

---

# **POSITIONING AND CONTOURING CONTROL SYSTEM**

## **APCI-8001, APCI-8008 AND CPCI-8004**

### **G-Code Interface**



<b>1 Procedure for processing G-code program files.....</b>	<b>7</b>
1.1 The McuWIN user interface .....	7
<b>2 Functionality of McuWIN .....</b>	<b>9</b>
2.1 Emergency power-off monitoring .....	9
2.2 Maximum lag error .....	9
2.3 Hardware limit switches .....	9
2.4 Software limit switches.....	9
2.5 Monitoring of the encoder error flag.....	10
2.6 Monitoring of position counters .....	10
2.7 Tool radius correction.....	10
2.8 Spindle error and angle error compensation.....	11
<b>3 Description of G-codes implemented .....</b>	<b>12</b>
3.1 G00 – Rapid positioning.....	12
3.2 G01 – Linear interpolation.....	12
3.3 G02 – Circular / helical interpolation .....	12
3.4 G03 – Circular / helical interpolation .....	12
3.5 G04 – Dwell time.....	13
3.6 G17 – Plane selection .....	13
3.7 G18 – Plane selection .....	13
3.8 G19 – Plane selection .....	13
3.9 G21 – Reflection of Y axis.....	13
3.10 G22 – Reflection of X axis.....	13
3.11 G23 – Reflection of X and Y axes .....	14
3.12 G24 – Disable reflection of all axes .....	14
3.13 G39 – Program positioning factor .....	14
3.14 G40 – Cancel tool radius correction.....	14
3.15 G41 – Tool radius correction, left.....	14
3.16 G42 – Tool radius correction, right.....	14
3.17 G51 – Program effective radius .....	15
3.18 G53 – Disable zero offset.....	15
3.19 G54 - G58 – Set zero offset .....	15
3.20 G60 – Define interpolation axes.....	16
3.21 G70 – Measurement in inches .....	16
3.22 G71 – Measurement in mm.....	16
3.23 G74 – Reference run.....	16
3.24 G90 – Absolute distance mode .....	16
3.25 G91 – Relative distance mode.....	16
3.26 G92 – Set zero offset .....	16
3.27 G93 – Reverse-time feed encoding .....	17
3.28 G94 – Time unit in minutes .....	17
3.29 G98 – Set position of software limit switch - left .....	17
3.30 G99 – Set position of software limit switch - right .....	17

3.31	G150 – Spline off .....	17
3.32	G151 – Spline on .....	18
3.33	G153 – Read zero offset .....	18
3.34	G154 – Reprogram zero offset returned .....	18
3.35	G161 – Center coordinates, relative or absolute .....	18
3.36	G162 – Center coordinates always relative .....	18

#### **4 M-codes.....19**

4.1	M00 – Program stop.....	19
4.2	M01 – Optional stop .....	19
4.3	M02 – End of program .....	19
4.4	M03 – Spindle clockwise.....	19
4.5	M04 – Spindle anti-clockwise.....	19
4.6	M05 – Spindle stop .....	20
4.7	M06 – Change tool.....	20
4.8	M08 – Coolant on.....	20
4.9	M09 – Coolant off.....	20
4.10	M17 – End of sub-routine.....	20
4.11	M26 – Set output.....	20
4.12	M27 – Reset output.....	21
4.13	M30 – Program stop.....	21
4.14	M80 – Write to outputs (word by word) .....	21
4.15	M96 – Unconditional jump.....	21
4.16	M98 – Call sub-routine .....	22
4.17	M100 – Reset “Program end” output .....	22
4.18	M150 – Start recording for graphical systems analysis .....	22
4.19	M901 – Save common-integer-variable as resident .....	22
4.20	M902 – Read back common integer variable saved as resident .....	23
4.21	M903 – Save common double variable as resident.....	23
4.22	M904 – Read back common double variable saved as resident .....	23
4.23	M910 – Switch off PC after program end.....	23

#### **5 Other codes .....24**

5.1	Labels.....	24
5.2	Special functions supported .....	24
5.3	F-command .....	24
5.4	S-command .....	24
5.5	T-command .....	25
5.6	D-command.....	25
5.7	String output.....	25
5.7.1	WRITE command.....	25
5.7.2	WRITELN command .....	26

#### **6 Comments.....27**

#### **7 Conditional program execution.....28**

<b>8</b>	<b>Loops .....</b>	<b>29</b>
8.1	Do-while loop .....	29
8.2	Repeat-until loop .....	29
8.3	For loop .....	29
<b>9</b>	<b>Integrating include files.....</b>	<b>30</b>
<b>10</b>	<b>Incorporation of rw_SymPas commands .....</b>	<b>31</b>
<b>11</b>	<b>Calculation parameters .....</b>	<b>32</b>
<b>12</b>	<b>Variables .....</b>	<b>33</b>
<b>13</b>	<b>Handling McuWIN.....</b>	<b>34</b>
13.1	Troubleshooting .....	34
13.1.1	McuWIN error messages.....	34
13.1.1.1	Error # 1: Unknown function code! .....	34
13.1.1.2	Warning # 1: Encode error! .....	34
13.1.1.3	Error # 2: Lag error (axis ?)! .....	34
13.1.1.4	Status #2: Wait until door is closed! .....	34
13.1.1.5	Error # 4: Hardware limit switch (left axis? / right axis ?)! .....	34
13.1.1.6	Status #4: Program was started! .....	34
13.1.1.7	Error # 8: Software limit switch (left axis ? / right axis ?)! .....	35
13.1.1.8	Status #8: Program was ended! .....	35
13.1.1.9	Status #10: Emergency stop .....	35
13.1.1.10	Warning # 10: Position error detected! .....	35
13.1.1.11	Error # 20: Conflict in configuration data! .....	35
13.1.1.12	Status #20: Referencing was started! .....	35
13.1.1.13	Error 21 when loading <<??>>.....	35
13.1.1.14	Error # 40: Error with reference run (axis ?)! .....	36
13.1.1.15	Status #40: Wait until spindle speed is reached... ..	36
13.1.1.16	Error # 80: Version conflict McuWIN - SAP! .....	36
13.1.1.17	Status #80: Error bit detected in ErrorReg! ? hex .....	36
13.1.1.18	Error # 100: Configuration error! .....	36
13.1.1.19	Status #100: Error bit detected in IFS register! ?? hex .....	36
13.1.1.20	Error 200: Error in tool radius correction! .....	36
13.1.1.21	Status #400: Feed block active (UI)! .....	36

13.1.1.22	Error # 800: Amplifier not ready! .....	36
13.1.1.23	Status #1000: Runtime error in SAP task! .....	37
13.1.1.24	Status #2000: Parameter error at runtime!.....	37
13.1.1.25	Error # 4000: Invalid S-value!.....	37
13.1.1.26	Error # 80000000: System was stopped! Reset required ... ..	37

