds (μs) without CANopen, without serial communications and when the motor is not running. The Main Loop Time is termed MLT in the following text.

A single program line in MacTalk can generate more than one instruction. For example, assigning a constant value to a register uses two instructions: First load the value to the internal stack and then Store from the stack to the target register. The above table in *Motor Connections*, page 414 reflects this operation.

The main loop time will vary depending on a number of factors: The serial communications speed and load, whether CANopen is installed, and the CANopen communications speed and load.

Serial communications on the RS-485 line can load the motor up to 1% at 19.200 baud, which is insignificant, but at the maximum baud rate of 921.600 the communications can load the motor up to 45%, which would result in an MLT of ~60 μs.

When CANopen firmware is installed, the basic MLT will change from 30 to 90 μs with no communications.
When loading the CANbus with communications, the MLT can rise significantly. For example, when using seven transmit PDOs with an event timer value of 1 ms and a CANbus link speed of 500 kbits/s, the MLT can rise to 150-200 μs. Also using RS-485 communications at high baud rates can result in even longer MLT values. However, this scenario is very unlikely.

Note: In applications where program timing is critical, tests must be performed to ensure that timing is satisfactory when communication is running according to conditions used in production!