POSITIONING AND CONTOURING CONTROL SYSTEMS
APCI-8001, APCI-8008 and CPCI-8004

User interface installation

McuWIN

Edition: 6/052015

www.addi-data.com

1 Version information

The following description is valid for:

McuWIN.EXE	from version 2.5.3.122
IniCfg.EXE	from version 2.5.3.94
MCFG.EXE	from version 2.5.3.94
MCUG3.DLL	from version 2.5.3.105
NCC.EXE	from version 2.5.3.71
RWMOS.ELF	from version 2.5.3.123
ToolEdit.exe	from version 2.5.3.8

2 Installation

McuWIN runs under almost all 32-bit Windows platforms (Windows 95/98/ME, NT, 2000, XP, Vista and Windows 7). Please ensure that before the installation of McuWIN the programs Miniport, fwsetup and mcfg are installed. The installation of McuWIN is executed by calling SETUP.EXE. The user will be guided through the installation process.
You can uninstall the package again with the menu or with the system control.

3 Remarks for the installation

After the installation of McuWIN, the environment must be adapted to the user specific requirements and conditions. In the following the single steps are described. For additional information, required supplements, and any other needs we are at your disposal:

Phone: +49 7229 1847-0

E-mail: info@addi-data.com

3.1 Creating a User directory

With the installation, a user directory was created that contains the files System.DAT and RWMOS.ELF. Updates and additions must be copied into this directory. The program mcfg.exe must be adjusted to thes files under „File – Project Parameter“.
If mcfg was installed before installing McuWIN, the necessary settings are defined automatically.

3.2 Installation of the axes with mcfg

Before you can use the program McuWIN, the complete system must be installed and configured with the installation program mcfg.exe . The following characteristics are of special significance:

- Axis name
  Caution: When using McuWIN as G-Code interface, the axis names may not contain numeric values.
- Mechanic parameters
  (encoder resolution, gear factor)
- Setting of the bearing controllers
- Maximum position error
- Configuration of the inputs and outputs, limit switch , reference switches, amplifier enable etc. See also chapter „Hardware-preconditions“
- Default of the Jog- and Home-accelerations and speeds
- Stop-Deceleration
  (Target-Velocity must always be 0)
- Declaration of the software limit switch

The axes must be moved controlled with the Motion-Tools in mcfg. These values, which are set in mcfg.exe, will be stored in the indicated file SYSTEM.DAT during the storation. It is important that McuWIN.EXE also accesses to this file as system file.

Caution: If the file SYSTEM.DAT was copied from a CD, it could be necessary that you must remove the file attribute ReadOnly, so that you can store it.

3.3 First call of McuWIN

Now the program McuWIN is called for the first time and terminated at once. Herewith the MCUWIN.INI is created and modified. After this, this file will be adapted with the program IniCfg.exe .

3.4 Installing McuWIN

The application specific adaptation of McuWIN will be executed with the program IniCfg, which was installed with McuWIN. The executed settings will be stored in the configuration file McuWIN.INI. Manual modifications in the file McuWIN.INI are obsolete.However, chapter 5 explains the contents of the INI-file.

3.5 Displaying a bit variable

The also installed program RegDisp allows displaying, grouping and marking bit information on the screen. Bit variables are for instance: Digital inputs, outputs, contents of CI-variables or also I/Os of additional PCI-groups. This program is very helpful during the system development.
The parameterizing of the several register cards and display elements is realized by menues that are opened with a click of the right mouse key on the specific display element.

3.6 Editing programs

In the Edit-mode in the editor window McuWIN programs can be loaded, saved and edited. Before the execution of the program a syntax test can be done. The result of the syntax test will be displayed in the error window.

3.7 Executing programs

With the „Start“-button you can start the program that is active in the editor. The editor then is switched to the trace mode.

3.8 Executing programs step by step

If you have selected in the INF file correctly, the program can be tested with the „step“-button in gradual operation. At spooler movement sequences (G01, G02, G03) the actual movement will be executed firstly after the execution of the last line that belongs to the corresponding section. By a mouse click in the editor window the traversing movement can be started at an earlier point at time.

3.9 Stops in the program execution

With M00 absolute stops can be set. When reaching M00, McuWIN switches to single step operation and then can be continued with „Step“ or „Forts“ – button.
With M01 absolute stops can be set. These stops can be enabled with the mouse on the button “Desired stop” („Wahlweiser Halt“).

3.10 Teaching of position values

You can open this with the mouse button „Teach In Maske“. Here, the axes can be traversed manually via the mouse button. By clicking on the key “Teach”, the current position value is inserted at the cursor position in the source text file.
The displayed picture in the respective button can be changed in the file File McuWIN.INI in the section [TEACHMASK] (see also section [TEACHMASK]).

Note: The “Teach” button is only enabled in a referenced system.

Moreover, manual traversing through joystick is enabled via the display of the teach-in mask.
From V2.5.3.111, this functionality is contained in Task1.SRC. More information on this can be found in the following Chapter.

3.11 Manual traversing through analog joystick

For manual traversing through joystick or analog input signals, the user-specific configuration of some common integer variables in the range CI300 .. CI399 is necessary. These variables should be configured in AppStartInit.inc. To activate these entries, Task1.src always needs to be compiled and McuWIN.EXE restarted as this include file is integrated in Task1src.
In CI300 in bit 31, McuWIN.EXE shows if the teach-in mask is applied. Only then, manual traversing through analog joystick is enabled. The assignment of the analog input channels to the physical axis channels of the board is preset: 1st axis = channel 1, 2nd axis = channel 2, etc.
Traversing through analog joystick is not possible with the boards PA 8000 and PS840.

3.11.1 Initialisation of joystick traversing in AppStartInit.inc

First of all, traversing must be enabled through analog values. This is done in the variable CI300. Here, the respective axes must be entered in bit-coded form. Only axes 1 to 8 can be used.

Example:

```
CI300 := $7;      // Activate axes 1, 2 and 3 for joystick traversing
```

After that, the zero points of the single analog channels must be entered in the variable CI31x. The read-in values of the analog channels can be shown or determined in CI39x via the program CiShow.exe, for example, in the event that McuWIN is active and the respective channels in CI300 are enabled. To refresh these values, the "Teach-in mask" in McuWIN must be opened as well.

Example:

```
CI310 := 16300;
CI311 := 12800;
CI312 := 8635;
```

After these values have been entered in AppTaskInit.inc and been activated by compiling Task1.src and restarting McuWIN, the respective total deflection can be determined and entered in the variables CI32x.

Example:

```
CI310 := -3200;
CI311 := 2800;
CI312 := 5600;
```

After compiling Task1.src and restarting McuWIN once again, manual traversing through analog joystick is available.

3.12 Override

The Override function is configured in the INI file. With the shift controller, the Override can be set during the execution of the program. The Override value can also be set by an external source. This functionality must be programmed by the user, e.g. in AppTask2.SRC in a closed loop. This can be done for all interpolation axes via the system variable trovr and the procedure call utrovr.

3.13 Further remarks

After the first calling of McuWIN, the following values in particular should be set by IniCfg:

3.13.1 Used axes (usedAxis)

Bit coded numeric value of the axes used in the system:

- At these axes the control loop will be closed
- For these axes positioning display windows are generated
- These axes are taken in consideration at the Teach-In
- These axes can be traversed manually
- These axes are effected by the Override-function
- Firstly, when all these axes were referenced, hardware limit switches will be displayed as errors in McuWIN

3.13.2 Axes in the interpolation connection (InterpolationAxis)

Bit coded numeric values of the axes that are connected with the interpolation. Here, it is possible that are less axes than indicated in usedAxis.

3.13.3 Customer specific layout of the user interface

From version 2.5.3.24 on you can integrate a bitmap file into the status display window. Herefore you must copy a bitmap file named „LOGO.BMP“ into the McuWIN-directory. The optimal resolution of the of the file is 529 x 285 pixel with a screen resolution of 024 x 768 points. The relation of the pages shall be kept in order to avoid distortions of the pictures.

3.13.4 Errors and warnings

The SAP programming environment can set error states in the variable CI10 in bit-coded form. A change in this register results in a text output in the error window of the McuWIN user interface (Chapter 5.7). Thus, 32 different error displays are possible. From McuWIN V2.5.3.102, the Common Variable CI48 is added to the error display possibility. In CI48, error bits can be set as before in CI10. The respective error messages have to be implemented in the Ini file in the section [FEHLERTEXTE48], similarly to error displays for CI10.
In addition to the error states, warnings or status information can be displayed. For this, the variable CI15 is written on in bit-coded form (Chapter 5.8). With this method, a maximum of 31 different warning texts is possible. Bit 31 (8000 0000 hex) in CI15 is assigned for deleting the display in the error window.
When setting or resetting bits in these registers, please note that only the relevant bits have to be controlled through a logic operation:
Setting bits through disjunction (OR)
Resetting bits through AND relation (AND)

4 Hardware-preconditions

4.1 Main spindle

The main spindle is switched on or switched off with a digital output. The direction of the main spindle also is switched with a digital output. The assignment of these outputs must correspond with the programmed data in TASK0.SRC or in the corresponding Include-files.
Since McuWIN version 2.5.3.59, it has been possible to declare a spindle axis in IniCfg.EXE. If an axis is defined here, the main spindle processing described here is disabled.

4.2 Output end of program

When starting an automatic program, McuWIN calls the special command 1001 in Task 0. Herewith the output “Program – is running” is set. After the termination of the automatic program the output “Program - is running” is reset. This is done with the calling of the special command 1000 of McuWIN. The output “Program – is running” is 08 of the first axis channel as default.
With the first single step of a program of McuWIN the special command 1002 is called in Task 0. In this also the output ”Program – is running” is set.

4.3 Output cooling

The assignment of the output cooling must correspond with the data programmed in TASK0.SRC or in the corresponding Include-files.

4.4 Overview assignment of the outputs

Table: Assignment of the outputs of the **APCI-8001**

Axis	Input	Pin / Board / Connector
1	O1	
1	O2	
1	O3	
1	O4	Cooling On
1	O5	
1	O6	Direction main spindle
1	O7	Main spindle On
1	O8	Display program finish

The declarations of the outputs can be found in GCode.INC and can be modified, if necessary.
If the corresponding G and M commands are not to be used, or are to be used in a different form, the relevant commands must be re-declared in the Include files.

4.5 Reference switch

If the axis referenciation shall be executed on a reference switch, the latch inputs that correspond with the axes should be used as reference switch inputs.

Table: Allocation axis / latch input

Axis	Input	Pin / Board / Connector
1	I14	47 / APCI-8001 / X1
2	I15	48 / APCI-8001/ X1
3	I16	49 / APCI-8001/ X1
4	I30	47 / OPMF / X1
5	I31	48 / OPMF / X1
6	I32	49 / OPMF / X1
7	I39	15 / OPMF / X2 (SUB-D ext.)
8	I40	16 / OPMF / X2 (SUB-D ext.)

5 Variable in McuWIN.INI

The following settings are executed by the program IniCfg.exe and used by McuWIN.exe for the configuration of and adaptation to the application. A manual editing of this file is usually not required, because the following settings are edited in IniCfg.exe. However, you can use this section in order to know the several settings more comprehensively.

5.1 [DESKTOP]

In this section the appearance of the user interface is administrated. The values WindowState, Left, Top, Height, and Width save the position of the user interface and are administrated of McuWIN
Further parameters are described in the following part:

5.1.1 SingleStep

Marker for showing/not showing the singlestep button (Default: 1 = On).

5.1.2 Override

Marker for showing/not showing the Override-trackbar (Default: 0 = Off).

5.1.3 OverrideEnable

Marker, if the Override track bar can be modified in the user interface by the mouse (Default: TRUE). If the Override value shall be set e.g. in the task environment OverrideEnable=0. In this case the Override-track bar is only for display. For this, see also Chapter 3.12.

5.1.4 OverrideMax

Maximum value of the Override track bar in % (Default: 125). This value is also used in case of a possible spindle override.