## **14 Version information .....38**

14.1	Changes in earlier versions.....	38
------	----------------------------------	----

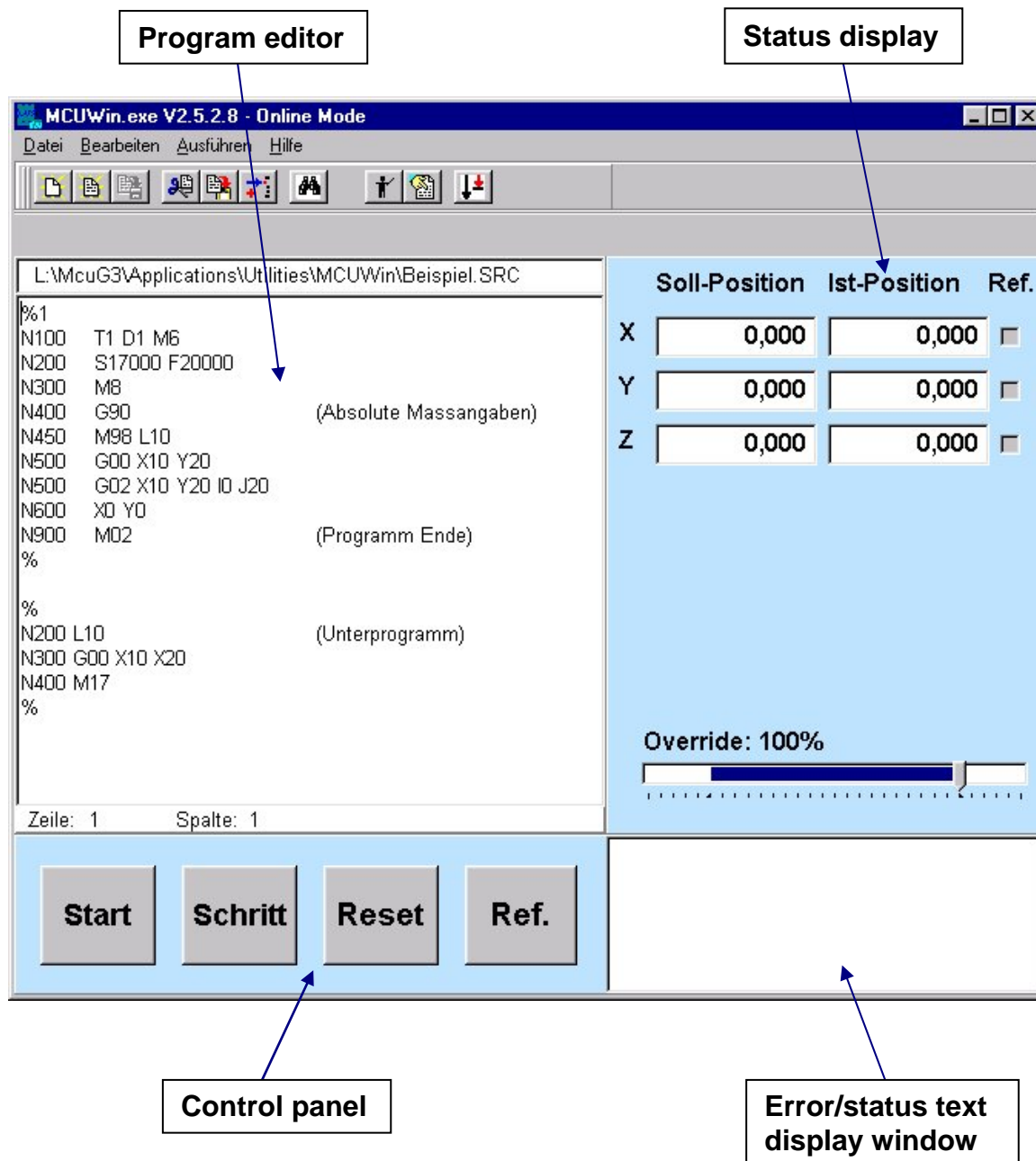
# **1 Procedure for processing G-code program files**

## **1.1 The McuWIN user interface**

The Windows application program McuWIN.exe provides the user with a user interface to run G-code programs. This interface can be used to edit programs, check their syntax and execute them.

It also displays target and actual positions, reference status and error messages. The required initialization steps for the controller are automatically executed by this program. Programs running in one-step mode can also be processed.

Additional information on processing G-code programs can be found in the file "McuWin User Interface". This will be helpful if McuWIN is not to be used because an in-house user interface is to be deployed or enhancements are needed.

**Figure:** Screenshot of McuWIN user interface



## 2 Functionality of McuWIN

### 2.1 Emergency power-off monitoring

When the axes are set up with the configuration program mcfg, digital inputs can be configured as emergency power-off inputs. If an EPO input is activated, any programs that are running will be interrupted, referencing may be cancelled depending on the configuration, the control circuits will be opened and the EPO status will be displayed.

When the EPO condition has been cleared, the control circuits will be closed again. The reference run, or a G-code program, can now be restarted. This allows controlled operation to be resumed after the emergency power-off.

**Warning:** The EPO signal **must** act directly on the drives (DC link voltage). The relevant security instructions must be observed.

### 2.2 Maximum lag error

When the axes are set up with the configuration program mcfg, an axis-specific “maximum lag error” can be entered. If this is exceeded with the control circuit closed, the program will be terminated and the cause of the error displayed.

### 2.3 Hardware limit switches

When the axes are set up with the configuration program mcfg, digital inputs can be configured as hardware limit switches. When a limit switch is triggered, the program will be terminated and the cause of the error displayed. Monitoring of the hardware limit switch via McuWIN only becomes active after a successful reference run. However, the limit switches will always act on traversing profiles with the specified option.

### 2.4 Software limit switches

When the axes are set up with the configuration program mcfg, digital inputs can be configured as software limit switches. When a limit switch is triggered, the program will be terminated and the cause of the error displayed. Monitoring of the software limit switch via McuWIN only becomes active after a successful reference run. The limit switches only act on traversing profiles with the specified option after referencing.

## 2.5 Monitoring of the encoder error flag

This functionality enables errors in positioning with incremental encoder systems to be detected early. This monitoring can be enabled at an axis-specific level. If a problem is detected, a warning will be output to the user interface.

## 2.6 Monitoring of position counters

This monitoring can also be enabled at an axis-specific level. The position is verified on every zero pulse, giving a reliable indication of when the system is miscounting.

This option can only be used where:

- Monitoring of the encoder flag is enabled;
- Incremental encoder systems are used;
- The denominator unit for the system value slsp is set with deg;
- Servo axes (not steppers) are used

If a problem is detected, a warning will be output to the user interface.

## 2.7 Tool radius correction

Tool radius correction supports commands G01, G02 and G03. Circuits must be located in the respective primary plane. The correction data has to be held in an INI file. The file name can be selected in the IniCFG configuration program on the "Files" tab. **ToolComp.INI** is the default for this file name.

This file has to have a fixed layout. For each correction level, there is a section that has to be identified according to the following pattern:

[ToolsXY]

The last two letters indicate the correction level. Other possibilities are:

[ToolsYZ]

[ToolsZX]

The letters X, Y and Z are fixed and are independent of any other axis names that may be selected.

