

---

# **POSITIONING AND CONTOURING CONTROL SYSTEM APCI-8001, APCI-8008 and CPCI-8004**

## **ELCAM Interface**



<b>1</b>	<b>Introduction .....</b>	<b>4</b>
<b>2</b>	<b>Using the ELCAM functionality .....</b>	<b>5</b>
2.1	Initialising ELCAM .....	5
2.2	Functions of the ELCAM module .....	6
2.2.1	Additional information on the activation of the AxisCompensationMode.....	7
2.3	Multi-line tables .....	9
2.4	Error request from ELCAM.....	9
2.5	Further characteristics of the ELCAM functionality .....	9
2.5.1	Repeating a table .....	9
2.5.2	Final velocity .....	9
2.5.3	Programming of several slave tables.....	9
2.6	Using ELCAM .....	10
2.7	Transfer of complete tables by PCI direct access.....	10
2.7.1	Design of the ELCAM table with 64-bit float point presentation .....	11
2.7.2	Design of the ELCAM table with 32-bit float point presentation .....	11
2.8	Execution of ELCAM tables in the PC working memory .....	12
2.8.1	PCAP function allocPhysMem .....	12
2.8.2	PCAP function freePhysMem .....	12
2.8.3	Notes on the use of physical memory.....	13
2.8.4	Use of the ELCAM table interpolation with physical memory .....	13
<b>3</b>	<b>Spindle inclination and angle error compensation .....</b>	<b>14</b>
3.1	Spindle inclination error compensation .....	14
3.2	Angle error compensation .....	14

# 1 Introduction

The ELCAM functionality of the boards APCI-8001 and CPCI-8004 enables individual axes of the system to track a master axis via a table function. To do this, a calibration table is defined with almost any variables. Intermediate values of the calibration points are calculated using linear interpolation.

The ELCAM functionality can be used for different applications:

- Multi-Line tables: As option it is possible to define a further axis, which defines a line index for the table. In this way it is possible to define a two-dimensional table, with which the slave axis is controlled, depending on two master axes.
- Several tables: It is possible to define more than one table. In this way it is possible to enable different axis bundles at the same time.
- Spindle inclination error compensation: With a specially configured table the spindle inclination error of an axis can be compensated.
- Angle error compensation. With a specially configured table the angle error of an axis system can be compensated.

The ELCAM functionality is configured via the universal object interface of the APCI-8001 / CPCI-8004. ELCAM is an option within the RWMOS.ELF operating system software. Using the fwsetup service programme, you can check if the existing RWMOS.ELF software supports this option in the booted system. By default, there is a 100,000 byte memory for ELCAM tables. If this is not sufficient, it can be increased using the SZELCAMPBUFFER environment variable. The maximum value for this memory depends on the memory configuration for the control system used.

At very large tables (a few hundred kilo bytes to a few mega byte) the load period can be a few minutes. In order to prevent this, it is possible to copy the whole table by PCI memory access directly from the PC working memory into the RAM working memory of the control. In this way the load period can be reduced significantly. For this, there are special loading commands.

## 2 Using the ELCAM functionality

### 2.1 Initialising ELCAM

The following values for the universal object interface must be used when using the ELCAM module:

Table 1: Object descriptor elements

Object descriptor element	Value
BusNumber	1200
DeviceNumber	1 With function 30 (see Table 2) a function for DeviceNumber = 0 is also available.
Index	0, 1, ... Current number of the table (is assigned by the user) Each table is referenced by a number and can serve different axis bundles.
SubIndex	Function number according to table 2.

For more information on the object descriptor elements, see the document "Universal Object Interface".

