

---

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

## **Resource Interface**



|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction .....</b>   | <b>5</b>  |
| <b>2</b> | <b>Using the Resource Interface .....</b>                             | <b>6</b>  |
| 2.1      | Initialisation.....   | 6         |
| 2.2      | Functions of the Resource Interface .....                             | 7         |
| <b>3</b> | <b>The SyncMode functionality .....</b>                               | <b>11</b> |
| 3.1      | Introduction.....   | 11        |
| 3.2      | Resources of SynchMode .....  | 12        |
| 3.3      | Use instructions.....   | 12        |
| <b>4</b> | <b>Busmaster access to the working memory of the host system.....</b> | <b>14</b> |
| <b>5</b> | <b>The GEAR functionality of the APCI-8001 / CPCI-8004 .....</b>      | <b>15</b> |
| <b>6</b> | <b>ENDAT-Interface.....</b>   | <b>16</b> |
| 6.1      | Introduction.....   | 16        |
| 6.2      | Initialisation of the ENDAT interface .....                           | 16        |
| 6.3      | ENDAT objects and functions .....                                     | 16        |
| 6.4      | Note on the use of the Endat-Interface .....                          | 18        |
| <b>7</b> | <b>DMA latch with the APCI-8001 / CPCI-8004 .....</b>                 | <b>19</b> |
| 7.1      | Notes on the versions.....  | 19        |
| 7.2      | Resources for DMA-RTS handling.....                                   | 20        |
| 7.3      | The resource RTS_DATABLOCK.....                                       | 20        |
| 7.3.1    | The Index element of the resource RTS_DATABLOCK .....                 | 21        |
| 7.3.2    | The SubIndex element of the resource RTS_DATABLOCK.....               | 21        |
| 7.3.3    | Handling of the resource RTS_DATABLOCK.....                           | 21        |
| 7.4      | Note on the use of DMA-RTS .....                                      | 22        |



# 1 Introduction

The Resource Interface enables you to access internal system variables of the RWMOS operating system software directly. Furthermore, resources (system variables) can be defined, which can then be recorded using the scanner function.

The access methods to the resource interface are described in the manual “Universal Object Interface“. To use the corresponding functionalities, some options are necessary in the RWMOS.ELF operating system software. The current available options can be found after a booting process in fwsetup.exe.

## 2 Using the Resource Interface

### 2.1 Initialisation

Access to the resource interface is only available if the RWMOS.ELF operating system software contains the option „optionRESOURCE“. Moreover, for access to the PCI area of the PC, the option “optionPCI” has to be contained in RWMOS.ELF. The following values for the universal object interface must be used when using the Resource Interface:

Table 1: Object descriptor elements

| Object descriptor element | Value   |
|---------------------------|---|
| Handle                    | Must be initialised with 0 when starting the application or after rebooting the control system, and is then managed/used by the system.<br><b>For PCAP programming:</b> After the resource functionality is cleaned, the handles for all elements <b>must</b> be reset to zero. |
| BusNumber                 | 1000  |
| DeviceNumber              | 1, 2, ...<br>Function number according to table 2.  |
| Index                     | 0, 1, ...<br>Parameters of the respective function, according to table 2.   |
| SubIndex                  | Parameters of the respective function, according to table 2.<br>Unless otherwise specified = 0  |

For more information on the object descriptor elements, see the manual “Universal Object Interface”.

#### Note for PCAP programming:

- When the function Clear is called up, the handles of all option descriptor elements of the resource interface (BusNumber = 1000) are to be set to 0.  
The function Clear must not be called up as long as the resource elements are used (e.g. with the Scanner functionality)
- Access type r/w means that the relevant variable can be accessed in read-only or write mode. Note that for each access type, a separate ObjectDescriptor element has to be defined, holding the access type in question. It is **incorrect** to use access type “ATAccessInputOutput” (# 3) here.
- To access variable of type ‘float’, the function wrOptionInt or rdOptionInt should be used. The pointer to the parameter may have to be converted into an integer pointer.