After the code for the section, a value is given for each tool index for the radius and length correction, e.g. in the following form:

[ToolsXY]

R0=0.0

L0=0.0

R1=5.0

L1=2.0

**Note:** The tool radius correction table is loaded from the configuration file after every restart of McuWIN or after a reboot of the control. This file can be processed using a normal text editor or the add-on program **ToolEdit.exe**. If no tool radius correction table exists, then the function of the tool radius correction is not used. In this case, there is also no assignment of circular levels. It is not possible to select a tool index or other functions of the tool radius correction in this case. The standby of the tool radius correction is shown in C130 in bit-coded form.

## 2.8 Spindle error and angle error compensation

McuWIN provides the user with options to compensate spindle errors and angle errors where there is a Cartesian axis arrangement. To do this, simple text files containing the compensation tables must be added to the McuWIN directory. The compensation is activated by the existence of the relevant files. Notes on creating these files can be found in the "McuWIN User Interface" manual.

## 3 Description of G-codes implemented

### 3.1 G00 – Rapid positioning

The target positions can be given as absolute or relative coordinates, depending on the mode selected (G90 / G91). Velocity and acceleration are the axis-specific default values entered in MCFG (File – System Data – tab "Motion Parameters" – {jvl} and {jac}), or the values set in the initialization task. The position units for G00 are always the axis-specific units set in mcfg, regardless of the choice of interpolation unit.

This command is persistent, i.e. G00 does not need to be included in every subsequent line. After continuation lines, the command recognized as persistent is the last command in the source code preceding the relevant line, not the command last executed.

#### Example:

```
N0400 G00 X20 Y22  
N0410 X30 Y23 Z-5
```

### 3.2 G01 – Linear interpolation

Execution of a linear interpolation with all positioning axes: the target positions can be given as absolute or relative coordinates, depending on the mode selected. The track speed can be defined e.g. with the F command. The acceleration is programmed in the initialization routine for a SAP task (0, 1 or 2).

This command is persistent.

### 3.3 G02 – Circular / helical interpolation

Circular or helical interpolation, clockwise: The target positions are given as absolute or relative coordinates, depending on the mode selected. The track speed can be defined e.g. with the F command. The acceleration is programmed in the initialization routine for a SAP task (0, 1 or 2). The target coordinates for the arc are entered as parameters. The required center coordinates are entered as parameters I and J. The center coordinates may be entered as relative coordinates even in absolute mode (G90). This option can be selected in the configuration module.

Command G02 is persistent.

### 3.4 G03 – Circular / helical interpolation

Circular interpolation, anti-clockwise: otherwise, as G02.

### 3.5 G04 – Dwell time

Dwell time in seconds. Here, floating point numeric values or variables or arithmetic expressions can be entered.

**Examples:**

```
N100 G04 3  
N100 G04 CD601
```

**Warning:** With G04, a possible maintenance of G-commands is cancelled.

### 3.6 G17 – Plane selection

Select X-Y plane for circular interpolation and tool radius correction.  
The X-Y plane is the default value after system startup.

### 3.7 G18 – Plane selection

Select Z-X plane for circular interpolation and tool radius correction.

### 3.8 G19 – Plane selection

Select Y-Z plane for circular interpolation and tool radius correction.

### 3.9 G21 – Reflection of Y axis

The coordinates programmed for the Y axis are reflected about the zero point. Any reflection of the X axis already active will not be affected by this command.

### 3.10 G22 – Reflection of X axis

The coordinates programmed for the X axis are reflected about the zero point. Any reflection of the Y axis already active will not be affected by this command.

### 3.11 G23 – Reflection of X and Y axes

The coordinates programmed for the X and Y axes are reflected about the zero point.

### 3.12 G24 – Disable reflection of all axes

All programmed reflection functions (G21, G22, G23) are switched off.

### 3.13 G39 – Program positioning factor

A positioning factor can be programmed for every reflected axis in the system. Where axis reflection is selected, the positioning values for the axis concerned will be multiplied by this factor. The default value is -1 (axis reflection). Command G39 is thus only valid in combination with a specified axis reflection (G21, G22 or G23).

.

**Example:**

N200 G39 X2 Y2

When G23 is active, the contour size will be doubled.

N500 G39 (Reset all positioning factors to -1)

### 3.14 G40 – Cancel tool radius correction

Switch off tool radius correction.

### 3.15 G41 – Tool radius correction, left

Switch on tool radius correction, left.

### 3.16 G42 – Tool radius correction, right

Switch on tool radius correction, right.

### 3.17 G51 – Program effective radius

G51 can be used to program an axis-specific radius in the specified interpolation path units. This will be used in the interpolation calculation where rotatory axes are involved in translatory interpolation runs. This supports processing of the surface of a cylindrical object turned about a rotatory axis. The unit of measure for the position values with rotatory axes is then not a rotatory value but the interpolation unit. This value is then converted using the radius entered into the angle to be traversed.

**Example:**

```
N200 G51 C120 D40
```

For this functionality, please refer to the version information in Chapter 14. Version dated 16.07.2003 required as a minimum.

### 3.18 G53 – Disable zero offset

Switch off zero offset for all interpolation axes. This is not a spooler command and so cancels any associated contour programmed with spooler commands such as G01. If zero offset needs to be switched off for axes other than the interpolation axes, the G92 command can be used in absolute mode, for example.

**Example:** N0100 G53

This instruction is equivalent to the following sequence of commands (for a 3-axis system):

```
N0100 G90          ' Absolute path information
N0101 G92 X0 Y0 Z0 ' Set zero offset to 0 for X and Y
```

### 3.19 G54 - G58 – Set zero offset

This command can be used to set the zero offset for all interpolation axes to the values in the associated zero offset table. The zero offset table is stored as part of the system data and can be edited with the commissioning program mcfg (zero offset). With G54, record 0 will be selected, with G55 record 1, etc.

To set the zero offset to any desired value, command G92 should be used. This is not a spooler command and so cancels any associated contour programmed with spooler commands such as G01.

### 3.20 G60 – Define interpolation axes

Selection of the axes involved in interpolation commands (G01, G02, G03).

**Example:**

G60 X Y Z

This command defines the axes associated with the interpolation processed when the above commands are used. These details are used for initialization and are generally retrieved only once at program start.

If the interpolation axes are to be changed while the program is running, this can also be changed in a `rw_SymPas` task, e.g. in Task 0 as part of an M or G command. The least significant bits of the ***IPOLMODE*** system variables are to be written on here. Please see the relevant documentation “McuWIN User Interface”.

### 3.21 G70 – Measurement in inches

Measurements for positions and velocities in inches.

### 3.22 G71 – Measurement in mm

Measurements for positions and velocities in mm (default value).

### 3.23 G74 – Reference run

Command G74 is designed to trigger axis referencing. This function has to be programmed / modified in Task 0 on an application-specific basis.

### 3.24 G90 – Absolute distance mode

Select absolute distance mode.

### 3.25 G91 – Relative distance mode

Select relative distance mode.

### 3.26 G92 – Set zero offset

Program zero offset for the specified axes to any desired value. This is not a spooler command and so cancels any associated contour programmed with spooler commands such as G01. This is a maintained command.