## 2.2 Functions of the ELCAM module

Table 2: Functions of the ELCAM module SubIndex (or index with DeviceNo. 1 command 30)

No.	Name	Type	Explanation
1	ERROR	integer r/w	Read/reset error status For bit coding, see table 3.
2	RESET	integer r/w	Reset table A write call of this function ends the table tracking and rejects the programmed table. This can now be reprogrammed.
3	BUFSIZE	integer r/w	Read/write number of table calibration points. This function can be used to reserve a data buffer for a table. The size cannot be increased retrospectively. The parameter/return value is the maximum number of calibration points. A table calibration point consists of two coordinate values (for master and slave axis). At a multi-line table BUFSIZE must contain all lines of a table. This buffer size must be used to reserve the memory for more than one table and has to be set one time.
4	AXIS	integer r/w	Read/write axis number of the axis which is to be tracked with this table.
5	MASTER	integer r/w	Read/write the axis number for the master axis of this table
6	GAIN	double r/w	Read/write gain of the slave axis. Gain for the table calibrations of the slave axis. Default value: 1.0
7	PHASE	double r/w	Read/write phase of the control input Value for the shift of the table calibrations for the master axis. Default value: 0.0
8	SHIFT	double r/w	Read/write shift of the slave axis. Value for the shift of the table calibrations for the slave axis. Default value: 0.0
10	MODE	integer r/w	Read/write operating type By writing on this register, the tracking mode can be activated. The tracking process can also be parameterised. For the bit coding of this register, see table 4. With every write operation on this register, the currently set bits must be transferred.
11	ADDMASTER	double w	Enter calibration point in the master table SIZEMASTER is increased. The calibration values must be constantly increasing (exception: first value in a line for multi-line tables) but must not be equidistant.
12	ADDFOLLOW	double w	Enter calibration in the slave table SIZEFOLLOW is increased.
13	SIZEMASTER	integer r	Read the number of calibrations in the master table
14	SIZEFOLLOW	integer r	Read the number of calibrations in the slave table An equal number of values must be entered in the master table and the slave table.
15	MLAXIS	integer r/w	Read/write axis for line assignment (Multi-line tables)
16	MLSTART	double r/w	Read/write start value of the line axis (Multi-line tables)
17	MLEND	double r/w	Read/write final value of the line axis (Multi-line tables)
18	MLCOUNT	integer r/w	Read/write number of lines (Multi-line tables) This value activates the multi-line function Default value: 0

No.	Name	Type	Explanation
19	CALC TARGET POSITION	double r	With this function, when the tracking mode is switched off, the current position of the tracking axis can be calculated. Thus, it is possible to drive the start position before activating the tracking mode and therefore avoid uncontrolled jumps of the axis. The calling shall be done at standing master axis. If the tracking mode is already activated during the call, the function rdOptionDbl returns the value BUSY (2). (Functions available in OS version V2.5.3.25)
20	ReadElCam Memory Address	integer r	Read the memory address of the ElCam table. This function is used internally and is not used in application programs.
21	SetTabSize	integer w	Set TabSizeMaster and TabSizeFollow. This function is only required if the execution is realized from the working memory.
30	Axis Compensation Mode	integer r/w	DeviceNr = 0 / Command 30 in <b>Index</b> : Spindle compensation mode Enable / disable with 1 / 0 (applies for all tables) DeviceNr = 1: Command 30 in <b>SubIndex</b> : Select table as set for spindle inclination error compensation The transferred parameter contains a Boolean value which indicates if the functionality is activated or deactivated. With the value 0, the respective functionality is switched off. <b>Warning:</b> Do not confuse function for Device No. 0 and Device No. 1 (see below for additional information)
31	MultiLine Linear Interpolation	integer r/w	With multi-line compensation tables, assigning a value equal to or not equal to 0 can be used to indicate whether the calibration values are interpolated between the compensation lines. If a value of 0 is assigned, no interpolation is performed, and the compensation line is converted strictly to the band limits. <b>Warning:</b> By accident, this information may be deleted again by writing on the mode register if bit 6 is not set there.
300	ElCamPhysM emAdr	integer r/w	With this function a physical memory address can be transferred to the ELCAM module. The variable BUFSIZE should only be written on after the transfer. Otherwise, it would not be possible to define tables that exceed the memory size in RWMOS.