5.1.5 OverrideMin

Minimum of the Override track bar in % (Default: 0). Here also negative values are possible. Then you can drive back within a traversing profile. This value is also used in case of a possible spindle override.

5.2 [EDITOR]

5.2.1 SrcFileName

In Parameter SrcFileName the name and path of the user task is saved. This parameter is administrated by McuWIN.

5.2.2 AutoLoadOnActivate

When this parameter is set on 1, the indicated source text file is loaded newly every time when activating McuWIN under Windows, as soon as the system is in the edit mode, i.e. when no user program is executed.

5.2.3 DisableEdit

When this parameter is set on 1, user programs can be loaded and executed, but not edited.

5.2.4 SrcFilter

Here default extension for source text files can be indicated. This extension will always be proposed after the first selection when opening a file (when a source text file shall be opened).

5.3 [MCU]

In this section the control specific parameters can be found.

5.3.1 AutoSetNum

If this marker is set (≠ 0), then the line numbers (Nxxx), which are normally required in the G-Code mode, are not needed anymore.

5.3.2 usedAxis

Indication of the actual used axes (bit coded). With this variable, the axes of the operation system, which are processed by McuWIN, can be limited (Default FFh)

Example: You have a system with 3 axes (X, Y and Z)
Then here the value 7 must be entered because here the three bits with the lowest value are set.

5.3.3 InterpolationAxis

Indication of the axes that are in the connection of interpolation!

5.3.4 EECheck

Indication of the axes (bit coded), at which the encoder-error is tested. In step motor systems this variable can be set on 0. With this variable also the position monitoring is activated by index-latch function.

5.3.5 FehlerVarCI, RefVarCI

Indication of the CI-variable for error display and for reference status display. These variables are preset by the programs TASK0.SRC and TASK1.SRC and may not be changed.

5.3.6 RefAfterEO

This marker indicates, if the system must be referenced newly after an emergency stop (default value=1).

5.3.7 BaseAdress

In MCUG2-systems (PA8000, PS840) here the base address of the control must be entered (Default = 768 = 300H).

5.3.8 SystemFileName

Path and name of the system file (Default: SYSTEM.DAT in the program directory).

5.3.9 BootFileName

Path and name of the boot file (Default: RWMOS.ELF in the program directory).
Caution: With the PA8000 and PS840, usually RWTOS.BTL has to be entered here.

5.3.10 Task?Aktiv

? is the index of task 0..3. With the respective marker it is indicated if the respective task shall be loaded at the program start. If yes, under ***Task?CncFileName*** the name of the file to be loaded is indicated with the path (Default: TASK?.CNC in the program directory).
In connection with Task 3, two more file names can be indicated: ***PreIncFileName*** indicates the name of an Include file which is included at the beginning of the G-code file. This Include file may contain variable and procedure declarations in rw_SymPas-Syntax that can be used in the program, but which are not visible to the machine user.
PostIncFileName indicates the name of an Include file which is included at the end of the G-code file. This Include file may contain declarations from subroutines in G-code syntax that can be used in the program, but which are not visible to the machine user.

5.3.11 TRAC, TRVL

Here the default values for the track speed and track acceleration for interpolation commands are entered. The units for these values shall be set in the monitoring task. If they should not be set there, mm/sec and mm/sec^2 are selected.

5.3.12 FreeingVelFactor

Free movements, e.g. at the referenciation of the axes, are executed normally with the rapid traverse – speed / acceleration (Jog-velocity, Jog-Acceleration). With this parameter, these values can be reduced or increased.

5.3.13 ReferenceSwitchLatch

In this marker the axes are entered bit coded, where the reference switch input is at the same time a fast latch input. If this is not the case or when the latch signal is at the moment a fast latch input. If this is not the case or if the latch signal recognition shall be checked, this can be deactivated axis specific with help of the marker.

5.3.14 NoRefToLimitSwitch

If not to one reference switch is referenced, the referenciation is executed on a limit switch. If this should be not wanted, this can be deactivated (bit coded) with the marker. In this case the referenciation is only to the index signal or there is no referenciation.

5.3.15 ReferenzUeberwachung (reference monitoring)

In this parameter it is indicated bit coded which axes must be referenced so that a program can be traversed (Default 0xFF);
As reference switch always the latch inputs of the respective inputs must be used (inputs 14, 15 and 16 at the first three axes).

5.3.16 UeberwachungsTask (monitoring task)

Here the task must be indicated, which monitors the system. If there is no monitoring task available, here a negative value must be indicated (Default: -1).
The monitoring task must always be activated with the task Task?Aktiv=1 and started automatically with the task Task?Autorun=1. With Task?CncFileName= the file name and path are indicated (Default: TASK?.CNC).

5.3.17 UserTask

Here the task must be entered in which the program shall be executed in the editor. Default is 0.

5.3.18 DINGCodes

With the value 1 for this parameter the option G-Code processing can be activated. Then the UserTask must be Task 3. Hereto a command interpreter task must be loaded in Task 0 and a monitoring task in Task 1.

5.3.19 UserTaskAutoRun

With this marker it is indicated if the UserTask shall be started automatically after the program start.

5.3.20 UserTaskAutoStop

With this marker it is indicated if the system shall be reset after finishing McuWIN. If yes, the file ***StopUserTask.CNC*** in the User-Task is loaded and started.

Caution: ***StopUserTask.CNC*** must be compiled for the correct task.

5.3.21 SProfile

Indicator, shows if the S-profile or the trapezoid-speed-profiles shall be traversed (Default = 0).

5.3.22 JerkRel

Factor for jolt at S-profiles (Default = 0.5):
0.0 = max. allowed jolt, corresponds trapezoid-speed-profile
1.0 = min. jolt, corresponds triangular acceleration phase
Values between 0 and 1 correspond trapezoid acceleration phase

Here a floating point value between 0.0 and 1.0 can be entered, if SProfile = 1.

5.3.23 LookAhead

Indicator, indicates if the Look-Ahead speed limitation is active (Default = 1). In the active Look-Ahead for the several axes the maximum allowed velocity jumps is indicated, which is indicated in A?.mdvel.

5.3.24 LookAheadDeep

Whole number variable indicates in how many inerpolation-traverse commands in the spooler the spooler processing begins. So this number is always the max. number of traverse profiles, with which a LookAhead calculation can be realized.

5.3.25 NoTriangle

Indicator! Indicates whether Look-Ahead profile sections in which the maximum speed is not reached (triangular profiles) should be speeded up, or whether they should be traversed at constant speed. This indicator can be used to prevent a rough “sawing” movement when running small profile sections with Look-Ahead monitoring. However, it does increase the positioning time.

5.3.26 A?.mdvel

An axis specific max. velocity jump in the LookAhead mode (Default = 0,1). For „?“ the respective axis number (1..n) is significant. The unit is the respective interpolation unit.

5.3.27 CenterPointRelative

Marker with which can be indicated if the circle centre coordinates are indicated as relative or absolute coordinates in the absolute mode (G90) .

5.3.28 RotatoricUnit

Marker with which can be indicated if rotatory axes (axes with a rotatory user unit) are programmed in the selected compiling unit or in the user specific rotatory unit when used within translatory defines interpolation travels (G01, G02, G03). The translation between translatory and rotatory units is realized with the axis specific effective radius, which can be programmed e.g. with the command G51.

5.3.29 ShortestRotatoricDist

Flag which can be used to indicate whether rotatory axes (axes with a rotatory user unit) in absolute mode should always take the shortest path to reach the end point. If this flag is set, an axis with a traverse path of 359 degrees would be traversed from 1 degree by 2 degrees anti-clockwise. If the flag is not set, the axis traverses 358 degrees clockwise.

5.3.30 CmdTpDisplay

The user interface shows the desired position of the relevant axes as the target position. If the CmdTpDisplay flag is set, the target position of the current profile is shown as the target position.

5.4 [TEACHMASK]

In this section each traversing button can be assigned to a bitmap.

5.4.1 GlyphLeftX=Y

This command assigns the buttons of the left column to a picture. X is the line of the button (counting from 0 upwards). Y is the index of the available buttons (according to the table).

0	1	2	3	4	5	6	7	8	9	10	11	12	13
													

5.4.2 GlyphRightX=Y

This command assigns the buttons of the right columns to a picture. X is the line of the button (counting from 0 upwards). Y is the index of the available buttons (according to the table).

0	1	2	3	4	5	6	7	8	9	10	11	12	13
													

5.5 [SYSTEM]

5.5.1 CompileAlways

If this marker is <> 0 at each program start the active SRC-file is compiled firstly and then loaded to the control. If CompileAlways = 0 is set, it is firstly checked, if the CNC-file already exists. If yes, only the SCRC-file is compiled, if the date of the SCRC-file is more current than the date of the CNC-file.
Herewith in a lot of cases time can be saved for the (redundant) compilation process. This is especially useful with slow PCs or large programs.

5.5.2 RebootEnabled

With this marker it can be controlled, if the system is rebooted by double-clicking on the error window.

5.5.3 Axis compensation

With this marker axis compensation can be loaded. Hereto it is accessed to a separate INI-file, whose name and paths are indicated in *KompensationsFile* (Default: CP.INI).
This INI-File can be created with the program WrParams.EXE. If this option is used, RWMOS must be used with the option GEAR.
Caution: In the Ini-file the path must be indicated, otherwise the file cannot be found!

5.5.4 HWStart(n)

With the marker HWStart one or several digital inputs can be indicated by bits, at which a hardware program start can be realised. The number of the axis channel is n. This function is only effective, if in the editor are no modifications available that were not saved.
Selecting bit 16 causes bit 0 to be referenced by CI63 (1 hex) at program start-up.

Example: Input 2 of the X-axis (first axis) shall start the user program:

HWStart1=2

5.5.5 HWStop(n)

With the marker HWStop one or several digital inputs can be indicated by bits, at which a hardware program start can be realised. The number of the axis channel is n.
Selecting bit 16 causes bit 3 to be referenced by CI63 (8 hex) at program start-up.

Example: Input 3 of the X-axis (first axis) shall stop the user program:

HWStop1=4

5.5.6 HWRef(n)

With the marker HWRef one or several digital inputs can be indicated by bits, at which a hardware start of the reference travel can be realised. The number of the axis channel is n.
Selecting bit 16 causes bit 2 to be referenced by CI63 (4 hex) at program start-up.

Example: Input 3 X-axis (first axis) shall start the reference travel.

HWRef1=4

5.5.7 HWReset(n)

With the marker HWReset one or several digital inputs can be indicated by bits, with which a hardware-activation of the reset-buttons can be realised. The number of the axis channel is n.
Selecting bit 16 causes bit 1 to be referenced by CI63 (2 hex) at program start-up.

Example: Input 4 of the X-axis (first axis) shall cause a reset:

HWReset1=8

5.5.8 HWSingleStep(n)

With the marker HWSingleStep one or several digital inputs can be indicated by bits, with which a hardware-activation of the SingleStep-buttons can be realised. The number of the axis channel is n.
Selecting bit 16 causes bit 6 to be referenced by CI63 (40 hex) at program start-up.

Example: Input 5 of the X-axis (first axis) shall cause a reset:

HWSingleStep1=$10

5.5.9 HilfsSpannungEin(n) (auxiliary voltage on(n))

Inicfg.exe page “Hardware outputs”, line “Hilfsspannung Ein:” (auxiliary voltage on)

With the HilfsSpannungEin (auxiliary voltage on) flag, a digital output can be specified with which a hardware system start can take place. n is the number of the axis channel (starting from 1). If an output is specified here, a screen mask will be shown in McuWIN after starting or after all errors in CI10 have been cleared. In this mask, the specified digital output can be set using a button. As soon as an error is displayed in CI10, this output is removed again.

Example: Output 2 of the X axis (first axis) should be used to switch on the auxiliary voltage.
HilfsSpannungEin1=2

5.5.10 UserSpecCode

A user-specific value can be entered here to enable application-specific attributes in McuWIN.

5.6 [REFERENZFAHRT] – Reference travel

5.6.1 Order

Order of the referenciation. The axis number is filed in the respective decimal places (counting from 1 upwards). The axis with in the place with the lowest value is referenciated firstly (units digits).