**Example:**      N0100 G92 X100 Y-200

If G90 has been programmed, the zero offset will be relative to the absolute zero point. If G91 has been programmed, the zero offset will be relative to the zero point currently set.

A G53 call will reset the zero offset, but only with the selected axes (interpolation axes). Calling G54 – G58 will cancel a zero offset set with G92 and set a zero offset according to the table.

A G92 call without a positioning command will not cause any movement of the axes.

In rw\_SymPas, the currently set zero offset can be accessed using the axis qualifier **ZEROOFFSET**.

### 3.27 G93 – Reverse-time feed encoding

Reverse-time feed encoding. This function is not yet implemented, and should not be used.

### 3.28 G94 – Time unit in minutes

Time unit given in minutes, i.e. feed unit in [mm/min] or [inches/min].

G96 – Activate constant cutting speed

G97 – Deactivate constant cutting speed

### 3.29 G98 – Set position of software limit switch - left

The software limit switches are not activated with this command. They are activated automatically when the respective axes are referenced, assuming this functionality has been defined in mcfg.exe. If this command is not used, the limit switch position will be initialized to the default position entered in mcfg.exe.

### 3.30 G99 – Set position of software limit switch - right

As section 3.29, but for software limit switch, right.

### 3.31 G150 – Spline off

Deselect spline interpolation.

### 3.32 G151 – Spline on

Select spline interpolation.

### 3.33 G153 – Read zero offset

This command can be used to read the current absolute position of the zero offset for all axes in the system. The values returned will be stored in the system variables CD[50 + axis index] in the user-specific interpolation unit.

**Example:**

```
N100 G153 ' Read zero offset in CD50 ff.
```

### 3.34 G154 – Reprogram zero offset returned

This command can be used to reprogram the position values for the zero offset for any number of axes read in via G153. The values are taken from the system variables CD[50 + axis index].

**Example:**

```
N100 G154 ' Set zero offset (absolute)
```

### 3.35 G161 – Center coordinates, relative or absolute

When executing circuits in absolute mode (G90), the center coordinates are also given as absolute coordinates. This option can be set in the configuration program IniCfg.

### 3.36 G162 – Center coordinates always relative

Center coordinates are always given as relative coordinates.

## 4 M-codes

The M-codes listed below are present in McuWIN. Further M-codes can be defined by the user. Information on these can be found in the manual "McuWIN User Interface".

Comments in round brackets are not allowed after M-codes, as M-codes can be called with parameters. These parameters may also be expressions in round brackets.

### 4.1 M00 – Program stop

Program stop, spindle, coolant and feed off: Setting/resetting the associated digital outputs must be implemented/modified in Task 0. In step mode, the instruction switches over. The program can be restarted from the point at which it stopped.

### 4.2 M01 – Optional stop

Optional stop: if the "Optional stop" button has been pressed, this instruction switches over to step mode.

### 4.3 M02 – End of program

End of program. The program stops. No further outputs will be changed!  
This command does not trigger any call to Task 0. See also [section 4.13].

### 4.4 M03 – Spindle clockwise

This command is dependent on the type of machine and signifies e.g. <<Spindle on, clockwise>>. The relevant functionality must be programmed in Task 0.

### 4.5 M04 – Spindle anti-clockwise

This command is dependent on the type of machine and signifies e.g. <<Spindle on, anti-clockwise>>. The relevant functionality must be programmed in Task 0.

## 4.6 M05 – Spindle stop

This command is dependent on the type of machine and signifies e.g. <<Spindle stop>>. The relevant functionality must be programmed in Task 0.

## 4.7 M06 – Change tool

This command signifies <<Trigger change of tool>>. The relevant functionality must be programmed in Task 0.

## 4.8 M08 – Coolant on

This command signifies <<Coolant on>>. The relevant functionality must be programmed in Task 0.

## 4.9 M09 – Coolant off

This command signifies <<Coolant off>>. The relevant functionality must be programmed in Task 0.

## 4.10 M17 – End of sub-routine

End of sub-routine and return to calling program. From version 2.5.3.41, this command no longer causes an intermediate stop in an interpolation run.

## 4.11 M26 – Set output

Set digital output.

Syntax: M26 Parameter

Parameter is a 3-digit number, with the first digit indicating the axis number and the second and third digits the output to be set. The value 00 in the second and third digits causes all outputs for this axis channel to be set.

**Example:**      N0100 M26 102      ‘ Set output 2 for axis 1

This command is a spooler command and does not cause any intermediate stop when called during an interpolation run.

#### 4.12 M27 – Reset output

Reset digital output.

### Syntax: M27 Parameter

Parameter is a 3-digit number, with the first digit indicating the axis number and the second and third digits the output to be reset. The value 00 in the second and third digits causes all outputs for this axis channel to be reset.

**Example:** N0100 M27 102 ' Reset output 2 for axis 1

This command is a spooler command and does not cause any intermediate stop when called during an interpolation run.

### 4.13 M30 – Program stop

Program stop with reset, spindle, coolant and feed off (similar to M00): setting/resetting the associated digital outputs must be implemented/modified in Task 0.

Program execution is stopped and the program pointer returned to the start of the program. Task 0 is then called. Additional functions have to be processed at this point. The program can then be restarted with `contcnct (3)`.

#### 4.14 M80 – Write to outputs (word by word)

This command can be used to write to an output register word by word. Note that output information updated by the system (e.g. PAE output) may be overwritten. Note also that the outputs for a given axis group can only be set together.

**Example:**

N220 M80 X 4      ‘ For the X axis, the 3<sup>rd</sup> output is set; all other outputs are  
‘ reset.

#### 4.15 M96 – Unconditional jump

Unconditional jump to the label specified. Labels are entered with the prefix L. From version 2.5.3.41, this command no longer causes an intermediate stop in an interpolation run.

**Example:**

```

N220 M96 L114          ' Unconditional jump to label 114
N221 ...
...
N348 L114              ' Label 114
N349 ....

```

**4.16 M98 – Call sub-routine**

Sub-routine call when L label specified. An O counter can optionally be used to enter the number of sub-routine cycles.

**Example:**

```

N220 M98 L114 O5        ' Call sub-routine 114 (5 times)
N221 ...                ' Further program
...
%114                   ' Sub-routine 114
N100 ....              ' Sub-routine body
N900 M17               ' Sub-routine end
%

```

Sub-routines can be inserted locally into the relevant source code after the end of the program module. Global sub-routines can also be added to the source code in include files. Further details can be found in Chapter 9.

From version 2.5.3.41, this command no longer causes an intermediate stop in an interpolation run.

**4.17 M100 – Reset “Program end” output**

Application-specific command: Reset “Program end” output. This function must be programmed in Task 0.

**4.18 M150 – Start recording for graphical systems analysis**

Calling this command records the change of position in the first three axes over 5 seconds. The change of position can then be displayed in mcfg using the graphic screen.

**4.19 M901 – Save common-integer-variable as resident**

An integer variable for the common-integer range can be saved as resident using this command. This information can then be restored at any time using M902, even after restarting the program. Information is saved in the INI file of McuWIN.

The index of the variables to be saved must be specified as the parameter (0..999).