### 2.2.1 Additional information on the activation of the AxisCompensationMode

With the ELCAM axis bundles, you have to distinguish between the normal ELCAM mode, spindle inclination error compensation (see Chapter 3.1) and angle error compensation (see Chapter 3.1). In the normal ELCAM mode, the position of the axis to be tracked results from the dependence stored in the corresponding table. Here, it is no longer possible to serve the tracked axis with traverse commands. In compensation tables, the compensation calculated from the table is added to the setpoint position. The corresponding axes can still be used like normal axes; their position, however, is corrected at any time.

Table 3: Error error status word of the ELCAM module

Bit No.	Name	Explanation
0..7		reserved
8	<b>MemErr</b>	Memory is not sufficient for the table.
9	<b>SizeErr</b>	Different number of calibrations or fewer than 2 values entered for Master and Follow
10	<b>SizeMasterErr</b>	Too many calibration values entered for master axis.
11	<b>SizeFollowErr</b>	Too many calibration values entered for slave axis.
12	<b>MasterErr</b>	Master axis defined incorrectly

Bit No.	Name	Explanation
13	<b>NotAscending</b>	Calibration value of the master axis not monotonic increasing
14	<b>LineTableError</b>	Error in the definition of multi-line tables
15	<b>MasterLineError</b>	For multi-line tables: error in initialising the multi-line axis
16	<b>MLTableSize ERROR</b>	For multi-line tables: number of table elements must be an integer multiple of the number of multi-line rows
17	<b>LineStartPos ERROR</b>	For multi-line tables: initial values for the master axis in multi-line tables must be the same in all rows
18	<b>LineEndPos ERROR</b>	For multi-line tables: final values for the master axis in multi-line tables must be the same in all rows
19..31		reserved

Table 4: Mode configuration register of the ELCAM module

Bit No.	Name	Explanation
0	<b>Run</b>	This bit is used to activate the table tracking
1	<b>RpModeTable</b>	This bit is used to set the table tracking to track actual values. The default is setpoint tracking. If in stepper systems, the AuxModeTable bit is set at the same time, the AUX register is used as an actual value. <b>Important:</b> In order for the AUX register to be used, the flag "Use encoder for position feedback" on the tab "Motor specific parameters" of the system data must be set to Yes with the respective axis.
2	<b>RpModeLine</b>	This bit is used to set the calculation of the line index according to the actual value. By default, the axis setpoint value is used here. If in stepper systems, the AuxModeLine bit is set at the same time, the AUX register is used as an actual value.
3	<b>MultiDimTable</b>	This bit shows that a multi-line table is being used. This bit is set by the system and cannot be set directly by the user.
4	<b>SinglePrecision</b>	With this bit, the data type float (32-bit float point) can be selected for the RWMOS internal presentation of the table points. Other than in the standard presentation with 64-bit float point, only half the memory space is required. This bit may be changed only if no table points have been defined yet and the number of table points (BUFSIZE) is not set yet. Otherwise, the error 10 hex will be returned.
5	<b>AxisComp Set</b>	This bit shows that this is a set for the compensation of the spindle inclination error / angle error.
6	<b>MultiLineLinear Interpolation</b>	This bit can also be set with the function SubIndex 31.
7	<b>AuxModeTable</b>	With this bit, table tracking can be set to the Auxiliary (aux) register. For this, the <b>RpModeTable</b> bit must be set at the same time. This functionality is only possible with stepper axes (master axis). Usually, for reasonable scaling of the position units, the gfaux value of the master axis must be correctly set at runtime. (This bit is only available from RWMOS V2.5.3.115.)
8	<b>AuxModeLine</b>	With this bit, the determination of the line index can be set to the Auxiliary (aux) register. For this, the <b>RpModeLine</b> bit must be set at the same time. This functionality is only possible with stepper axes (master axis). Usually, for reasonable scaling of the position units, the gfaux value of the master axis must be correctly set at runtime. (This bit is only available from RWMOS V2.5.3.115)

**Note on the configuration register:** During the configuration of the system, some of these bits may have to be initialised before the activation of table tracking (e.g. SinglePrecision). In the event of subsequent write operations, these register values must be preserved, e.g. at the start of table tracking.



Therefore, it may be useful to have a shadow variable of this register in the application program and to write this after the relevant modification.