Example:
Order = 123

Note: When commissioning it is useful if you firstly enter only the axis that shall be referenciated firstly, and realizes with this axis a successful reference travel (e.g. Z-axis = 3. axis)
Order = 3

Then you add the following axis **before** this number (e.g. X-axis = 1. axis).
Order = 13

Continue this until all axes are successfully referenciated.

5.6.2 ReferenzSchalter (reference switch)

Indication of the axes, which are referenciated on the reference switch (bit coded).

5.6.3 Indexsuche=0 (Index search=0)

Indication of the axes, which are referenciated on the index pulse (bit coded).

5.6.4 Direction

Reference – search direction (bit coded). 1 causes a referenciation to the negative direction.

5.6.5 GotoZeroAfterRef

With this variable it is indicated if the respective axis shall travel to its. The default value is 0.

5.6.6 ReferenzPosX / Y / ...

Reference position in the reference point.

5.6.7 RuhePosX / Y / ...

Rest position which can be moved to after referencing.

5.7 [FEHLERTEXTE] – Error texts

In this section, the display texts for errors are defined. A few errors are already predefined by McuWIN. You can copy this paragraph from the file “FehlerTexte.INI”. Further errors can be added according to the application. The error texts can be called in bit-coded form using the variable CI10.
From McuWIN V2.5.3.103, the variable CI48 and the assigned section [FEHLERTEXTE48] (error texts) is available as an error variable, especially for user-specific extensions

5.7.1 User specific error texts

Section [FEHLERTEXTE] (error texts) paragraph error number (decimal).
Here the text must be entered. This text must contain %X for the error number.

5.7.2 [FEHLERINFO] – Error info

Section variable number (decimal)
Here a CI-address must be entered that specifies one or two axes that caused the error. If two axes are to be indicated, the left axis must be entered *100 (e.g. 1213 – axis left in CI12 and axis right CI13)
For each axis that is specified, an error string is added automatically, in which the axis name is shown.
Example: Two axes are selected
[FEHLERTEXTE] – Error texts
1=Fehler # %i: Unknown function code!
2=Fehler # %i: Position error
[FEHLERINFO] – eror info
1=0
2=14

5.7.2.1 Error list

5.7.2.1.1 200 hex (512 dec) error in tool radius correction

This means that an error has been detected in the TC module of the universal object interface. Either a function could not be accessed or the module variable "ERROR" returned one or more error bits. Errors in the TC module are also displayed as hexadecimal error codes. The meanings of these errors are listed in Table 4 in the manual “TC Interface”. Multiple error bits may appear in combination.

5.8 [WARNTEXTE] – Warning texts

In this section warning and status texts are defined. These are bit coded and are be output when in CI15 the respecting bit is set. With the text output the corresponding bit in CI15 is removed.
If in addition the bit with the highest value is set in C15, the error window is deleted before the output. Warning texts are only output if no error (bit in CI10) is active.

5.8.1 User specific warning texts

Section [FEHLERTEXTE] (error texts) section error number (decimal)
Here a text must be entered. This text must contain %X for the warning / status number.

5.9 [TOOLCOMPENSATION] tool compensation

5.9.1 TcFileName

With this variable an Ini-File with information about the tool radius correction can be indicated. Sample data can be found in „ToolComp Demo.ini“.

5.10 [DDE]

With McuWIN you can communicate with a DDE master. Hereto “DDE communication” (“DDE Kommunikation”) must be observed. The settings for the DDE communication are realized in section DDE.

5.11 [SPINDEL]

In this section, the function of a spindle axis can be configured. Where a spindle axis has been configured, the processing of the main spindle as described in section 4.1 is disabled.
This functionality includes a spindle override, allowing the spindle speed to be adjusted manually. For thread cutting, the spindle and axis overrides are synchronised. If the spindle override is to be set other than with the override slider, the JOG override for the spindle axis (e.g. A.jovr) must be written to. A value of 1 for the jog override equals 100%.

5.11.1 SpindelAxis

Specifies an axis channel to act as the spindle axis. A value of 0, equivalent to entering OFF in the IniCfg program, deactivates a spindle axis.
This value is passed to RWMOS.ELF by C164 in bits 0..3.

5.11.2 SpindelType

This variable indicates the spindle type.

Value	Spindle type
0	Analog output
1	Controlled axis
2	User-programmed

This information is passed to RWMOS.ELF by C164 in bits 0..3, not as the numeric value given above but in bit-coded form.

5.11.3 InSpindelReady

This variable can be used to define the number of an input to the spindle axis to show that the spindle speed has been reached. A value of 0 deactivates the function. When the spindle is switched on, the program execution stops at the command in question (e.g. M03) until the spindle speed is reached.
This value is passed to RWMOS.ELF by C164 in bits 8..15.

5.11.4 SpindelOvrVisible

This variable can be used to indicate whether the override slider for the spindle override should be visible in the programming interface. A value of 1 causes the slider to be displayed. This value is passed to RWMOS.ELF by C164 in bit 16. The maximum and minimum values of the spindle override slider are adapted to the settings of the traverse override slider.

5.11.5 SpindelOvrEnabled

This variable is used to define whether the spindle override slider can be operated with the keyboard and mouse. This value is passed to RWMOS.ELF by CI64 in bit 17.
An alternative to operating the slider with the keyboard/mouse is e.g. to read an output value from a potentiometer. This query must be implemented by the user, e.g. in AppTask2.src. The override value is passed to the control via the JOG override for the spindle axis.
This value is passed to RWMOS.ELF by C164 in bit 17.

5.11.6 SpindelMaxVelocity

This variable is used to define the maximum speed of the spindle in the user unit (e.g. rpm).
This value is passed to RWMOS.ELF by C165.

5.11.7 SpindelSollIst

This variable is used to pass the details of whether target/actual values for the spindle axis should be displayed in the UI. If the target/actual display is activated, the symbolic axis name of the spindle axis is also displayed.

6 Variable assignment

For the meaning of the common variables, please read Chapter 10.2.

6.1 CD variable

No.	Meaning	Program
0-17	Parameter of G, T or other codes The used variables are coded by bits in the system variable AXSEL	McuWIN, TASK0
20	TRAC – Default track acceleration	McuWIN, TASK1
21	TRVL – Default track acceleration	McuWIN, TASK1
23	Buffered value of override at Value saved at advance lock	TASK0, TASK1
24	Velocity factor for free traversing	TASK0
30	Start position value A1 for position monitoring	TASK1
31	Start position value A2 for position monitoring	TASK1
32	Start position value A3 for position monitoring	TASK1
33	Start position value A4 for position monitoring	TASK1
34	Start position value A5 for position monitoring	TASK1
35	Start position value A6 for position monitoring	TASK1
36	Start position value A7 for position monitoring	TASK1
37	Start position value A8 for position monitoring	TASK1
CD40	Value max. velocity jump {mdvel} for axis A1	TASK1, McuWIN
CD41	Value max. velocity jump {mdvel} for axis A2	TASK1, McuWIN
CD42	Value max. velocity jump {mdvel} for axis A3	TASK1, McuWIN
CD43	Value max. velocity jump {mdvel} for axis A4	TASK1, McuWIN
CD44	Value max. velocity jump {mdvel} for axis A5	TASK1, McuWIN
CD45	Value max. velocity jump {mdvel} for axis A6	TASK1, McuWIN
CD46	Value max. velocity jump {mdvel} for axis A7	TASK1, McuWIN
CD47	Value max. velocity jump {mdvel} for axis A8	TASK1, McuWIN
CD48	JerkRel	TASK1, McuWIN
CD50	Return currently set zero shift axis A1	TASK0
CD51	Return currently set zero shift axis A2	TASK0
CD52	Return currently set zero shift axis A3	TASK0
CD53	Return currently set zero shift axis A4	TASK0
CD54	Return currently set zero shift axis A5	TASK0
CD55	Return currently set zero shift axis A6	TASK0
CD56	Return currently set zero shift axis A7	TASK0
CD57	Return currently set zero shift axis A8	TASK0
CD60	Reference position axis A1	TASK0, McuWIN
CD61	Reference position axis A2	TASK0, McuWIN
CD62	Reference position axis A3	TASK0, McuWIN
CD63	Reference position axis A4	TASK0, McuWIN
CD64	Reference position axis A5	TASK0, McuWIN
CD65	Reference position axis A6	TASK0, McuWIN
CD66	Reference position axis A7	TASK0, McuWIN
CD67	Reference position axis A8	TASK0, McuWIN
CD68	Control value for analog spindle output	TASK0, TASK1

No.	Meaning	Program
CD70	Rest position for axis A1	TASK0, McuWIN
CD71	Rest position for axis A2	TASK0, McuWIN
CD72	Rest position for axis A3	TASK0, McuWIN
CD73	Rest position for axis A4	TASK0, McuWIN
CD74	Rest position for axis A5	TASK0, McuWIN
CD75	Rest position for axis A6	TASK0, McuWIN
CD76	Rest position for axis A7	TASK0, McuWIN
CD77	Rest position for axis A8	TASK0, McuWIN
CD80	Traverse path for manual traverse (0 = infinite) in UU	Task1
90	Version of TASK0.SRC	TASK0
91	Version of TASK1.SRC	TASK1
92	Version of TASK2.SRC	TASK2
94	Version of McuWIN.EXE	McuWIN
95	Version of RWMOS.ELF	TASK1
600 to 699	Application specific	

6.2 CI variable

No.	Meaning	Program	
0	Command transfer for command interpreter G	McuWIN.EXE	
1	Command transfer for command interpreter M	McuWIN.EXE	
2	Command transfer for command interpreter Sonderbefehle (special commands)	McuWIN.EXE	
3	Command from Task -> McuWIN.EXE 1 = Delete error window and others	Task0 / McuWIN	
4	Parameter from Task -> McuWIN.EXE	Task0 / McuWIN	
9	Axes in the system are bit coded (reduced)	McuWIN.EXE T0	
10	Error variable This variable is bit coded. The individual errors are defined in GCODE.INC.	McuWIN.EXE T0	
11	Axes referenciated (bit coded)	McuWIN.EXE T0 T1	
12	Marker: Record protocol data	T0 T1	
13	Axes for EE-monitoring (bit coded in bit 0..15) and for counter monitoring by index (bit 16 .. 23)	McuWIN.EXE T1	
14	(WW: MPE causing axis) Marker: Do not start Task3 after command execution	(McuWIN.EXE) McuWIN.EXE T0	
15	Warning No. bit-coded MSB (bit 31) set means: Delete error window. This bit is deleted after it has been set by a task of McuWIN.exe!	MCUWIN.EXE T1	
16	Number of the actually available axes after boot of the control unit	McuWIN.EXE T1	