**Example:**

N020 M901 601

**4.20 M902 – Read back common integer variable saved as resident**

Using this command, an integer variable in the common integer range, which was previously saved with the command M901, can be restored. If the variable has not been saved previously, then this command sets it to 0. If common variables are protected with indices < 600, these cannot be restored. The index of the variables to be saved must be specified as the parameter (0..999).

**Example:**

N020 M902 601

**4.21 M903 – Save common double variable as resident**

With this command, a floating point variable in the common double range can be saved as resident. This information can then be restored at any time using M904, even after restarting the program. Information is saved in the INI file of McuWIN. The index of the variables to be saved must be specified as the parameter (0..999).

**4.22 M904 – Read back common double variable saved as resident**

With this command, a floating point variable in the common double range, which was previously saved with the command M903, can be restored. If the variable has not been saved previously, then this command sets it to 0.0. If common variables are protected with indices < 600, these cannot be restored. The index of the variables to be saved must be specified as the parameter (0..999).

**4.23 M910 – Switch off PC after program end**

Since Windows XP, using M910 before the end of the program, the PC can be switched off after terminating the G-code program.

Example:

```
....  
M901  
M30  
%
```

## 5 Other codes

### 5.1 Labels

Labels are used to identify sub-routines and jump destinations, and are entered as numbers prefixed with the letter L. The definition of a label does not cause an intermediate stop in an interpolation run. The module name of a sub-program is also held as a label, and can be used for sub-program calls.

**Example:**

```
N220 L114      ' Label 114
%UP114        (Sub-program UP114)
...
M17
%
```

### 5.2 Special functions supported

Instruction code	Instruction no	Parameter	Comments
S	1	Parameter in CD0	Main spindle revolutions
T	2	Parameter in CD0	Tool change

These codes (instruction numbers) are passed to Task 0 in variable CI2 and executed as sub-routines.

### 5.3 F-command

Set feed velocity for G01, G02, G03 commands in the unit selected in the initialization tasks. If a line with a G-command also contains an F-command, the newly-programmed velocity will apply for this profile. Any braking of the axis will be executed in the preceding profile section. Any acceleration of the axis will be executed in the current profile. This instruction does not cause an intermediate stop in an interpolation run.

### 5.4 S-command

Setting the main spindle revolutions: This functionality must be programmed in Task 0. The function code for this command is 1. The parameter is passed in CD0. The S-command programmed in Task 0 interrupts the interpolation contour and executes the preceding lines using the SSMSIW command. This is a command specifically for this purpose, which is not used in standard rw\_SymPas programming.



If S-commands are to be implemented that do not interrupt the interpolation contour, this command must be omitted. In this case, appropriate spooler commands must be executed to control the main spindle of the processing tool (SSF).

## 5.5 T-command

Tool change: This functionality is programmed in Task 0. The function code for this command is 2. The parameter is passed in CD0.

To correct the tool radius, this command is used to select a correction record defined in the tool table. By default, tool no 0 is selected at system startup. A tool selection is only possible if a tool table with contents has been loaded into the control at the start of McuWIN, i.e. if the tool file (default name: ToolComp.ini) really contains tools.

## 5.6 D-command

Select tool correction table: This parameter is currently ignored and so not evaluated. This functionality must be programmed in Task 0. The function code for this command is 3 (in CI2). The parameter is passed across in CD0.

## 5.7 String output

The commands WRITE or WRITELN allow display information to be generated while an application program is running. This means that the display string is placed over McuWIN in a closable window. By outputting a space character or ' ', this window can be automatically closed again.

From McuWIN V2.5.3.101, it is possible to close out of each task the window of this or another task by outputting the string '!0' or '!3', with the number representing the index of the task number whose window is to be closed.

### 5.7.1 WRITE command

This command is used to generate a string for display and, where applicable, to attach it to a string that is not yet closed and will not be closed after this, so that further outputs can still be attached.

Parameters may be string constants, integer, double und Boolean variables as well as arithmetic expressions.

#### **Example:**

```
WRITE "current position: " X.rp
```

### 5.7.2 WRITELN command

This command is used to generate a string for display and, where applicable, to attach it to a string that is not yet closed but will be closed after this, so that further outputs cannot be attached.

Parameters may be string constants, integer, double und Boolean variables as well as arithmetic expressions. The output of a space character closes the output window.

**Example:**

```
WRITELN "current position: " X.rp
```

## 6 Comments

Comments can be enclosed in round brackets. It is also possible to mark the end of a line as comments with a single quote. Comments can be nested.

### Examples:

```
N210 G00 X10 Y-20   (This is a comment)
N220 G90           ' This is a comment
```

**Warning:** Round brackets can still be used within expressions without them being treated as comments. In the example below, the contents of the brackets are **not** a comment.

```
N230   G01 X(CD601 - Y.rp)
```

An arithmetic expression is also expected wherever a parameter is possible, e.g. after M-commands. In this case, the compiler error 2074 is shown.

In case of doubt, marking comments with a single quote is preferred. Comments must not be nested.

## 7 Conditional program execution

Based on Boolean values, sections of programs can be executed conditionally. The standards for expressions and operators are based on APCI-8001 SAP programming. With these standards, it should be ensured that program instructions are concluded with a semi-colon (not begin) and that comments are marked with // instead of with a single quote, or with curly brackets instead of round brackets.

### **Example:**

```
N350 $if (expression) then begin
N400 ..... (instructions)
N450 $end
```

The (expression) is a Boolean expression. Some examples:

```
(CI600 < 20) // Result of a comparison
BOOLEAN (CI600) // Conversion of a numeric value
(X.digib.5) // Query digital input 5 for the X axis
```

The conditionally executable program block must be terminated with \$end. If required, a \$else branch can be declared.

### **Example:**

```
N350 $if (expression) then begin
N400 ..... (instructions)
N450 $end else begin
N500 .... (instructions)
N550 $end
```

It is also possible to construct if-else-if chains.

### **Example:**

```
N350 $if (expression 1) then begin
N400 ..... (instructions)
N450 $end else if (expression 2) begin
N500 .... (instructions)
N550 $end else if (expression 3) begin
N600 .... (instructions)
N650 $end else begin
N700 .... (instructions)
N750 $end
```

## 8 Loops

### 8.1 Do-while loop

**Example:**

```
N350 $while (expression) do begin
N400 ..... (instructions)
N450 $end;
```

(Expression) must return a Boolean value (see IF). The loop will be executed until (expression) is false. The value of (expression) is checked at the start of the loop, i.e. if the value is false when the loop is reached, the loop is by-passed.

### 8.2 Repeat-until loop

**Example:**

```
N350 $repeat begin
N400 ..... (instructions)
N450 $end until (expression);
```

(Expression) must return a Boolean value (see IF). The loop will be executed until (expression) is true. The value of (expression) is checked at the end of the loop, i.e. the loop is executed at least once.

### 8.3 For loop

Examples:

```
N350 $for CI600:=1 to 10 do begin
N400 ..... (instructions)
N450 $end;
```

```
N750 $for CI600:=10 downto 1 do begin
N800 ..... (instructions)
N850 $end;
```

The run-time variable (here CI600) must be an integer variable. The definition of the “for” loop includes a start and an end value. Start and end values may also be expressions.