## 2.3 Multi-line tables

To use this function, an axis must initially be defined that prescribes the table index (MLAXIS). Then the start position, which represents the index 0, must be assigned using the MasterLineStart function, and the end position, which represents the maximum index, must be assigned using the MasterLineEnd function. The number of lines is defined using the MasterLineCount function. This function is deactivated when the MasterLineCount value = 0 (default value).

When programming the calibration points for two-dimensional tables, the calibrations are programmed consecutively line by line. Each line must have the same number of calibration points, and the same start and end value for the slave axis. Before entering the calibrations, the MasterLineCount must be defined; otherwise it is not possible to begin with small values for the slave axis when entering the second line. In this case, the NotAscending error bit is set. These conditions are checked when the tracking mode is activated. In the event of an error, the LineTableError error bit is set in the status word. After tracking has started, the line number can no longer be changed.

## 2.4 Error request from ELCAM

To query an error, the error register (SubIndex = 1) must be read. For the bit coding of this register, see table 3. If an error is displayed, table tracking must not be enabled, as this may result in uncontrolled behaviour. However, the corresponding configuration bits in the mode register must already be set during the error request as otherwise, plausibility conflicts may arise.

## 2.5 Further characteristics of the ELCAM functionality

### 2.5.1 Repeating a table

A programmed table is repeated at the execution, each time after running the whole master traversing range. If the repeat is not wished, it can be switched off by programming an external value out of the master traversing range.

Table repetition is used especially where the tracking values for the first and last entries in the table are different. In this case, a gain change during tracking is only ever “dynamic” from the position of the change.

### 2.5.2 Final velocity

When the table interpolation of a tracked axis is disabled, while the axis is moving, the current velocity of the axis is kept. If this is not wished, e.g. the commando Jog-Stop (js) must be send to the respecting axis.

### 2.5.3 Programming of several slave tables

When programming/using different tables at the same time, the accesses are referenced by the table index in the index field of the corresponding ObjectDescriptor element. The table index is defined by the user.

From RWMOS V2.5.3.64 on it is possible to manage several tables for one slave axis. The activation of the Run-Mode causes automatically the deactivation of all other tables for the same axis. In this way it is possible to switch the slave directly from one table to another one.

However, the switching should be realised in a traverse range, in which the slave position of both tables is the same to prevent uncontrolled axis jumps of the slave axis. Furthermore, in this case it is not possible to vary with Shift and Gain because this also would lead to axis jumps when switching.

## 2.6 Using ELCAM

- Assignment of an index and allocation of an axis
- Allocation of the table size
- Multi-line tables: Define the number of table lines
- Multi-line tables: Set up other multi-line properties
- Enter the master and the slave tables
- Request status/error
- Detect ideal position of the slave axis and start
- Activate tracking via the mode register  
     Bit 0 = Tracking on  
     Bit 1 = Actual value tracking
- Start the master axis

## 2.7 Transfer of complete tables by PCI direct access

With very large tables (a few mega byte) the load period can be a few minutes. In order to prevent this, it is possible from RWMOS.ELF V2.5.3.67 (mcug3.dll V2.5.3.43) on to prepare a table in the PC working memory and to transfer it with a single function call into the working memory of the control. In this way the load period can be reduced to a few seconds. Here it is important to distinguish if the numbers are presented internally in 64-bit float point numbers or in 32-bit float point numbers (see also Bit 4 in the configuration register mode of the EICAM module).

The transfer of the table from the PC working memory into the working memory of the control is realized with the command *wrEICamTable64* in 64 bit float point presentation and with the command *wrEICamTable32* in 32-bit float point presentation.

The return value of this function must be checked for success. If the loading of the table was successful, the value 0 is returned. Values unequal 0 show errors according to the following table:

Return value	Description
0	Function realized successfully
7	Parameter size must be an integer product of 8 in 32-bit float point presentation or 16 in 64-bit float point presentation.
1	The EICam table is already activated or the function call was not selected according to Bit 4 (float or double) of the EICam status register; or the access to the EICAM-Object-Interface is not possible.
2	The parameter size is too large or too small.
3	System error, e.g. if the system was rebooted from another application during program execution.
4	Address of the EICAM memory cannot be detected, e.g. RWMOS does not support this function (available from version V2.5.3.67 on)
5	DLL internal memory access is not possible, EICAM memory is too large or the file mcug3.dll must be adapted.
6	System error, e.g. if the system was booted from another application during program execution.