## 2.2 Functions of the Resource Interface

Table 2: Functions of the Resource Interface

| Dev. No | Name            | Type        | Explanation  | Parameter Index [Subindex]                   |
|---------|-----------------|-------------|--|--|
| 0       | Clear           | integer w   | Delete existing resources.<br>This write access must be called before defining a group of resources, e.g. after restarting an application. The value 1 must be entered as the parameter value (in value).<br><b>For PCAP programming:</b> After calling 'clear', the handles for all object descriptor elements must be reset to zero. | 1<br>[0]                                     |
| 1       | Dp              | double r    | desired position - setpoint position   | Axis number (0, 1, ...)                      |
| 2       | Rp              | double r    | real position - actual position  | Axis number (0, 1, ...)                      |
| 3       | Axst            | integer r   | Axis status register<br>(see PCAP command rdaxst)  | Axis number (0, 1, ...)                      |
| 4       | Digi            | integer r   | Digital inputs<br>(see PCAP command rddigi)  | Axis number (0, 1, ...)                      |
| 5       | Scntr           | integer r   | Sample time counter<br>Counter that is increased by 1 in each scan interval.   | 0<br>[0]                                     |
| 6       | Digo            | integer r   | Digital outputs  | Axis number (0, 1, ...)                      |
| 7       | Poserr          | double r    | Position error   | Axis number (0, 1, ...)                      |
| 8       | Trvl            | double r    | Trajectory velocity of the current spooler command<br>Return takes place in the currently selected trajectory units  | Axis number (0, 1, ...)                      |
| 9       | Dv              | double r    | desired velocity - setpoint velocity   | Axis number (0, 1, ...)                      |
| 10      | Rv              | double r    | real velocity - actual velocity  | Axis number (0, 1, ...)                      |
| 11      | Aux             | double r    | aux - auxiliary register   | Axis number (0, 1, ...)                      |
| 12      | CI              | integer r/w | Common integer register  | Index (0, 1, ...999)                         |
| 13      | CD              | double r/w  | Common double register   | Index (0, 1, ...999)                         |
| 14      | Lp              | double r    | Latched position   | Axis number (0, 1, ...)                      |
| 15      | Lpndx           | double r    | Index latched position   | Axis number (0, 1, ...)                      |
| 16      | RefOffset       | double r/w  | Point zero shift for G-Code interface  | Axis number (0, 1, ...)<br>[Line No.] (0..5) |
| 17      | Mirror          | integer r/w | Axis reflection for G-Code interface<br>1 = Reflection on<br>0 = Reflection off  | Axis number (0, 1, ...)                      |
| 18      | Position Factor | double r/w  | Position factor at axis reflection<br>(Default = -1)<br>The value 0 is not allowed.  | Axis number (0, 1, ...)                      |
| 19      | LookAheadDeep   | integer r/w | Depth of the LookAhead calculation when the function AutoSpool is set in MODEREG (only for SAP programming)<br>(Default = 0)<br>With the value 0, the LookAhead calculation depth is only limited by the spooler size.   | Of no importance                             |
| 20      | DTV[0]          | double r    | Desired Trajectory Velocity, programmed value of the spooler command which is being executed   | Of no importance                             |
| 21      | DTV[1]          | double r    | Desired Trajectory Velocity, limited value of the spooler command which is being executed  | Of no importance                             |
| 22      | PIR             | integer r   | Profile Info Register  | Of no importance                             |
| 23      | PTP             | double r    | Profile Target Position  | Axis number (0, 1, ...)                      |
| 24      | TaskLineNr      | integer r   | Cnc-Task-Line No.  | Task number (0, 1, 2, 3)                     |