## 9 Integrating include files

The \$I instruction can be used to incorporate include files into a source code. Include files may contain further include files. The file name may also contain a drive and path information. The file name and the path information must not contain a space character.

### Example:

\$I UP2.SRC

This function can be used to insert a series of sub-routines consisting of standalone programs into a piece of source code.

### Example:

(in the main program)

```
N400  M98 L2000 O3      (Call sub-routine L2000 3 times)
N500  ...
N900  M02              ' End main program
%
```

(Definition of a local sub-routine)

```
%107
N0100 L107              (Local sub-routine 107)
N0200 ...              (Program body)
N0300 M98  L1000 O2      (Call sub-routine L1000 2 times)
N1000 M17              ' Return
%
```

\$I Unterprogramme.inc (Include library of local sub-routines)

The file **Unterprogramme.inc** may contain e.g. the following text:

```
$I UP1.SRC  (Module 1000: Bore out keyhole)
$I UP2.SRC  (Module 2000: Add tapped hole for compressed air)
$I UP3.SRC  (Module 3000: Bevel top edge)
```

Another example of the global sub-routine UP1.SRC:

```
%1000              (Module name = Label)
N200  G00 X0 Y0 Z0
N300  G00 X10 Y10 Z-10
N400  M17          ' Return from sub-routine
N500  %
```

## 10 Incorporation of rw\_SymPas commands

In programs conforming to DIN 66025, rw\_SymPas source code elements can be incorporated. In this case, DIN mode is terminated with the instruction \$DINEND.

**Example:**

```
N200 $DINEND
```

After this instruction, rw\_SymPas instructions can be inserted without line numbering. To switch back to DIN mode, the compiler instruction {\$DINSTART} is used.

**Example:**

```
{ $DINSTART}  
N220 G00 X10
```

It is also possible within a G-code program to convert a single program line with the \$ character to rw\_SymPas syntax.

Example:

```
$wt(2000);
```

Before a G-code program starts (before the first % character), an rw\_SymPas program block can be included, in which variables, constants, labels and procedures are declared. These objects can then be used in the G-code program with the mechanisms described above. Note that constant and variable names in G-code programs must not contain any numeric characters.

## 11 Calculation parameters

For mathematical operations like calculating target positions, calculation parameters can be used. These calculation parameters can also be used to decide on conditional jump functions. Calculation parameters are referenced as CD<n> for floating-point numbers and CI<n> for integer values. Values for n of 600 – 699 are reserved for the user.

**Warning:** The system also allows other values of n. However, any values outside the specified range may be system variables, and using them could cause instability in the system.

**Example:**

```
N220  CD620 := 100
N230  G00 X2*CD620
```



## 12 Variables

By switching to the `rw_SymPas` syntax, variables which can be used in the G-code program can also be declared. A sample program is listed below. This program is included in the scope of delivery.

```
' G-Code sample program
' - Declaration of variables and constants
' - use of mathematical functions in G-Code programs

$DINEND
// rw_SymPas program block for the declaration of variables}

var
    sRadius, step : double;      { floating point variable }
    cntr           : integer;    { integer variable }

const
    Loopcntr = 500;              { constant }

{$DINSTART}

%SAMPLE_PROGRAM
G90
G00 X100 Y200 Z-10
Z-40
step := 0.5                      ' Assignment to a variable
G90 F20
$for cntr := 0 to Loopcntr do begin // for-loop in rw_SymPas syntax
    C1601 := cntr
    C1602 := Loopcntr
    G01 X(X.tp + step) Y (200.0 + 5.0 * sin (X.tp / 10.0)) ' Use of the sine function
$end;

M30
%
```

From McuWIN V2.5.3.90, an initialisation area of this kind (shown in bold) can be swapped out via PreInclude file to an area which cannot be seen by the user. Likewise, an area attached at the end can be swapped out via PostInclude file.

## 13 Handling McuWIN

### 13.1 Troubleshooting

#### 13.1.1 McuWIN error messages

Any status or error information is made visible to users in the McuWIN error window. Here, a distinction must be made between compilation errors and runtime errors. Compilation errors are displayed while the program is being compiled or started. Runtime errors occur during program processing. Errors can partly be acknowledged by clicking on the error window. In most cases, the program needs to be reset. This means that referencing of the system is also cancelled. In general, the cause of the error, e.g. in the program source text, must be rectified.

##### 13.1.1.1 Error # 1: Unknown function code!

An undefined command (e.g. G or M) is used in the program. Under certain circumstances, the command interpreter must be adapted on an application-specific basis.

##### 13.1.1.2 Warning # 1: Encode error!

The encoder error flag of at least one axis was detected. In this case, the exact position of the axis affected can no longer be guaranteed. This flag indicates a hardware or cabling problem.

##### 13.1.1.3 Error # 2: Lag error (axis ?)!

It was detected that the predefined maximum lag error was exceeded on one axis. The axis causing the error is shown.

##### 13.1.1.4 Status #2: Wait until door is closed!

Wait until the safety door has been closed.

##### 13.1.1.5 Error # 4: Hardware limit switch (left axis? / right axis ?)!

At least one hardware limit switch was detected. Axis channel and right/left is specified in the message. This monitoring is only active if the system is fully referenced since moving towards limit switches during referencing can be fully intentional.

##### 13.1.1.6 Status #4: Program was started!

An application program was started! – normal operating state

#### 13.1.1.7 Error # 8: Software limit switch (left axis ? / right axis ?)!

At least one software limit switch was detected. Axis channel and right/left is specified in the message. This monitoring is only active if the system is fully referenced since it is only worthwhile monitoring a position on a referenced system.

#### 13.1.1.8 Status #8: Program was ended!

An application program was ended! – normal operating state

#### 13.1.1.9 Status #10: Emergency stop

Using a digital input declared as EO, a program which is running is aborted and prevented from restarting. This error is automatically acknowledged if the relevant input is inactive. In this way, the control unit responds to the emergency stop signal. Here, it should be noted that the emergency stop state will always have a direct impact on the power electronics in order to meet the relevant security instructions.

#### 13.1.1.10 Warning # 10: Position error detected!

Position monitoring has diagnosed a position error via a zero trace signal and indicates a problem with the position measuring system. However, the correct configuration of position monitoring is required for a reliable display.

#### 13.1.1.11 Error # 20: Conflict in configuration data!

McuWIN is set up incorrectly. This error can have different causes and, for example, points to the declaration of non-permitted axes or the specification of 0 in standard velocity or acceleration values. Another possibility is incorrectly defined software end limit switches.

#### 13.1.1.12 Status #20: Referencing was started!

The reference run was started! – normal operating state.

#### 13.1.1.13 Error 21 when loading <<??>>

This compilation error can occur if a compiled program (.cnc file) is to be loaded on the control unit, e.g. after clicking on the start or stepper button. In this case, the desired file cannot be loaded on the control unit.