No.	Meaning	Program	
17	Axis No. (bit coded) at: Axis at error reference travel Axis at error MPE Axis at configuration error Axis at encoder error Axis at counter error (detected by index)	T0 T1 McuWIN.EXE	
18	Axis at ESL Hard + Soft (bit coded)	T1	
19	Axis at ESR Hard + Soft (bit coded)	T1	
20	Status of counter monitoring (bit coded)	T1	
21	Order of the reference travel (decimal coded)	McuWIN.EXE T0	
22	Referenciation on reference switch? (bit coded)	McuWIN.EXE T0	
23	Index search at reference travel? (bit coded)	McuWIN.EXE T0	
24	Direction of the reference travel (bit coded)	McuWIN.EXE T0	
25	Index of the X-axis (for ToolCompensation)	T1 McuWIN.EXE	
26	Index of the Y-axis (for ToolCompensation)	T1 McuWIN.EXE	
27	Index of the Z-axis (for ToolCompensation)	T1 McuWIN.EXE	
28	Realize a customer specific reference travel, if <> 0	T1 McuWIN.EXE	
29	Traverse zero point after reference travel? (bit coded)	T1 McuWIN.EXE	
30	Indicates bit coded options of RWMOS.ELF Bit 0 = Resource interface available Bit 1 = TC interface available Bit 2 = TC table values are programmed		
31	Status information for absolute stop or desired stop	McuWIN.EXE	x
32	TC-error word	TASK1.SRC	
33	Configuration word for MODEREG - S-profile - Look-Ahead		
34	Control generation MCUG2 = 2, MCUG3 (ADDIPOS) = 3	McuWIN.EXE All tasks according to requirements.	
35	LookAhead depth	McuWIN.EXE	
36	Marker if the system is dereferenced after emergency stop	McuWIN.EXE TASK1.SRC	
37	Information if the program is running: 0 – Program is not running 1 – Program is running 2 – Program is running in the single step mode	McuWIN.EXE	
38	additional error status information	T0, T1	
39	Bits to traverse axes manually	T2 McuWIN.EXE	
40ff	Oelmaier CNC-7A: Output variable for AWS		
48	Application-specific extension of error variable CI10 (bit-coded). For this, see also Chapter 3.13.4 and 10.2	application-specific, McuWIN.EXE	
49	Additional error information on warnings (bit code)	T1 McuWIN.EXE	
50ff	Oelmaier CNC-7A: Input variable for AWS		
50	At MBAWIN: Switch the ToolButton manually	MbaWIN.EXE	
58	free – no longer assigned as of 29/01/2015		
59	AliveCounter McuWIN.exe -> RWMOS / SAP	McuWIN.EXE, Task1.src	
60	Synchronisation variable for program start 1 = Task 1 must wait -1 = Task 1 is in continuous monitoring	McuWIN.EXE, TASK1.SRC	
61	Reference switch is latch input (coded by bits)	McuWIN.EXE, TASK0.SRC	

No.	Meaning	Program	
62	No referenciation on limit switch (coded by bits)	McuWIN.EXE, TASK0.SRC	
63	Register for program control (coded by bits)	McuWIN.EXE	
64	Encryption of spindle data by bits	McuWIN.EXE, TASK0.SRC	
65	Spindle maximum speed in user units	McuWIN.EXE, TASK0.SRC	
66	Override control for thread cutting Bit 0 = traverse override slider blocked Bit 1 = spindle override slider controls spindle and axes	TASK0.SRC writes McuWIN.EXE reads	
67	BoardType (of InitMcuSystem3)	McuWIN.EXE	
69	In bit form – axes in open-loop operation	McuWIN.EXE, TASK1.SRC	
70	Unknown G command (for runtime error 1)	TASK0	
71	Unknown M command (for runtime error 1)	TASK0	
72	Unknown command (for runtime error 1)	TASK0	
73	Unknown command in CI3 (for runtime error 1)	McuWIN.EXE	
90	Specify version error: 10 / 20 = Version of Task 0 is too small for Task 1 11 / 21 = Version of Task 0 is too small for Task 1 12 / 22 = Version of Task 0 is too small for Task 1 14 / 24 = Version of RWMOS is too small for Task 1 15 / 25 = Version of McuWIN is too small for Task 1		
100 ff	DDE writing the status variable	McuWIN.EXE	
200 ff	DDE reading the status variable	McuWIN.EXE	
300	Status information on manual joystick traversing in bit-coded form Bit 0..7: Active axes coded in bit form Bit31: Traversing enabled (TeachIn mask is viewable)	TASK1.SRC	
301	indicates active axis channel, is necessary for safe axis stop		
302	indicates if in SingleStep mode, it is possible to traverse manually (from V2.5.3.122)	McuWIN.EXE, Task1.src	
310 .. 317	Zero offset of analog input for analog joystick Units digit of the index = Index of the axis and index of the analog channel	TASK1.SRC	
320 .. 327	Maximum deflection of the analog channel, negative sign causes inversion of the direction Units digit of the index = Index of the axis and index of the analog channel	TASK1.SRC	
390 .. 397	Debugging display AnalogIn can be used for setup	TASK1.SRC	
500	At MBAWIN: Buttons Visible	MbaWIN.EXE	
501	At MBAWIN: Buttons Enable	MbaWIN.EXE	
600 to 699	Application specific		

7 Description of the operating modes

7.1 EditMode

This is the operation mode for editing programs. Program start is possible if the axes are referenciated (if this characteristic is not disabled).
In the EditMode the axes can be traversed manually by the TeachIn-window. The manual procedure is still possible with the CI-variable CI39.

Table: Bit coding in CI39:

Bit	Axis	Direction
0	1	positive
1	1	negative
2	2	positive
3	2	negative
4	3	positive
5	3	negative
	etc.	

The description of CI39, depending on digital inputs, can be realized e.g. in Task2 of a closed loop.

7.2 RunMode

A program is realized in the automatic mode. Referenciation and new program start is not possible. The program also cannot be edited.
With “Stop” you can stop, with “Step” („Schritt“) you can switch to the StepMode and with “Reset” you can reset the control.
The RunMode is finished automatically when the UserTask and function task are fixed.
With the commands M00 and M01 you can switch into the Step-Mode. M01 is only working when the button “Desired stop” („Wahlweiser Halt) is activated.

7.3 StepMode

SingleStep mode: This terminates automatically when no new rows are reached after a step. Manual traversing of the axes (via CI39) is possible if the flag “Manual Move in SS” is set on the tab “Desktop / System” in IniCFG.exe (Ini File Section [TEACHIN] – „ManualMoveInSS“ → CI302 <> 0).

7.4 HaltMode

Transition status between RunMode and EditMode.

7.5 ManMode

Manual procedure is active (Teach-In, reference travel). In this mode no automatic cycles may be started.

7.6 Reference travel

The sequence of the reference travel can be configured with the corresponding variables in McuWIN.INI. If the reference travel cannot be configured with the predetermined variation possibilities, in the file „KdRefFahrt.INC“ an application specific sequence can be programmed. In order to activate this, in the variable CI28 the value 1 must be entered (e.g. in the file „AppCommands.INC“).
The axes are always referenciated sequentially in the order that is indicated in the variable „sequence“.
Firstly, the set values for Jog-velocity, Jog-acceleration, Home-velocity and Home-acceleration are checked. If here a error is detected (value <= 0) the error „conflict in the configuration data!“ („Konflikt in den Konfigurationsdaten!“) (20H) will be displayed.
If within the sequence an axis is indicated several times or if an axis is indicated, which is not available in the system, this error also is displayed. Then the axis will be travelled free by the limit switch, if necessary. Now the axis is started into the search direction, which is indicated in the variable “direction” (“Richtung”) in the bit corresponding with the axis. 1 means search in negative direction (with smaller position values).

7.6.1 Referenciation on reference switch

This variant is selected axis specific, if the bit corresponding with the axis is set in the variable „reference switch“. Then the axis is traversed in the selected search direction to the reference switch. As reference switch input the “fast latch input” of the corresponding axis **must** be used (1. axis = I14, 2. axis= I15, 3. axis = I16).

7.6.2 Referenciation on limit switch

This variant is selected axis specific if the bit corresponding with the axis is not set in the variable „reference switch“. Then the axis is travelled on the hardware limit switch and is stopped by this and then released again.

7.6.3 Fine positioning on zero track

By setting the axis specific bit in the variable „index search“ („Indexsuche“) now a fine positioning to the zero track of the incremental encoder is realized.

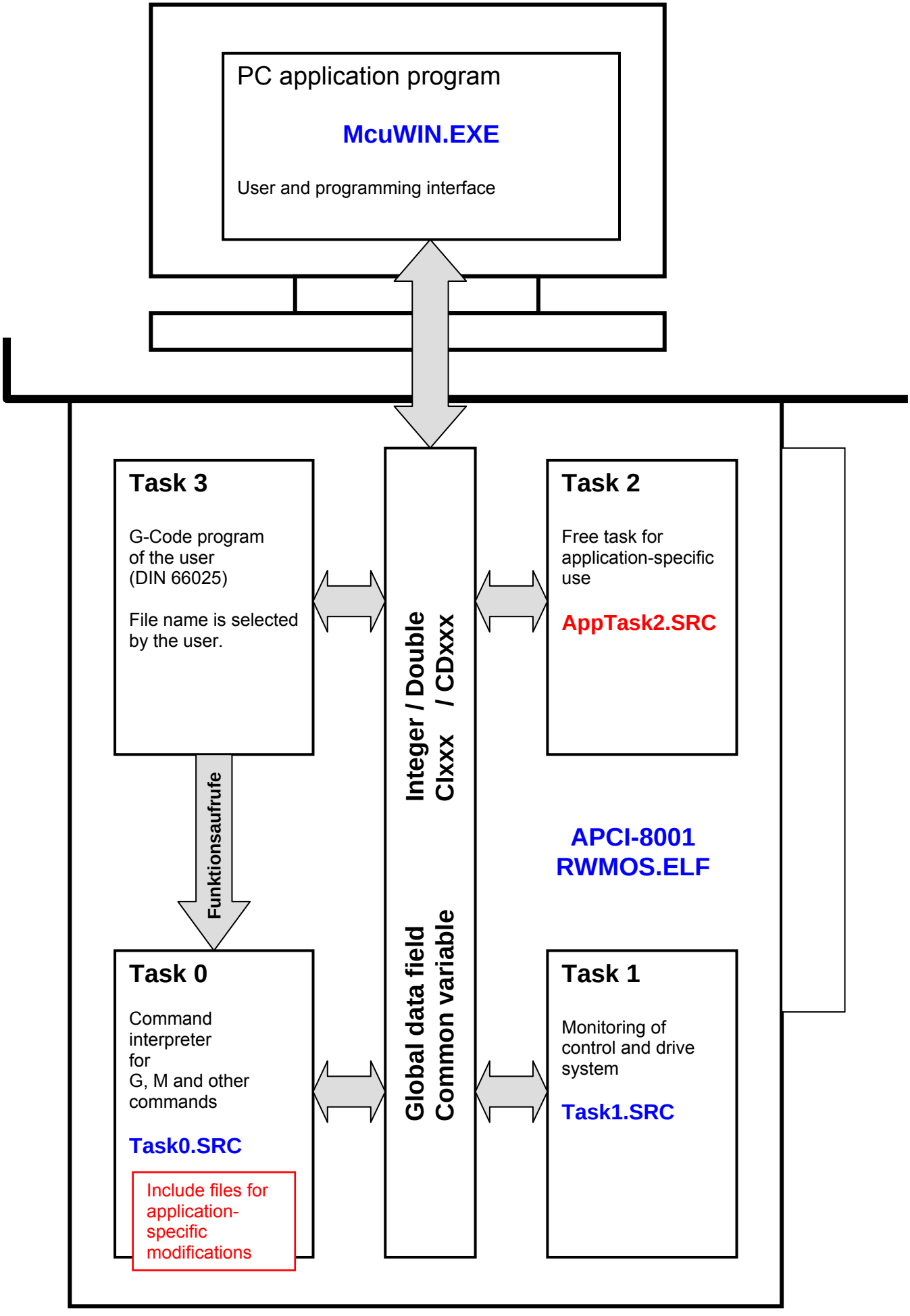
7.6.4 Setting the home position

In the position that is found now, the home position is set on a value, which is indicated in the variable „ReferenzPos?“ for the corresponding axis. After all axes have been referenciated, the hardware and software limit switch monitoring is activated.