| Dev. No | Name                  | Type        | Explanation  | Parameter Index [Subindex]   |
|---------|-----------------------|-------------|--|--|
| 26      | mcp                   | integer r   | Motor-Command-Port   | Axis number (0, 1, ...)  |
| 27      | Backlash              | double r/w  | Backlash compensation (in axis-specific unit, default value 0)   | Axis number (0, 1, ...)  |
| 28      | MCP_MAX               | float r/w   | Maximum value of setpoint value output port (for servo axes, analog value in digits with prefix - 10V = 32768 dig.)  | Axis number (0, 1, ...)  |
| 29      | MCP_MIN               | float r/w   | Minimum value of setpoint value output port (for servo axes, analog value in digits with prefix - 10V = 32768 dig.)  | Axis number (0, 1, ...)  |
| 30      | Actual Backlash Value | double r    | Actual value of backlash compensation (in axis-specific unit, default value 0)   | Axis number (0, 1, ...)  |
| 31      | PosErrAux             | double r    | Position error AUX   | Axis number (0, 1, ...)  |
| 32      | PcapIndex             | integer r   | Profile index from spooler   | Axis number (0, 1, ...)  |
| 33      | PosKorrRot Axis       | double r    | Accumulated revolutions in rotatory systems  | Axis number (0, 1, ...)  |
| 36      | ZP Offset             | double r    | Zero Position Offset   | Axis number (0, 1, ...)  |
| 37      | DpOffset              | double r    | Position offset dpoffset (see PCAP command wrdpoffset)   | Axis number (0, 1, ...)  |
| 61      | Expand SampleTime     | integer r/w | Extension of the controller- sampling time (only at the options)   | Max. value of delay in microseconds  |
| 62      | EpmRev SdiCh0         | integer r   | Rev. No. from U23 to APCI-8001   |  |
| 63      | EpmRev SdiCh1         | integer r   | Rev.No. from U29 to OPMF   |  |
| 64      | FAST PULSE OUT        | integer r/w | <b>For special hardware versions only:</b><br>Access to these resources allows software to be used to enable a rapid hardware output (RS422 or 24V Digital Out) via a PCAP or SAP instruction. (see also section on Scan Trigger Output in the manual "Scanner Interface") | Parameters: Bit-coded value in which the outputs to be set are indicated with 1 and the outputs to be reset with 0. Each axis is assigned one bit. |
| 70      | TASK STATUS           | integer r   | Function value that is also returned with the PCAP-function gettskinfo()   | Task number (0, 1, 2, 3)   |
| 73      | Compensation Position | double r    | Effective correction value of all axis compensation tables (ELCAM module) in UserUnit  | Axis number (0, 1, ...)  |
| 100     | ain_CH                | integer r   | Analog Input Channel   | Channel number (0..7)  |
| 101     | WTLSTRB               | integer r   | Wait Latch Strobe<br>Scanner to wait until Latch Strobe is active. Latch Strobe may also be reset for a 'read'; if Latch Strobe is not set for a 'read', Busy (2) is returned. (see manual "Scanner Interface")  | Channel number (0..7)  |
| 200     | cp[]                  | double r/w  | Controller parameters column 0<br>e.g. for GEAR (chapter 1)  | Axis number (0, 1, ...) [Line] (0..14)   |
| ...     | cp[]                  | double r/w  | Controller parameters column 1..13<br>e.g. for GEAR  | Axis number (0, 1, ...) [Line] (0..14)   |
| 214     | cp[]                  | double r/w  | Controller parameters column 14<br>e.g. for GEAR   | Axis number (0, 1, ...) [Line] (0..14)   |