This error sometimes occurs with very large files which, for example, are generated automatically from CAD data. By default, CNC files up to a size of 100,000 bytes can be processed (size of compiled file with the extension .CNC). If larger files are generated, the control-specific environment variable "SZTSK3" can be set to a higher value (max. approx. 12 Mbytes). If this size is still not sufficient, then the files should be generated with a lower resolution.

#### 13.1.1.14 Error # 40: Error with reference run (axis ?)!

Runtime error during referencing. Can occur if the conditions for the reference run are set incorrectly, for example.

#### 13.1.1.15 Status #40: Wait until spindle speed is reached...

Wait until the required speed has been reached. – Normal operating state, but may also indicate a problem with the spindle or spindle feedback.

#### 13.1.1.16 Error # 80: Version conflict McuWIN - SAP!

The versions of McuWIN.EXE, RWMOS.ELF and/or the task programs are not compatible. An update has not been carried out correctly. Under certain circumstances, the task programs need to be recompiled. It may be necessary to reboot the PC before restarting so that all programs are reinitialised.

#### 13.1.1.17 Status #80: Error bit detected in ErrorReg! ? hex

Runtime error which shows an error bit in the control-specific register ErrorReg of the APCI-8001. 20 hex indicates a traverse profile without a traverse path, 200 hex shows that an error message is displayed in the fwsetup monitor screen. Further hex values are described in the Programming Manual (PM) for the control unit.

#### 13.1.1.18 Error # 100: Configuration error!

The control unit configuration data is not saved. This can, for example, occur after changes to the configuration or after replacing the control module. In this case, the system data should be saved in mcfg on a booted system.

#### 13.1.1.19 Status #100: Error bit detected in IFS register! ?? hex

An error bit was detected in the IFS register of the control module. A more exact diagnosis is possible using mcfg. The fault-free execution of programs can no longer be guaranteed.

#### 13.1.1.20 Error 200: Error in tool radius correction!

Due to a configuration problem, the tool radius correction entered an invalid state.

#### 13.1.1.21 Status #400: Feed block active (UI)!

Using a digital input declared as UI, the axis feed can be blocked. This error is automatically acknowledged if the relevant input is inactive.

#### 13.1.1.22 Error # 800: Amplifier not ready!

The DNR (Drive Not Ready) signal from at least one axis shows that a power amplifier is not ready.

#### 13.1.1.23      Status #1000: Runtime error in SAP task!

A runtime error has occurred in a stand-alone task which means that the system is no longer ready. The cause can be determined using mcfg. A system program error may be the cause of this. These errors cannot generally be cleared by the user himself.

#### 13.1.1.24      Status #2000: Parameter error at runtime!

Incorrect parameters have been passed to a command in the user program.

#### 13.1.1.25      Error # 4000: Invalid S-value!

The S-command was used with a non-permitted speed value.

#### 13.1.1.26      Error # 80000000: System was stopped! Reset required ...

The system was stopped by an undefined process. This may occur, for example, if the control unit is accessed by an application other than mcfg while McuWIN is in operation.

## 14 Version information

This specification is valid for

McuWIN.EXE	from version 2.5.3.111
IniCfg.EXE	from version 2.5.3.87
MCFG.EXE	from version 2.5.3.91
MCUG3.DLL	from version 2.5.3.90
NCC.EXE	from version 2.5.3.63
RWMOS.ELF	from version 2.5.3.116
ToolEdit.EXE	from version 2.5.3.8

### 14.1 Changes in earlier versions

#### Version 2.5.3.111

- TC tool management revised :  
ToolEdit V2.5.3.8  
Write/read access to ToolCompensation-Group 0: accesses the group of the actively selected level

#### Version 2.5.3.110

- Task programming of McuWIN: Constant MyTaskNr new for debug display, e.g. in loops
- Task1.src: Integration of appeo\_off.inc shifted as the closed control circuits may be required for this

#### Version 2.5.3.108

- Reading of IniFile independent of country settings (DecimalSeparator)  
IniCFG – V2.5.3.89  
ToolEdit - V2.5.3.7
- RegDisp V2.5.3.6: is now possible with read-only INI file
- T-parameter may now also be a variable

#### Version 2.5.3.107

- New message windows also added for Task 1 and Task 2
- While starting Task1, a message "Initialisation ..." applied
- All error messages and warnings are written in a protocol file so that deleted messages can be verified afterwards --> IniFile variable [MCU] ProtocolFileName: Here, a file name including drive and path has to be entered. The setting can be adapted only manually in the Ini file, not via IniCfg.
- Character size can be set in the editor via IniFile
- [EDITOR] EditorFontSize (default 10)
- The setting can be adapted only manually in the Ini file, not through IniCfg.
- Editing of error texts if axes are specified now out of Inifile
- [FEHLERTEXTE\_EXT] (ERRORTEXTS\_EXT) Additional information?????
- The setting can be adapted only manually in the Ini file, not through IniCfg.

#### Version 2.5.3.106

- Maintenance of commands is now cancelled by DINEND and DINSTART
- Final % was not properly detected in automatic record number assignment
- Store BoardType in C167 if a board has been successfully initialised

- Checks in Task1.src can be switched off through SwitchOffChecks register; bit declarations SwitchOffCheck\_xxx for this in GCode.inc
- Monitor the flags in case of APCI-8008 line break

#### Version 2.5.3.105

- McuWIN could produce a deadlock at the program start between 1/2/2013 and 27/2/2013 using RWMOS.ELF
- Relieve reference switch, repeat as often as required if necessary
- Error # hex 8000 0000 generated a negative entry in the Ini file --> display problem

#### Version 2.5.3.103

- Error displays amended from CI10 to CI10 and CI48: In CI48, user-specific error bits are now possible.  
For this, new section [FEHLERTEXTE48] (ERRORTXTS48) in mcuwin.ini

#### Version 2.5.3.101

- Close message window in McuWIN (Task writeln) through "!", with ? = Task No.

#### Version 2.5.3.100

- Tool radius correction and tool length correction revised:  
Before: Maloperation in case of tool length and selection of other than the G17 level  
ToolEdit V2.5.3.6 program revised: Now, also tools for all levels are possible  
In case of initialisation: Changes in zero offset and TC
- Use of G04 without parameters generated an exception in RWMOS –  
now compiler error 2075

#### Version 2.5.3.99

- In McuWIN.ini section [Project data]: With SelectedCardNr, other than the 1st board in the system can be activated.
- Compatibility information: McuWIN with WebServices and mcug3.dll must fit
- Improvements in establishing Web-Services connection: Timeout / additional error information in log messages

#### Version 2.5.3.96

- Message prevented: "A deactivated or hidden window cannot get the focus!"
- Undo button for TeachIn
- Safely reset manual process at program start
- TASK1: Hardware limit switch handling changed
- Close message of Task 0 in "Forts."
- In IniCFG, ToolFile can now also be switched off.
- G-code parameter analysis was partly incorrect.