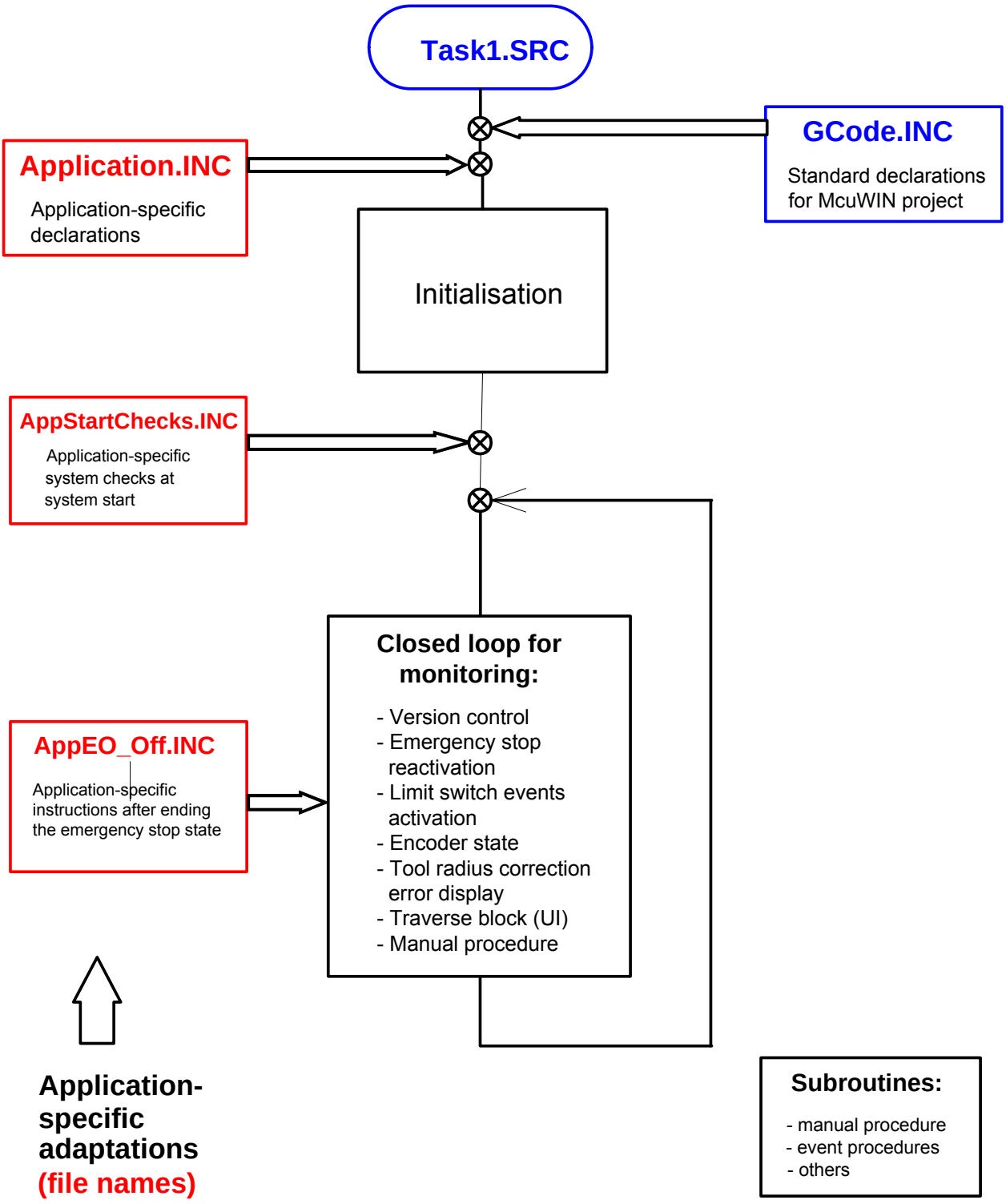
8 Description of the SAP task environment

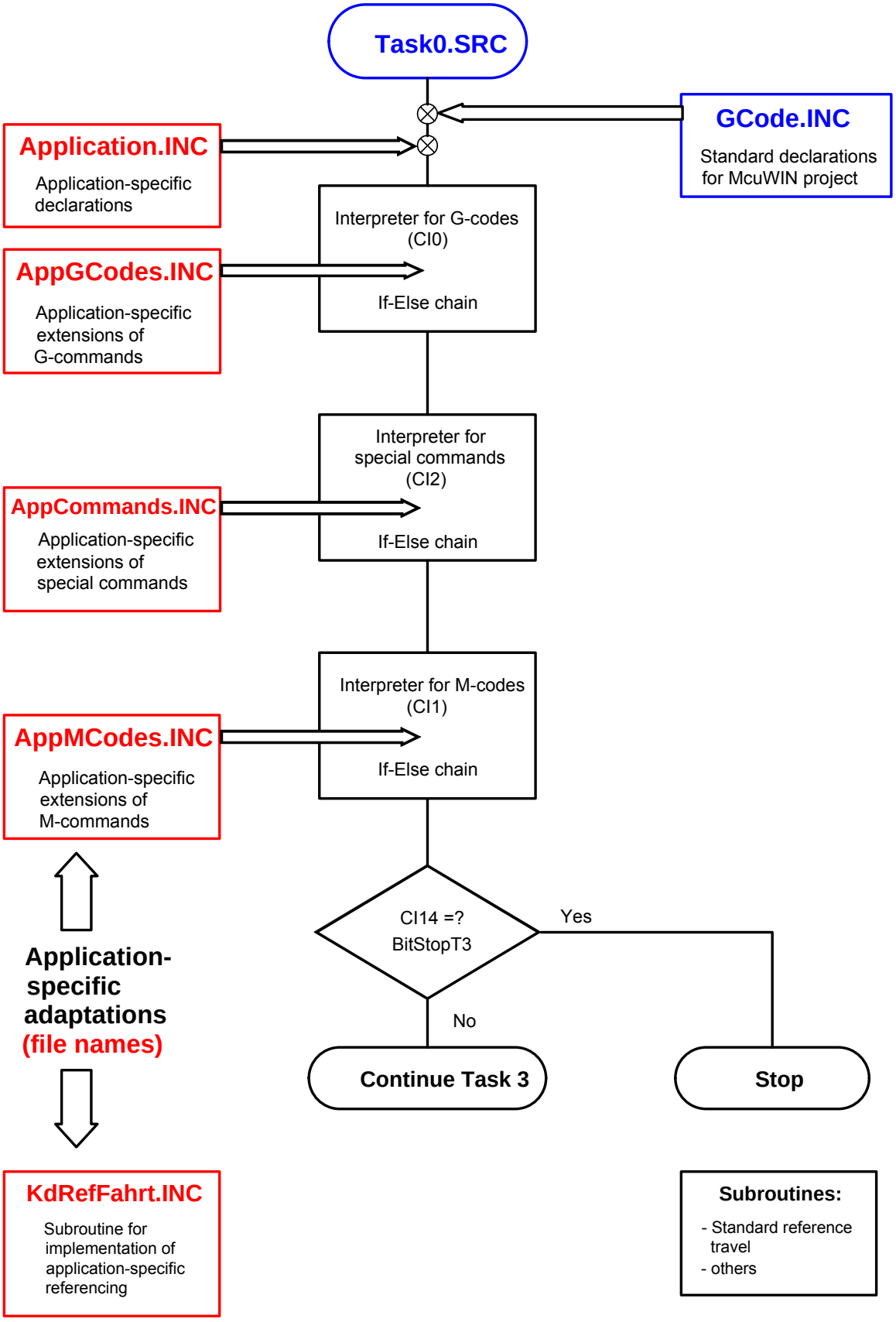
8.1 Background information on the APCI-8001 / APCI-8008

G-Code programs according to DIN 66025 can be processed with the APCI-800x in an SAP task. For this, the programs are compiled into a CNC file, loaded onto the control unit and started. Here, the available range of commands must be considered, which can be extended according to the application.
As described in the manual, the APCI-800x provides four tasks for processing NC programs. G-Code programs are always executed in Task 3. McuWIN uses these tasks as shown in the following diagram.
In case of changes in the source text files or in the Include files of the SAP tasks, the respective source text files must be recompiled and then become effective on the control unit by a restart of McuWIN. For compiling, the mcfg program in a text editor or the command line program ncc.exe is used. It is not possible to directly compile in McuWIN because this is configured for processing G-Code programs. The background task programs, however, are programmed in rw_SymPas.


PC application program
McuWIN.EXE
User and programming interface
Task 3
G-Code program of the user (DIN 66025)
File name is selected by the user.
Task 2
Free task for application-specific use
AppTask2.SRC
Integer / Double
CIxxx / CDxxx
Global data field
Common variable
Funktionsaufrufe
APCI-8001
RWMOS.ELF
Task 0
Command interpreter for G, M and other commands
Task0.SRC
Include files for application-specific modifications
Task 1
Monitoring of control and drive system
Task1.SRC

In Task 0 an application-specific command interpreter that is called from Task 3, if necessary, is loaded by McuWIN. In Task 1 still an initialisation and monitoring program is started, which executes the allocation of default values, e.g. track acceleration, start values for CI- and CD-variable and the monitoring of the system status, e.g. handling of the limit switch, emergency stop handling, etc. A sample program for the G-Code programming is available under the name BEISPIEL.SRC. The processes for Task 1 and Task 0 are shown below in flowchart form.




Task0.SRC
GCode.INC
Standard declarations for McuWIN project
Application.INC
Application-specific declarations
Interpreter for G-codes (CI0)
If-Else chain
AppGCodes.INC
Application-specific extensions of G-commands
Interpreter for special commands (CI2)
If-Else chain
AppCommands.INC
Application-specific extensions of special commands
Interpreter for M-codes (CI1)
If-Else chain
AppMCodes.INC
Application-specific extensions of M-commands
Application-specific adaptations (file names)
CI14 =? BitStopT3
Yes
No
Continue Task 3
Stop
KdRefFahrt.INC
Subroutine for implementation of application-specific referencing
Subroutines:
- Standard reference travel
- others

The blocks shown in red on the left-hand side of the flowcharts are the application-specific extensions. These are not replaced in case of updates.

8.1.1 Compiler mode for G-Code programs

The McuWIN program automatically compiles the programs in the editor window at start-up as source text files in DIN66025 modes (where McuWIN is configured for this). This operating mode can also be selected for the compilers of other add-in programs (e.g. for offline compilation).

8.1.1.1 Command lines Compiler NCC.EXE

The command line program NCC.EXE must be called with the parameter DIN.

Example:
NCC PrgName DIN

8.1.1.2 MCFG for MS-DOS

The program MCFG.EXE for DOS must be converted once to the DIN-mode. This is to be done as follows:

Start of the program with the parameter PLC
Select in the menu <<Editor>> <<Setup>> <<Set Compiler Mode>>
the option DIN-code by mousecklick.
(Switch off by selecting rw_SymPas)

This information will be stored in the file MCFG.DAT!

8.1.1.3 MCFG for 32bit-Windows

If an editor window is opened and activated, in the menu <<Compile>> <<Options>>, the option <<Select DIN 66025 Language>> can be selected. This information will not be stored and must be activated after each new call. After this action in the menu <<Trace>> the additional menu option <<Trace DIN66025 Task>> is available, with which corresponding programs can be compiled, loaded into Task 3 and started.

8.1.2 Program status information for the control

From V2.5.3.50, RWMOS.ELF provides various status information during the execution of a G-code program. This information can be read and displayed on a cyclical basis. To read this information, you need mcug3.dll, version 2.5.3.34 or above, and an up-to-date high-level language interface. The relevant commands are described below. This description is not included in the PHB manual, as these commands can only be used in conjunction with the G-code interface. It is only important to know this information if a separate programming interface or status display is to be programmed.

8.1.2.1 Information on the current traverse profile

The “Profile Information Register” {pir} is used to display the following information in bit-coded form:

Table: Bit-coding of pir

Bit / hex	Name	Description
0 / 0001	***Linear***	Linear or circular interpolation performed.
1 / 0002	***Circular***	Circular interpolation performed. Bit 0 is normally also set with this bit, as G02 and G03 traverse commands are always executed as helical interpolation.
2 / 0004	***Spline***	Spline interpolation performed
3 / 0008	***Jog***	G00 executed. This bit is also set for movements during referencing.
4 / 0010	***Absolut***	Set for absolute positioning; if 0, then relative positioning
5 / 0020	***ClockWise***	G02 command executed. The ‘Circular’ is always set with this bit. If the ‘Circular’ bit is set and this bit is 0, a G03 command will be executed.
6 / 0040	***Spooled***	This bit shows that spooled traverse commands are being executed (G01, G02, G03).
others	***undef.***	These bits are undefined / free for future enhancements.