| Dev. No             | Name                        | Type        | Explanation   | Parameter Index [Subindex]   |
|---------------------|-----------------------------|-------------|---|--|
| 300                 | HostMem PhysAdr             | integer r/w | Physical base address in the host working memory for busmaster accesses.  | [SetNr]<br>from RWMOS V2.5.3.71 the SetNr must be entered. In this way up to 8 physical memory addresses can be administrated. |
| 301                 | HostMem Byte                | byte r/w    | 8-bit access to host working memory via busmaster access<br>Base address def. by Device 300   | Offset on base address in byte<br>[SetNr] (see Dev.No. 300)  |
| 302                 | HostMem Word                | Word r/w    | 16-bit access to host working memory via busmaster access<br>Base address def. by Device 300  | Offset on base address in byte<br>[SetNr] (see Dev.No. 300)  |
| 304                 | HostMem Int                 | integer r/w | 32-bit access to host working memory via busmaster access<br>Base address def. by Device 300  | Offset on base address in byte<br>[SetNr] (see Dev.No. 300)  |
| 305                 | HostMem Float               | float r/w   | 32-bit access to host working memory via busmaster access (floating-point)<br>Base address def. by Device 300                           | Offset on base address in byte<br>[SetNr] (see Dev.No. 300)  |
| 308                 | HostMem Double              | double r/w  | 64-bit access to host working memory via busmaster access (floating point)<br>Base address def. by Device 300                           | Offset on base address in byte<br>[SetNr] (see Dev.No. 300)  |
| 310                 | IsisAxis PhysAdr            | integer r/w | Physical base address on host working memory for busmaster accesses on ISIS axis.   | [Axis number]  |
| 311                 | IsisHost MemByte            | byte r/w    | 8-bit access to host working memory via busmaster access<br>Base address def. by Device 300   | Offset on base address in byte<br>[Axis number]  |
| 312                 | IsisHost MemWord            | Word r/w    | 16-bit access to host working memory via busmaster access<br>Base address def. by Device 310  | Offset on base address in byte<br>[Axis number]  |
| 314                 | IsisHost MemInt             | integer r/w | 32-bit access to host working memory via busmaster access<br>Base address def. by Device 310  | Offset on base address in byte<br>[Axis number]  |
| 315                 | IsisHost MemFloat           | float r/w   | 32-bit access to host working memory via busmaster access (floating point)<br>Base address def. via Device 310                          | Offset on base address in byte<br>[Axis number]  |
| 318                 | IsisHost MemDouble          | double r/w  | 64-bit access to host working memory via busmaster access (floating point)<br>Base address def. by Device 310                           | Offset on base address in byte<br>[Axis number]  |
| 320                 | IsisSensor Frequency Factor | integer r/w | Relation of the sampling frequency between Isis sensor and APCI-8001  | [Axis number]  |
| 321                 | IsisPos Norm Factor         | double r/w  | Norm factor for transfer of desired position to RayDex systems<br>Default value linear axes: 20000<br>Default value rotation axes: 4000 | [Axis number]  |
| 322                 | IsisIRQ Enable              | int         | Interrupt after change of RayDex target position values on/off  |  |
| 323                 | HwSync Strobe               | int         | Switch Sample-Timer-Synchronisation from Latch-Strobe-Signal to any fast digital input (I14, I15, I16, etc., bit coded)                 |  |
| 3000<br>...<br>3100 | ENDAT_xxx                   |             | Function group for the ENDAT interface. A detailed description can be found in chapter 6.   |  |
| 7000                | SyncMode                    |             | Function group for profile synchronisation (see chapter 3)  |  |

This list can be extended user-specifically. The driver level remains unchanged in customized extensions. Only the RWMOS.ELF operating system file must be updated.

## 3 The SyncMode functionality

### 3.1 Introduction

Using the SyncMode functionality, it is possible to track a traverse profile of a reference variable ("flying cutter"). The reference variable may be obtained by an axis which has a lower index than the tracked index. From RWMOS.ELF V2.5.3.99, this option is contained in the option "optionRESOURCE"; in earlier versions, "optionFS" was still required.

This functionality is used if the traverse movement of an axis is to be synchronised with another axis, so that e.g. a cutting tool can track a workpiece which is moving. The axis that guides the cutting tool is below referred to as "slave axis" or "tracked axis". The movement of the workpiece is controlled with the "master axis" or is determined with a position measurement system. The parameters described below are always programmed with the slave axis. As soon as the tracking parameters are programmed, a traverse profile can be loaded on the slave axis. This may be, e.g. a Jog command, but also a spooled traverse cycle consisting of multiple traverse commands. When the trigger position is reached, the execution of the traverse profile of the slave axis starts. This profile is synchronised with the master value, which means that in case of a synchronisation with the actual position, also the movement of the slave axis is stopped if the master axis blocks.

When the traverse cycle of the slave axis is finished, the tracking mode is cancelled, too. Afterwards, the slave axis can be normally traversed again. For example, it can be reset to the starting position for the next cycle.

Table 3: Initialisations for SyncMode

| Object descriptor element | Value   |
|---------------------------|---|
| Handle                    | see above   |
| BusNumber                 | 1000  |
| DeviceNumber              | 7000  |
| Index                     | respective function according to Table 4  |
| SubIndex                  | Parameter for any function according to Table 4<br>if nothing else is entered = 0 |