When using these methods it is not checked if the X-values are increasing monotonously. Also the completeness of the table cannot be checked here.

For using this function, the initialisation of the ELCAM module is realized as usual.

- Setting of an index and allocation of a master and slave axis
- Setting of the SinglePrecision bits in the register mode, if necessary
- Allocation of the table size
- Multi-line tables: Define the number of table lines
- Multi-line tables: Set further multi-line characteristics

After this, instead of the ADDMASTER and ADDFOLLOW calls, x and y of a table are each written on in the PC working memory according to the structure below. After the writing on this table is finished, the table is transferred with the commands (see above) to the control. After the transfer of the table, there are the commands for the activation of the table as usual:

- Request Status / Error
- Detect the optimal position of the slave axis
- Activate the slave by the mode register  
Bit 0 = slave on  
Bit 1 = Setpoint value slave
- Starting the axis

### 2.7.1 Design of the ELCAM table with 64-bit float point presentation

double x0	double y0
double x1	double y1
double x2	double y2
...	...
double xn	double yn

Sample in C:

```
struct {
    double x, y;
} EICamTable[TABLESIZE];
```

Sample in Delphi:

```
TABLEPOINT = record
    x : double;
    y : double;
end;
ELCAMTABLE = ARRAY [0..TABLESIZE-1] of TABLEPOINT;
```

### 2.7.2 Design of the ELCAM table with 32-bit float point presentation

float x0	float y0
float x1	float y1
float x2	float y2
...	...
float xn	float yn

Beispiel in C:

```
struct {
    float x, y;
} EICamTable[TABLESIZE];
```

Sample in Delphi:

```
TABLEPOINT = record
    x : single;
    y : single;
end;
ELCAMTABLE = ARRAY [0..TABLESIZE-1] of TABLEPOINT;
```

## 2.8 Execution of ELCAM tables in the PC working memory

In order to work with tables that exceed the working memory of the APCI-8001, from RWMOS V2.5.3.68 on there is the possibility to create an ELCAM table in the PC working memory and to use it directly via PCI memory accesses. In this way the transfer to the control is not necessary anymore. However, it must be considered that the table needs to be stored in a so-called "physical memory". This memory is allocated continuously throughout the whole area and that the "physical memory address" is known. With usual data objects in PC programs, the user always works with "virtual addresses". Physical memory can be allocated with the function `allocPhysMem()`. This is a function of `mcug3.dll`. This prototype is declared in the programming language interfaces of the xPCI-800x Toolset software.

### 2.8.1 PCAP function `allocPhysMem`

<b>DESCRIPTION:</b>	With this function physical memory can be requested (allocated) from the Windows operating system.
<b>BORLAND DELPHI:</b>	function <code>allocPhysMem</code> (var <code>VirtualAdr</code> : Pointer; var <code>PhysAdr</code> : integer; size : Integer): integer;
<b>C:</b>	unsigned <code>allocPhysMem</code> (void ** <code>VirtualAdr</code> , unsigned * <code>PhysAdr</code> , unsigned size);
<b>PARAMETER:</b>	<code>VirtualAdr</code> : Pointer on the virtual address. This must be used in the Delphi-program to use the memory. <code>PhysAdr</code> : Wildcard for the physical memory address size: Size of the memory to be requested in bytes
<b>RETURN VALUE:</b>	0 when successful, error code when failed
<b>NOTE:</b>	The success of this function needs to be checked by all means. Memory areas that are allocated with this function need to be released again with <code>freePhysMem</code> before quitting the program.