The “Profile Information Register” is read with the dll function rdpir:

DESCRIPTION:	This command can be used to read the “Profile Information Register” (pir). The bit-coding of this register can be seen in the table above.
BORLAND DELPHI:	function rdpir (var value: integer) : integer;
C:	int rdpir (int *value);
VISUAL BASIC:	Function rdpir (value As Long) As Long
Return value:	< 0: Command not supported by RWMOS.ELF = 0: Commando executed successfully > 0: Time-out; command not executed
NOTE:	Under rw_SymPas, pir can be accessed with the system variable *pir*.

8.1.2.2 Information on the axis-specific target position

DESCRIPTION:	This command can be used to read the axis-specific target position of the current profile (ptp) into the value variable as an absolute value in the axis-specific unit. The parameter *an* gives the axis index of the axis channel to be read (0..n). The target position of the sub-profile currently being run is also determined within an interpolation contour. In contrast, the function rdtp() only returns the target position of the whole contour.
BORLAND DELPHI:	function rdptp (an: integer; var value: double) : integer;
C:	int rdptp (integer an, double *value);
VISUAL BASIC:	Function rdptp (ByVal an As Long, value As Double) As Long
Return value:	< 0: Command not supported by RWMOS.ELF = 0: Commando executed successfully > 0: Time-out; command not executed
NOTE:	Under rw_SymPas, ptp can be accessed with the axis qualifier ptp.

8.1.2.3 Information on the programmed traverse speed

DESCRIPTION:	This command can be used to read the programmed and actual track speed of the current interpolation profile (dtv) into the Array value as an absolute value in the programmed interpolation unit. For G00, a value of 0 is returned
BORLAND DELPHI:	function rddtv (var value: array of double) : integer;
C:	int rddtv (double value[2]);
VISUAL BASIC:	Function rddtv (ByRef value As Double) As Long
Return value:	< 0: Command not supported by RWMOS.ELF = 0: Commando executed successfully > 0: Time-out; command not executed
NOTE:	Under rw_SymPas, dtv[] can be accessed with the system variables *dtv0* and *dtv1*. **Warning:** The function must be passed an array with at least two elements, or an undefined area of memory will be written to. The first element returns the track speed programmed by the user. The second element returns any limited track speed.

8.2 Supplementing of customer specific codes

The list of G- and M-commands can be extended by the user on an application-specific basis. The NUMPARAM, AXSEL and common-double variables are provided for evaluating parameters. The following parameter types are possible:

List with numerical values: The numerical values are transferred to the task environment in CD0, CD1, etc. The number of these parameters can be found in the NUMPARAM variable. The AXSEL variable is zero.

Example: M24 20.6 7.3

AXSEL = 0
NUMPARAM = 2
CD0 = 20.6
CD1 = 7.3

List with axis names: This list does not contain any numerical values. In the AXSEL variable, the specified axes are shown in bit-coded form and with their axis indices. NUMPARAM is the number of specified axes. In the CD variables with the index of the relevant axis, the index is entered in the order specified (starting with 0). This means that the order of an axis list in a G- or M-command can be evaluated.

Example: M24 X Z

The 1st axis would be the X axis, the 3rd axis would be the Z axis.
AXSEL = 5
NUMPARAM = 2
CD0 = 0.0
CD1 is undefined
CD2 = 1.0

List with axis names to which a numerical value is assigned: In the AXSEL variable, the specified axes are shown in bit-coded form and with their axis indices. NUMPARAM is zero. The numerical values are transferred to the task environment in CD0, CD1, etc., i.e. in the CD variables whose index corresponds to the axis index. Using this parameter transfer, axes can be selected and given a position or velocity index.

Example: M24 X20.6 Z7.3

The 1st axis would be the X axis, the 3rd axis would be the Z axis.
AXSEL = 5
NUMPARAM = 0
CD0 = 20.6
CD2 = 7.3

8.2.1 G-codes

The compiler processes only several G-codes directly. G-codes that are not processed directly by the compiler, cause a calling of Task 0 with the transfer of parameters in CI-variables and, if necessary, in CD-variables. In Task 0, the corresponding functionality of the command can be realized, if this is not already provided by the control.
The G-codes listed below are processed directly by the operating system software RWMOS.ELF and do not cause the call of Task 0:

G00, G01, G02, G03, G04, G53 – G60, G7, G71, G90 – G94, G150, G151.

All other G-commands cause a calling of Task 0. Its functionality must be tested there. The commands

G17-G19, G21 – G24, G39 – G42, G50, G51, G74, G98, G99, G161 and G162

are programmed in Task0.SRC and can, if required, be reprogrammed user specifically. A few G-codes cause the execution of a movement contour with the support of spooler commands. These are the commands

G01, G02, G03, G17, G18, G19, G40, G41, G42, G90, G91, G200 - G299.

With all other G-commands a contour is finished and executed. When the user programs out these G-codes, it must be observed, that these commands are in fact assigned with spooler commands, especially at the functions that are still free

G200 to G299.

During the execution of an unknown G-code, Task 0 is called with the G-code number in CI0. An allocation to X, Y or Z is transferred in CD0, CD1 or CD2 (index of the CD variable = index of the indicated axis, see above). The axes, which were indicated at the calling of the command, are shown bit coded in the system variable AXSEL. So the parameter list can be checked.

8.2.2 M-codes

Also a few M-codes are processed directly by RWMOS.ELF. These are the commands

M17, M26, M27, M80, M96, M97, M98.

All other m-commands cause the calling of Task 0. Its functionality must be tested there. The commands

M00 - M06, M08, M09, M30

are programmed in Task0.SRC and can, if required, be reprogrammed user specifically.

The execution of a movement contour is not interrupted with the commands

M00, M01, M26, M27 und M200 – M299

When the user programs out these m-codes, it must be observed that these commands are in fact assigned with spooler commands, especially at the functions that are still free.

M200 to M299.

During the execution of an unknown M-code, Task 0 is called with the M-code number in CI1. This commando can now be called with various parameter types (see above):

Call with axis references, as with G-codes:

Example (axis X = 1st axis, Z = 3rd axis):
M37 X20 Z40

In this case, the system variable AXSEL shows the selected axes in bit-coded form (AXSEL = 5). CD0 contains the value 20, and CD2 the value 40.

It is also possible to specify axis qualifiers without numbers:

M37 X Z

In this case, in the relevant CD variables, the index in the parameter specifications is transferred starting with 0.

Call with double parameters and no axis reference:

Example:
M27 20.5

In this case, the system variable NUMPARAM shows the number of parameters in bit-coded form (NUMPARAM = 1). CD0 contains the value 20.5. Any other parameters would be contained in the following CD variables. The data type of the variable is always Double (floating point). Note that formulae can also be passed as parameters. Negative values must then be placed in brackets.

Example:
M27 10 –5

This call is interpreted as a parameter with the value (10-5). Where two parameters are to be passed, the call must look like this:

M27 10 -5

A mix of parameter types (axis qualifiers and numbers) is not allowed.

8.2.3 State information

For the implementation of specific G- or M-codes, it is often necessary to identify different system states. Access to some important system variables is described below:

8.2.3.1 Interpolation level

The tool radius and tool length corrections are always based on a right-handed Cartesian coordinate system. Here, it is assumed that the machine axis X is assigned to the axis index 0 (1st control axis), the machine axis Y to the axis index 1 (2nd control axis) and the machine axis Z to the axis index 2 (3rd control axis). If another assignment is to be used here, in the file GCode.inc, the constants NrXAxis, NrYAxis and NrZAxis need to be adapted, and after that, the source text file Task1.src has to be compiled.

The interpolation level set with G17/18/19 can be read in the object variable ***FTcPlane_r***. This contains the value 17, 18 or 19 and has a meaning in accordance with the relevant G-command.
The index of the X-axis can be taken from the object variable ***FTcXAxNr_r***.
The index of the Y axis can be taken from the object variable ***FTcYAxNr_r***.
The index of the Z axis can be taken from the object variable ***FTcZAxNr_r***.

8.2.3.2 Interpolation axes

The axes which are currently defined as interpolation axes are mapped in the least significant 16 bits of the IPolMode system variable (see also Section 10.2.1).

8.2.4 Application-specific use of the default outputs

In the file “GCode.inc”, some outputs of the circuit board are provided for default use. These outputs are also given in the description of McuWIN. If they are to be used in an application-specific way, all functions using these outputs must be patched. In the following table, the outputs, the functions and the corresponding source text files are listed.

Table: Default outputs and note for patch

Signal name	No.	Used in function	.. File	Patch in File
`DefaultKuehlung` (cooling)	4	M08 / M09	Task0.SRC	AppMCodes.inc
`DefaultRichtungHS` (main spindle direction)	6	M03 / M04 SpindelStopp - M00 - M05 - M30 - 1000	Task0.SRC	AppMCodes.inc Application.inc AppMCodes.inc AppMCodes.inc AppMCodes.inc AppCommands.inc
`DefaultHauptspindel` (main spindle)	7	M03 / M04 SpindelStopp - M00 - M05 - M30	Task0.SRC	AppMCodes.inc Application.inc AppMCodes.inc AppMCodes.inc AppMCodes.inc
`DefaultProgrammAktiv` (program active)	8	G74 1000 1001 1002	Task0.SRC	AppGCodes.inc AppCommands.inc AppCommands.inc AppCommands.inc

8.2.5 Special functions

During the execution of an unknown special command, Task 0 is called with the command number in CI2. A parameter is transferred in CD0.

8.2.6 Further calling conventions

After calling Task 0 e.g. with a self-defined G or M command, the execution of Task 3 is stopped. Task 0 must now process the command displayed. When execution is completed, the command must be acknowledged before continuing Task 3 with the instruction

contcnct (3);

The SRC-sample program still contains the following convention:

- If after the execution of the command Task 3 shall be started again, in CI14 the bit 1 must be reset.

- If after the execution of the command Task 3 shall not be started again, in CI14 the value 1 is entered. This value is always reset again after the execution of the command.

8.2.7 Errors and warnings

If errors occur in the program flow, a bit can be set in the error variable CI10 or in CI48. The use of these bits must be accurately documented in order to prevent double assignment. Some bits in CI10 are already assigned by the system. These are documented or declared in GCode.inc. To avoid any conflict with user-specifically assigned bits after future updates, the user should only access the error variable CI48. If a bit is set in these variables, a clear text message will be displayed in the error window of the interface. The content of the message can be defined in McuWIN.INI, in the sections [FEHLERTEXTE] (error texts for CI10) and [FEHLERTEXTE48] (error texts for CI48). In case of changes or a creation of new lines, the respective syntax requirements must be met here. It is best to comply with other lines in the corresponding section (for this, see also Chapter 5.7). If the respective error is to provoke a breakpoint, the corresponding action has to be taken. A normal breakpoint can be achieved within Task 0 (and the corresponding Include files) with the following sequence, for example:

```
CI0 := 0;
CI1 := 0;
CI2 := 1000;      // Call stop command in Task 0
goto RestartT0;
```

Warnings or status messages can be generated by setting a bit in CI15. The message texts for this are defined in the section [WARNTEXTE] (warning texts). An active bit in CI15 is detected, displayed and deleted by the interface. The most significant bit in CI15 has a special function: If this bit is set, the error window will be deleted. Thus, it is possible to delete displays that have been applied before. In this case, all available error messages are deleted in the error window, too. However, they are immediately restored if the corresponding error bits in the error variables are still available.

Note: To apply a warning display securely, a raw Clear request must not be available in CI15. The subroutine

```
WaitForNoClearCI15;
```

is used to wait until this bit has the value 0. This ensures that the desired message is really displayed.

8.3 Task 0 – Command interpreter

In Task 0, usually a task is executed that provides the functionality of G-codes, M-codes and of special functions, which are not directly handled by the compiler. For this, a program is available under the name TASK0.SRC, which can be extended according to the application.

8.3.1 Implementation of customer specific G-codes

When calling a G-code command, which is not directly processed from the control, a function call in Task 0 is executed. Here the number of the G-code is entered in CI0. Values that are indicated with the axis names, are entered in the common-double-variable 0.1,… whereas the index of the CD-variable corresponds with the axis index. The axes, which are entered in the calling command, are shown bit coded in the system variable AXSEL. After this the G-code task (Task 3) is stopped and Task 0 is started. So the amendment of the interpreter in Task 0 allows the amending of own G-codes. The user usually receives an adequately adapted version of the file TASK0.SRC according to the requirement of amendment. However, the user also can realize own amendments. Hereto please observe the following chapter.