#### Version 2.5.3.95

- Use InhibitProfileRefuse for warning display; acceleration or velocity = 0 now stops the program flow; so far, such a traverse profile has been rejected.  
**Caution: Incompatibility possible**

Version 2.5.3.94

- appeo\_off.inc has been integrated in wrong place before:  
**Caution: Compatibility problem possible after update if this file has been used!**
- M-codes no longer cancel maintenance of G-commands

Version 2.5.3.93

- Possibility to close text output windows
- M910 – Shut down PC after end of program

Version 2.5.3.90

- Option to specify preinclude and postinclude files in the G-code program
- No M02 / M30 / M17 in G-code programs results in error 190 during compilation
- Parameter information AXSEL and Numparams revised

Version 2.5.3.89

- Selection of processing levels (G17 / 18 / 19) only if real tool tables are also active

Version 2.5.3.88

- Compensation of spindle errors and angle errors implemented
- In the target position, the target position of the active traversing profile can be displayed as an alternative
- When positioning axes manually, an increment can now be specified.
- In the mask for manual positioning, the button can be used to move to the rest position.
- With the M901 – M904 commands, common variables can be written to the hard drive as resident and restored.
- If rw\_SymPas was used in G-code programs, the automatic initialisation of the interpolation axes was incorrect.
- Zero offset is deleted after stopping the program and restarting.
- It is only possible to end McuWIN if it is not in automatic mode.
- Internal handling of error and warning messages amended.
- Prefix S-command taken into account in changes to spindle speed.
- With manual positioning, stop no longer with sdec, but with jac, resulting in smoother behaviour during manual positioning
- Reset spindle speed target value at program start.
- Compilation of files and syntax check is now also possible in offline mode.
- Override correction during deceleration improved.

Version 2.5.3.82

- Additions to the use of web services, use possible under Linux/WINE
- New application-specific AppEO\_Off.inc file for additional design options when installing McuWIN.

Version 2.5.3.78

- G60, G90/91 are now taken into consideration at the program runtime and no longer during compilation. At least RWMOS V2.5.3.89 is needed for this.
- New option for cutting speed interpolation
- In single-step mode, positioning motions have not always been started safely.



Version 2.5.3.76

- Additional settings are possible in IniCfg.EXE:  
"Axis def." tab – open loop operation can be selected for individual axes here.  
"Hardware outputs" tab – an auxiliary voltage output can be defined here.
- After program start-up or after an error, any defined auxiliary voltage output must be manually set in a mask.
- The new file AppStartChecks.inc is integrated in Task one, which enables a one-off application-specific system check at the start.

Version 2.5.3.75

- For rotatory axes, the approach to the target position can be set in IniCfg.EXE to the shortest path ("Desktop / System" tab)

Version 2.5.3.61

- The program ToolEdit.exe, used to edit the tools table, writes the current status of the tools table as soon as it is saved to the control. Changes are therefore effective online.
- McuWIN.exe and IniCfg.exe: Treatment of the main spindle revised; e.g. the target and actual values for the spindle speed can now be displayed in McuWIN.

Version 2.5.3.52:

- D command now supported. This function can be programmed in Appcommands.inc under Function = 3
- Row trace improved; for G01/02/03 now always displays the row actually being run.
- Compiler error display improved
- Default path for Include files was the installation folder, now path for source text file
- Bug in SAP instructions using \$ operator fixed (\$end)

Version 2.5.3.47:

- Now possible to move to a rest position not equal to 0 after referencing
- Bug in the use of G01/02/03 after jumps and in sub-programs fixed
- Bug where AutoSetNum set and maintained commands used
- From RWMOS V2.5.3.47, hardware reference switches displayed in the axst register. This feature will be used in the reference run immediately. The minimum version has therefore been raised to 2.5.3.46.
- Problems with commands with multiple parameters fixed  
Round brackets not always recognised as comments (e.g. after G04)  
Spaces after L and O no longer required
- Reference run enhanced to include play in the reference switch.
- Improvements in relation to spooler processing, particularly when calling sub-programs and when executing G200.299 / M200..299
- Error in McuWIN fixed: the settings for the reference position were destroyed in some cases.
- Teach-In now possible without row numbers.
- Problems in the use of unary operators (+/-) fixed.

Version 2.5.3.41:

Jump commands M96, M98 and M17, and declaration of labels no longer interrupt a spooler contour (G01, G02, G03).

Version 2.5.3.40:

The reference positions set in IniCfg used to be transferred to the control via the so-called common buffer. This is now done via common variables CD60 – CD67. It is therefore vital to ensure that these variables are not used by the user.

Version 2.5.3.38:

- New commands WRITE and WRITELN for string output
- Display of string output in McuWIN UI
- Full integration of rw\_SymPas mechanisms into G-Code programs possible
- Definition of specific M-Codes now possible, with parameter control
- G04 parameter can now be passed via variable or expression

Version 2.5.3.32:

- S-command in same line as G-codes sometimes executed incorrectly  
Flow/requirements for S-command substantially changed (see doc.).

Version 2.5.3.31:

- Various changes in Editor display
- Main spindle switched off on program stop
- NoTriangle option allowed with Look-Ahead

Version 2.5.3.30:

- Case-insensitive mode now supported, configurable in IniCfg
- Program stop also possible in referencing and in single-step mode
- Position and time units for interpolation commands now selected in IniCfg
- New M150 command
- Cleared bug in G-code programs with no M30 at the end
- Module name no longer required in programs

Version 2.5.3.25:

When comments were used, translation failures sometimes occurred, particularly with comments in quotes in lines with G01, G02 or G03. It is now also possible to nest comments with round brackets.

Version 2.5.3.24:

- A velocity factor can be entered for freeing motions  
(FreeingVelFactor)
- Application-specific reference run can now be set in IniCfg.
- Configuration variable ReferenceSwitchLatch to disable check on reference switch latch
- Configuration variable NoRefToLimitSwitch to allow referencing only on the index track
- A customer-specific logo can now be incorporated into the program interface
- The need for line-numbers can be switched off in IniCfg

Version 2.5.3.23:

- New commands:

- G153: Read zero offset
- G154: Reprogram zero offset
- Zero offset, particularly G54...G59 revised,  
G59 no longer exists, as there are only 5 zero offset registers

- Fixed bug in the use of non-consecutive interpolation axes;
- Program can be started in simulation mode;
- Counter monitoring by index latch and encoder error monitoring now selectable independently;
- Fixed bug in override setting at start of reference travel;

Version 2.5.3.19:

New commands G98, G99, G161, G162.

Commands G94 and G70/G71 were still sometimes handled incorrectly. With the updated version, there may be changes in velocity and acceleration. After an update, this must be investigated and corrected if necessary. (25.06.2004)

Version 2.5.3.18:

From this version onwards, it is possible to enter relative center coordinates even in absolute mode (G90).

Version 2.5.3.16:

It is possible to enter a LookAhead depth in IniCfg. Once the number of corresponding interpolation records has been programmed (G01, G02, G03), processing begins automatically.

In single-step mode, processing also starts automatically. It is no longer necessary to click within the editor screen to start the axes. Splines are only projected as straight-line segments.

Version 2.5.3.15:

Commands G21 / G22 / G23 and G24 for axis reflections are new. Also the use of positioning factors with G39.

Version 2.5.3.14:

Command G54 now selects a zero offset from a zero offset table, as do G55 to G59. It is now possible to set the zero offset to any value with G92. The existing function of G54 has therefore been superseded by G92.