### 2.8.2 PCAP function `freePhysMem`

<b>DESCRIPTION:</b>	With this function physical memory that was allocated before, can be released again.
<b>BORLAND DELPHI:</b>	function <code>freePhysMem</code> ( <code>VirtualAdr</code> : Pointer): integer;
<b>C:</b>	unsigned <code>freecPhysMem</code> (void ** <code>VirtualAdr</code> );
<b>PARAMETER:</b>	<code>VirtualAdr</code> : Pointer to the virtual address, which was delivered by <code>allocPhysMem</code> .
<b>RETURN VALUE:</b>	0 when successful
<b>NOTE:</b>	

### 2.8.3 Notes on the use of physical memory

As Windows fragments the memory area sporadically during the runtime, it is not always possible to obtain the requested memory actually. When using this function Windows XP should be used. Furthermore, it is recommended to request the memory as soon as possible after the boot of the PC as each call of the program under Windows fragments the memory more.

|

It is also possible that the memory is at your disposal at the first call of a function, but is not available anymore after a new start.

Furthermore, it is possible with the miniport driver version 8.00e, to reserve physical memory in the boot phase for the first use. With this method on a test computer with Windows XP and 1 GB memory once 512 MByte memory was as physical memory available. At this method it is necessary that exactly the reserved memory size is requested with `allcPhyMem()`.

### 2.8.4 Use of the ELCAM table interpolation with physical memory

The ELCAM tables should be used as described in the sections above. Additionally, the following aspects need to be observed:

- Use of RWMOS.ELF V2.5.3.68 or higher version
- Allocate physical memory successfully
- Transfer the physical memory address to the ELCAM module with help of the function 300
- Define the maximum table size with BUFSIZE
- Table design as described in section 2.7.1 or 2.7.2.
- The transfer of the table to the control is not necessary, but the table size must be set with the variable `SetTabSize`. Here the number of table points must be transferred.
- Initialisation and activation of the table as described above.
- Before terminating the program, release the physical memory again.

### 3 Spindle inclination and angle error compensation

With the help of tables of the ELCAM module spindle inclination error compensation or angle error compensation in right-angled system of coordinates can be realised. This functionality allows to improve the positioning precision of axis systems through a compensation of known errors with the help of compensation tables.

The “spindle inclination and angle error compensation” is only available in RWMOS.ELF from version V2.5.3.68 and under the condition that the option “optionELAM” is contained.

The calculated effective compensation value, which is finally with an axis, is subtracted from the current position during the actual value acquisition and is thus not viewable for the user. If it is to be determined or displayed, this can be done from RWMOS.ELF V2.5.3.108 using the resource # 73. This compensation value contains the sum of all effective compensation tables (inclination and angle error compensation).

#### 3.1 Spindle inclination error compensation

With this compensation, a possible error is compensated in the spindle inclination. For this, a table for the corresponding axis is created in which the master and slave axis (parameter MASTER and AXIS) is the axis to be compensated. With ADDMASTER, the corresponding values on the axis to be compensated are inserted in the table. With the function ADDFOLLOW, the corresponding compensation value, i.e. the error to be balanced is inserted. The number of Master and Follow values must be the same. The master coordinate must be rising. Start and final value of the compensation table are selected in a way that they are outside the traverse range. Position values between points of the table are interpolated linearly between the neighbouring table points.

By writing the value “1” to the variable “AxisCompensationMode”, the table must be flagged as a compensation table. For each axis, a particular spindle inclination error compensation table can be programmed.

Furthermore, the spindle compensation mode must be activated by writing on DeviceNr = 0 and Index = 30. When activating this table (Subindex 10), bit 5 (AxisCompSet) in the parameter must be set.

#### 3.2 Angle error compensation

Positioning errors caused by mechanical angle errors also can be compensated with the help of a compensation table (as in section 3.1). In this case the indices of the Master and Error axis are different. For each axis combination, an own table can be programmed. The other aspects are as described in section 3.1.

It is also possible to define multi-line tables for angle error compensation. This allows the compensation table to be selected with reference to another axis, i.e. the rows in the table are modified according to a different axis. In this mode, function 31 can also be used to specify that the compensation value should be calculated by linear interpolation between the rows.

By writing the value “1” to the variable “AxisCompensationMode”, the table must be flagged as a compensation table.