Note: After reinstalling McuWIN, TASK0.SRC must be compiled newly if one of the following Include files contains customer specific amendments.

8.3.2 Include-Files in TASK0

8.3.2.1 GCode.INC

This file contains the project-wide installation-specific constants and variable declarations.
This file is replaced if McuWIN is reinstalled (new installation without previous uninstallation). Thus, any user-specific amendments at this point will be lost when updates are performed. This file should therefore not be modified by the user. The Application.INC file is provided for this.

8.3.2.2 Application.INC

In this file user specific constants and variable declarations can be entered.
This file is kept at a reinstallation of McuWIN (new installation without previous uninstallation). Thus, user specific amendments are also available after updates at this place.

8.3.2.3 KdRefFahrt.INC

The sequence of the reference travel is programmed in TASK0.SRC in the procedure “reference travel” („Referenzfahrt“). If this sequence shall be adapted customer specific, this sequence shall be programmed in the Includefile „KdRefFahrt.INC“. Then this function must be activated by entering a value in CI28 <> 0.
This option can be set in the installation program IniCFG.EXE. Before McuWIN V2.5.3.24 this entry had to be done in AppTask2.SRC.
This file is kept at a reinstallation of McuWIN (new installation without previous uninstallation). So user specific amendments are also available after updates at this place.

8.3.2.4 AppGCodes.INC

In this file application specific G-codes are programmed out.

This file is kept at a reinstallation of McuWIN (new installation without previous uninstallation). So user specific amendments are also available after updates at this place.

8.3.2.5 AppCommands.INC

In this file application specific special commands, e.g. S-command, are programmed out. This file remains preserved at a reinstallation of McuWIN (new installation without previous uninstallation). So user specific amendments are also available after updates at this place.

8.3.2.6 AppMCodes.INC

In this file application specific m-codes are programmed out. This file is kept at a reinstallation of McuWIN (new installation without previous uninstallation). So user specific amendments are also available after updates at this place.

8.3.2.7 Application.INC

In this file user specific constants and variable declarations can be entered.
This file is kept at a reinstallation of McuWIN (new installation without previous uninstallation). Thus, user specific amendments are also available after updates at this place.

8.4 Task 1 – Initialisation and monitoring task

Task 1 is available for the execution of an initialisation and monitoring task. Hereto a sample program under the name TASK1.SRC is available. This task can be amended application specifically. Firstly, this task processes an initialisation list and then remains in a closed loop, in which cyclic monitoring functions are processed.
The user usually receives an adequately adapted version of the file TASK1.SRC, if wanted, according to the required amendments.

8.4.1 Initialisations in module TASK1.SRC

The following descriptions are only important for the system adaptation. The user of the program will not get in contact with these system files!

8.4.1.1 Unit for interpolation commands

Hereto a time or position unit is allocated to the system variables TU or PU (see „Programming Manual“ chapter 6.2.1.2)

8.4.1.2 Acceleration for inpterpolation commands

In order to set the acceleration value for interpolation commands, a corresponding value is allocated to the system variable (see „Programming Manual“ chapter 6.3.1). The unit of this value is defined in the system variables PU and TU.

8.4.1.3 CI- and CD-variable

These are application specific variables, which can contain system variables, status and error variables. These variables can be read directly from a PC interface and evaluated. The following variables are used from the system, e.g. for parameter transfer at code-calls:

Table: Common variables that are used for the parameter transfer:

Variable	Use
CI0	G-code No.
CI1	M-code No.
CI2	Special command No.
CI10	Error variable
CI11	Referenciated axes (bit coded)
CD0	Parameter for several commands
CD1	Parameter for several commands
CD2	Parameter for several commands
CD3..7	Parameter for several commands

8.4.2 Monitorings

The monitoring functions are executed in Task 1, which is always active. Individual monitorings are, if required, activated or deactivated by system statuses, (see e.g. software limit switch).
A few system statuses are monitored by the RWMOS-operating software and generate a corresponding event-routine, when occurring. These routines must be activated before the use (e.g. EVMPE). Hereto please see the descriptions in the „Programming Manual“ (chapter 6.4).

Caution: The task, in which events are managed, may not contain long Wait commands, as during a Wait, e.g. wt (2000) the corresponding task is inactive.

8.4.2.1 Position error

The axis specific max. position error is defined in the program mcfg. Exceeding this position error can be treated in the event procedure EVMPE.

8.4.2.2 Hardware limit switch

A hardware limit switch can be treated in the event procedure EVLSH.

8.4.2.3 Software limit switch

A software limit switch event can be treated in the event procedure EVLSS. This event firstly will be monitored after the command SHP at the respective axis, as at not referenciated axes the monitoring of a software limit switch normally is not useful.

8.4.2.4 Emergency stop

If an input is active, to which the function EO was allocated, will be branched out into the event processing routine EVEO.
This routine may process only indirectly an emergency stop event. Please observe the corresponding safety regulations for the direct effects of the emergency stop to the drives.

8.4.2.5 Gain ready

The event routine EVDNR is executed if an input is inactive, which was declared as DR.

8.4.2.6 Encoder-Error-Flag

This flag can be monitored cyclically for an increased system safety.

8.4.2.7 Encoder verification

With the index-latch-function the counter value of the encoder logic can be monitored cyclically.

8.4.2.8 Further monitoring functions

In this chapter further monitoring functions could be added. For this, please refer to the documentation of the APCI-8001.

8.4.3 Include files in TASK1

8.4.3.1 GCode.INC

See Section 8.3.2.1

8.4.3.2 Application.INC

See Section 8.3.2.2

8.4.3.3 AppStartChecks.INC

This module is integrated in the initialisation phase of Task1.SRC and provides the option for one-time initialisations and system checks when the system is started or after a system reset.
This can, for example, be checks of I/O requirements such as air or oil pressure.
This file is retained during a reinstallation (new installation without previous uninstallation) of McuWIN. Thus, user-specific amendments at this point remain available, even after updates.

8.4.3.4 AppEO_Off.INC

In this file, application-specific instructions, e.g. the setting of an enabler output, can be added. These instructions are to be executed after the emergency stop state is cleared.
This file is retained during a reinstallation (new installation without previous uninstallation) of McuWIN. Thus, user-specific amendments at this point remain available, even after updates.

9 Spindle error and angle error compensation

Spindle errors and angle errors in a Cartesian axis arrangement can be easily compensated in McuWIN. To do this, simple text files containing the compensation tables must be added to the McuWIN directory. The names of these files are specified by McuWIN. At the start, McuWIN checks to see if these files exist in the McuWIN directory and, where necessary, initialises the appropriate compensation. The optionELCAM option in RWMOS.ELF is required for the system to be able to process compensation tables. The names of these files have the following structure:

CompTable_X_Y.txt

Here, Y is the symbolic axis name of the axis to be corrected; X is the symbolic name of the axis whose traversing range is used to determine the compensation. X and Y can also be the same, in which case it is a spindle error which is corrected. The symbolic axis name is the name assigned to an axis with the system data in mcfg.exe.

A compensation table contains a header and the actual table with the axis errors to be compensated. This table contains an axis position value for each line and the axis error at the point of the position value. The unit of these values is the position unit set on an axis-specific basis, e.g. mm.

Example of a line in a compensation table:

100.0 -0.015

A compensation table must comprise at least two calibration points (lines), but can also have several hundred calibration points. Here, the axis position value must always be specified in ascending order. The compensation values between these calibration points are interpolated linearly. The minimum and the maximum axis position value should be outside the traversing range of the relevant axis.

If one or more such tables are found, McuWIN generates the log file “SpindleComp_Protocol.txt” at the start in which comments on programming the table are written. If an error is detected during configuration of the error compensation, this is shown by an error window. In this case, the relevant file must first be corrected before using McuWIN.

9.1.1 Spindle error compensation

A table with the following layout must exist in order to compensate for a spindle error:

File name:	CompTable_X_X.txt
Content of first line:	Number of calibration points
Content of following lines:	Position value Compensation value

Example: The spindle error on the Y axis is to be compensated.
Traversing range of axis from 0 .. 500 mm
Error at 0: 0 mm
Error at 200 mm: +0.1 mm
Error at 500 mm: -0.05 mm

File name: CompTable_Y_Y.txt

File content:
5
-1000.0 0.0
0.0 0.0
200.0 0.1
500.0 -0.05
1000.0 -0.05

9.1.2 Angle error compensation

With angle error compensation, two different cases are generally possible:
1.) The error to be compensated on one axis depends on a second axis (master axis).
2.) The error to be compensated on one axis depends on the position of two other axes.
There is, therefore, a first and a second master axis.

In the first case, the compensation takes place analog to the compensation of a spindle error. The compensation file has the same structure, but the master axis is not identical to the axis to be compensated.

In the second case, the compensation file is somewhat more extensive. Only this case is dealt with below. A multi-line table is defined for this. In practice, a whole range of compensation tables exists which are distributed in a linear manner over the traversing range of the second master axis.
Below is a short example to aid understanding of this context:

Cartesian coordinate system with X, Y and Z axis
The position of the Z axis is to be compensated. Depending on the X and Y position, various compensation values result. For this reason, multiple tables describing the compensation value of Z over the X axis now need to be defined. These tables are distributed on the Y traversing range. The range over which these tables are distributed in a linear manner is defined with the position values MLStart and MLEnd (see following description of the file content).

In order to compensate a spindle error with a multi-line table, a text file with the following structure must exist:

File name: CompTable_X_Z.txt
X is the symbolic name of the first master axis, Z is the symbolic name of the axis to be compensated.

Content of first line: AnzahlStützpunkte (number of calibration points) AnzahlTabellen (number of tables) MLStart MLEnd MultiLineAchse (multi-line axis)
AnzahlStützpunkte (number of calibration points) is the number of values in a table (as above)
AnzahlTabellen (number of tables) is the number of multi-line tables. The first and the last position value of a table must always be the same. The number of lines in a table must also always be the same (AnzahlStützpunkte (number of calibration points)).
MLStart is the start value of the second master axis, MLEnd is the end value of the second master axis; the individual compensation tables are placed between these two position values.
MultiLineAchse (multi-line axis) is the axis number of the second master axis; here, a symbolic axis name is not used, but instead the number of the axis channel beginning with 0.

Content of following lines: Position value Compensation value
Structure as for the spindle error compensation; however, multiple tables are entered one after another. This is, therefore, followed by (AnzahlStützpunkte (number of calibration points) x AnzahlTabellen (number of tables)) lines in the file.

Example: The error on the Z axis is to be compensated.
Traversing range of X axis from 0 .. 500mm
Traversing range of Y axis from 0..100mm
The Y axis is the second axis in the system (index 1)

File name: CompTable_X_Z.txt

File content:

```
5          3          0.0       100.0     1
-1000.0    0.0
0.0        0.0
200.0      0.1
500.0      -0.05
1000.0     -0.05
-1000.0    0.1
0.0        0.1
200.0      0.2
500.0      -0.05
1000.0     -0.05
-1000.0    0.0
0.0        0.0
200.0      0.2
500.0      -0.05
1000.0     -0.05
```

In this example, each table has 5 calibration points; each table begins at –1000 and ends at +1000. The compensation values at the calibration points of the individual tables sometimes differ. The first table is valid for a Y value of 0.0, the third (last) table is valid for a Y value of 100.0. The second (middle) table is automatically used for the average Y value of 50.0. Intermediate values for other Y values are automatically calculated with linear interpolation.

10 Customer specific extensions and updates

The source tasks of the CNC-task files are delivered with the program package and therefore can be used from the user for amendments and adaptations. But in order to guarantee the compatibility of standard applications with future updates, user specific adaptations may be limited to specific files. So it can be guaranteed that the updates of the most important files (TASK0.SRC und TASK1.SRC) can be activated at any time. In order to install an update, the updated program version (Setup.exe) is installed over the existing installation in the same directory as before (with no prior de-installation). The application-specific files described below are not updated. After every update to McuWIN the source text files for all tasks must be compiled, e.g. with the current version of mcfg.exe, for the application-specific Include files to be valid.
Please remember: In task 0 the program TASK0.SRC is executed. It contains a command interpreter in which G-codes, M-codes and further functions are programmed.
In task 1 the monitoring program TASK1.SRC is executed.
The user has the possibility to realize in the files described below customer specific extensions / adaptations.
Warning: If other files than those listed below have been changed by the user, the relevant changes must be re-applied manually after an update. Full documentation of any changes is therefore essential.

10.1 AppTask2.SRC

This file is loaded and executed in Task2. Here customer specific single initialisations and in a closed loop cyclic monitorings can be implemented.
This file is retained during a reinstallation (new installation without previous uninstallation) of McuWIN. Thus, user-specific amendments at this point remain available, even after updates. If Task2 is to be started automatically when McuWIN is started, this must be selected accordingly in IniCFG.EXE on the “Files” tab.

10.2 System variables and special functions

Described below are system variables and functions which can be used to help in the McuWIN environment or when executing G-code programs, but which are not important for normal PCAP programming.

Special regard must be paid to the Common variables (CI0 to CI999 and CD0 to CD999) of the motion control boards. In the standard version, these variables are used to execute a data exchange between McuWIN.EXE and the task environment or between the different tasks. The operating mode of these variables must not at all be troubled by use in own functions, function extensions or accesses from the user level, as otherwise, the function of the whole packet may be significantly troubled. That is why the utmost discipline and accuracy are required when using Common variables in own extensions. Especially, the range from 0 to 99 is almost completely assigned in all cases. Also, variables that seem to be currently free may be suddenly used in future versions (e.g. after updates) and must therefore not be used by application-specific extensions. Thus, the following range conventions must be complied with:

Range	Common-Integer CI	Common-Double CD	Use
0 .. 99	System	System	McuWIN package
100..199	Send DDE	-	only in connection with DDE in application-specific extensions
200.299	Receive DDE	-	only in connection with DDE in application-specific extensions
300.399	+	+	analog joystick, assigned for system
400..599	+	+	can be used for application-specific extensions
600..999	+	+	free for user / machine operator

Exceptions of this are CI15 and CI48, which can be used for application-specific error handling (CI48) or for application-specific warning displays (CI15) (see Chapter 3.13.4) and CI58.

To protect the Common Integer ranges (CI) under 600 against accesses from the user level, it is possible to activate in IniCfg.exe, on the tab “Desktop / System”, the setting “Common-Integer Variable < 600 nur lesen:“ (read only). By default, this protection is active after a reinstallation. For the Common-Double variable, this protection does not exist.

10.2.1 System variable IPOLMODE

This is a bit-coded variable in which the interpolation axes are stored during a program’s runtime. These are stored in the least significant bits of the variable. The more significant 16 bits of the variable contain runtime information and must not be changed under any circumstances.

Programming sample:
The first 3 axes should be stored as interpolation axes in IPOLMODE.

```
IPOLMODE := (IPOLMODE and $FFFF0000) or $7;
```

11 Examples for application specific amendments

11.1 Coolant On/Off (M08 / M09)

With the command M08 the coolant shall be switched on. This is realized by using output 5 of the first axis channel. Hereto AppMCodes.INC is amended as follows:

```
// application specific m-commands
if (CI1 = 8) then begin       // M08 – Coolant On
      X.digob.5 := TRUE;      // Switch on output 5 of the X-axis
end else if (CI1 = 9) then begin    // M09 – Coolant Off
      X.digob.5 := FALSE;     // Switch off output 5 of the X-axis
end else
```

German original version :

```
// Applikationsspezifische M-Befehle
if (CI1 = 8) then begin       // M08 - Kühlmittel Ein
      X.digob.5 := TRUE;      // Ausgang 5 von X-Achse einschalten
end else if (CI1 = 9) then begin    // M09 - Kühlmittel Aus
      X.digob.5 := FALSE;     // Ausgang 5 von X-Achse ausschalten
end else
```

Note here the start of the If-Else chain, which must be ended with ”end else” to ensure that the standard command interpretation in TASK0.SRC, beginning with if ()..., follows immediately upon the above If-Else chain. This approach also guarantees that any commands re-declared by the user are actually recognised in the user program and so can be executed at this point.

11.2 Main spindle

With the S-command, the spindle speed shall be set. Here a minimum and a maximum value shall be monitored. The output of the spindle speed is realized through the analog output of the fourth axis channel (H). With the command M03 the spindle with run to the right shall be switched on. This is realized through output 6 = EIN (Spindle On) and output 7 = 0 (run to the right). With M04 the spindle with run to the left shall be switched on. This is realized through output 6 = EIN (spindle On) and output 7 = 1 (run to the left).
With M05 the spindle shall be switched off.
Here the reaching of the spindle speed is not monitored. Switching from run to the right to run to the left or reverse is not possible. When something like this is called, an error is issued with 00010000hex.

Hereto AppMCodes.INC is amended as follows:

In order to output with the S-command analog values, the AppCommands.INC is amended as follows:

```
if (CI2 = 1) then begin       // S-command:  Analog value at main spindle
                              //issue (axis H)
      // Firstly, checking the range of the analog value
      // Minimum value can be found in CI601
      // Maximum value can be found in CI602
      if CD0 < CI601 then begin
            CI10 := CI10 or $00020000;    // Indicate error in CI10
      end;
      if CD0 > CI602 then begin
            CI10 := CI10 or $00020000;    // Indicate error in CI10
      end;
      if CI10 = 0 then begin // If there is no error
            SSMSIW;          // Execute previous interpolation commands
            H.mcp := integer (CD0);       // Issue spindle speed on
                                          // analog output
      end;
end else
```

German original version:

```
if (CI2 = 1) then begin       // S-Kommando: Analogwert an Hauptspindel
                              //(Achse H) ausgeben
      // Zunächst Bereich des Analogwertes überprüfen
      // Minimalwert steht in CI601
      // Maximalwert steht in CI602
      if CD0 < CI601 then begin
            CI10 := CI10 or $00020000;    // Fehler in CI10 anzeigen
      end;
      if CD0 > CI602 then begin
            CI10 := CI10 or $00020000;    // Fehler in CI10 anzeigen
      end;
      if CI10 = 0 then begin // Wenn kein Fehler ansteht
            SSMSIW;          // vorhergehende Interpolationskommandos ausführen
            H.mcp := integer (CD0);       // Spindeldrehzahl auf
                                          // Analogausgang ausgeben
      end;
end else
```

12 DDE-communication

The program McuWIN can exchange data with a DDE-server application as DDE-client. In order to activate this functionality, the following values must be defined in the file McuWIN.INI:
Section [DDE]
DDE Server- and Topic-name:
ServerName=
TopicName=

Items for sending data (E0.? = examples)
OUT00=E0.0
OUT01=E0.1
OUT02=E0.2
...
Items for receiving data (A0.? = examples)
IN00=A0.0
IN01=A0.1
IN02=A0.2
...

The definition of IN and OUT must begin at 00 (zero and not the letter O) and be continuous. The data to be sent must be allocated to the common-integer-variables CI100 and the following. You can access to the data to be received with the die common-integer-variable CI200 and the following.
The validity of the DDE-channel is displayed in the status window of McuWIN.
At the moment the DDE-channels are provided with a clock rate of 20 ms.

13 Additional programs

13.1 RegDisp.EXE

This tool can be used to display and edit registers (CI variables), digital inputs and outputs and PCI-resources in bit form. The individual values can be grouped into pages and labelled. It is then possible to give a clear view of status information for the control and to undertake manual changes. This tool is especially helpful for program development and commissioning of a system.
The configuration of the pages and elements is handled via menu selection, right-clicking to open the desired elements. The elements programmed in this way are stored in the configuration file Inicfg.ini. After every restart, the previous position is restored. If you want to discard the whole configuration, simply remove the configuration file IniCfg.ini.

13.2 ToolEdit.EXE

This program can be used to edit the entries in the tool correction table (ToolComp.ini). The Hor.Feed and Vert.Feed fields are intended for future upgrades and do not have a function at present. From version 2.5.3.2, the current values are passed straight to the control when they are saved.
For each main level, a particular tool table can be acquired. With the level selection "All Planes", tools that are valid for all levels are acquired. With tool numbers that are acquired under both "All Planes" and a main level, the values specified for the respective levels are used.
Every time McuWIN is started, the tool table is read out from ToolComp.ini and transferred to the motion control board.

14 Notes on the use with the PA 8000 and PS840

For the ISA boards PA 8000 and PS840, there is also an McuWIN program version. However, with this version, some specifics or constraints need to be considered for setup and use.

14.1 Installation specifics

For the installation, it is not possible to use the same setup as for the PCI boards. A special installation package is required. After the installation, the communication with the mcfg program is checked at first and the base address of the board is set. This base address must be entered manually in the configuration file McuWIN.ini if the value 300hex is not set.

Entry in the section [MCU], in the variable BaseAdress:
768 corresponds to 300 hex

If a hexadecimal value is to be entered, 0x can be put in front.
Example: BaseAdress=0x320

14.2 Constraints with the PA 8000 and PS840

The ISA boards PA 8000 and PS840 do not include all functions of the boards APCI-8001, APCI-8008 and CPCI-8004. For this, you have to read the corresponding manuals. The parts Look-ahead and tool radius correction in particular are not available with these boards.

15 Error diagnosis

- **When starting a program after the compilation process Error 21 is displayed:**
  In this case the compiled program is too big for the main memory of the user task (TASK 3). In this case the system variable SZTSK3 of the control must be set to an adequate value. This value will be indicated in bytes and can have a few MB. Default is 100000.
  How to set environment variable is described in the chapter „Environment variable and control hardware“ of the „Commissioning manual“.

- **When compiling Task1.SRC in mcfg Error 89 is displayed:**
  In order to compile this file, the option „Full System“ must be set.

- **At the reference travel, the axis drives cyclically to the reference switch and again away:**
  As reference switch input not the axis specific input for position latch is used (see chapter “hardware preconditions” / reference switch). The reference switch of the corresponding axes must be wired differently or the reference travel routine TASK0.SRC must be changed.

- **Task 0 or Task 1 cannot be compiled:**
  In the first source text line appears the error message: Error 4, Duplicate Identifier.
  The cause for this is that a reserved system name is used as axis name. Check of all axis names.

- **During the referenciation the error „Conflict in the configuration data!“ (20hex) occurs:**
  The cause for this can be:
  - at least one value of the JOG-velocities or accelerations is initialised with 0 (jvl, jac, hvl. hac)
  - in the Ini-File in the section „reference travel“ [REFERENZFAHRT] under the value order one axis is indicated several times under the value order
  - in the Ini-File in the section “reference travel” [REFERENZFAHRT] under the value order an axis is indicated that is either not available on the APCI-8001 or that is in the File GCODE.INC in the constant BitAll not indicated.

- **Other causes for the error „Conflict in the configuration data!“ (20hex):**
  - in GCode.INC the axis allocations are not correct.
    (NrXAxis, NrYAxis, NrZAxis)

- **Status # 1000: Runtime error in SAP Task!**
  This error shows an error in the programming of the APCI-8001 task environment. So, it is possible that a program option is used, which is not supported by the current RWMOS-operating software.
  Using the following procedure, this error can be localised:
  Calling mcfg.exe and opening of a window „Cnc Task Status“, e.g. check with Shift-F5 if there is a runtime error at a task. If yes, here the error line and the program file can be detected.

- **An unknown error or warning is displayed**
  Within an application extension, it is possible to add to the bit allocations for the errors for the Warn variables. An error text then has to be defined in McuWIN.INI (see chapters 5.7.1 and 5.8.1).

- **Status #40: Error in reference run**
  The axis causing the error is displayed in CI17 in bit-coded form. This error may be caused by any of the following:
  - Limit switch area cannot be left with the free path
  - No limit switch found when referencing a limit switch
  - No index signal found when referencing an index signal