### 3.2 Resources of SyncMode

Table 4: Functions of SyncMode

| Index | Name            | Type        | Explanation  | Subindex                |
|-------|-----------------|-------------|--|-------------------------|
| 1     | SYNCMODE        | integer r/w | State of the synchronisations operating mode<br>0 = Idle<br>1 = Activate position tracking   | Axis number (0, 1, ...) |
| 2     | MASTER AXIS     | integer r/w | Index of the der reference axis (master axis)  | Axis number (0, 1, ...) |
| 3     | SYNC SOURCE     | integer w   | Indicates the reference variable<br>0 = dp<br>1 = rp<br>2 = aux  | Axis number (0, 1, ...) |
| 4     | START POSITION  | double r/w  | Start position of the reference axis in the user unit  | Axis number (0, 1, ...) |
| 5     | POSITION OFFSET | double r/w  | Position offset of the reference axis in the user unit   | Axis number (0, 1, ...) |
| 6     | MASTER VELOCITY | double r/w  | Setpoint velocity of the reference axis in the user unit   | Axis number (0, 1, ...) |
| 7     | GEAR FACTOR     | double r/w  | Conversion factor of the user unit in Counts / UserUnit (e.g. mm)<br>Must be written on by the user in case of tracking on aux.  | Axis number (0, 1, ...) |
| 8     | AUX FACTOR      | double r/w  | Conversion factor of Counts of the Aux-channel in Counts / of the tracking channel<br>Must be written on by the user in case of tracking on aux (default 1.0).<br>digits AX = digits AUX * AUXFACTOR | Axis number (0, 1, ...) |

### 3.3 Use instructions

First the position control loop of the slave axis must be closed. Then the values of MasterAxis, SyncSource, StartPosition, PositionOffset and MasterVelocity have to be initialised. *MasterVelocity* is the desired setpoint velocity of the master axis. *StartPosition* is the position of the master axis where tracking starts. This position can be shifted with *PositionOffset* to allow for a path for the acceleration phase. Thus it is possible that the axes are already moving synchronously when the *StartPosition* is reached.

If the master axis is to be traversed in the negative direction, a negative value must then be entered in MasterVelocity. StartPosition and PositionOffset also have to be defined by the sign regarding to the traverse direction of the master axis. Tracking is then activated by writing 1 to the variable SyncMode. Afterwards, a traverse profile can be entered in the tracking axis.

If the axes are to be synchronously traversed in one point, the trajectory velocity of the slave axis must be as high as the MasterVelocity. To ensure the synchronisation to the given start position of the master axis, the path which is covered by the master axis during the acceleration phase of the slave axis has to be entered with a negative sign in PositionOffset.

$$\text{PositionOffset} = \frac{V_{\text{Slave}}^2 * \text{MasterVelocity}}{2 * A_{\text{Slave}}}$$

In case of tracking on the system variable aux (encoder position with stepper motor axes), the conversion factor must be entered by the user to the user unit in *AuxFactor*. The unit of this conversion factor is Counts / UserUnit.

The master axis can then be started. During tracking, further traverse commands can be sent to the slave axis. When the profile of the slave axis is finished, the tracking mode is automatically deactivated. However, this mode can also be cancelled early by writing 0 to the variable SyncMode. After that, the slave axis can be normally used again.

## 4 Busmaster access to the working memory of the host system

The functions 300 to 308 allow the direct reading and writing access to the PC working memory. Required is a RWMOS operating software with the options **optionRESOURCE** and **optionPCI** from operating system version 2.5.3.13 on. Firstly, the control must be informed about the base address for the accesses. This address can be written with the function 300. The address indication must be a physical buffer address.

**Note:** No virtual buffer address, which normally is used in programs, may be used!

A respecting memory range can be allocated e.g. with the following DLL-function:

```
unsigned allocPhysMem (void **VirtualAdr, unsigned *PhysAdr, unsigned size);
```

It must be tested in any case the success of this function call. If there was an error, a value  $\neq 0$  will be returned. Memory, which was allocated in this manner, must be released before closing the application with the following DLL function:

```
unsigned freePhysMem (void *VirtualAdr);
```

These functions are realised in mcug3.dll from version 2.5.3.10 on.

**Caution:** If this functions is not used correctly, the PC system can be brought easily into a uncontrolled condition.

## 5 The GEAR functionality of the APCI-8001 / CPCI-8004

Using the GEAR functionality, it is possible to implement an electronic gear function. Here, one or more axes act as the MASTER axis for a slave axis. The gear factor must be entered in the Controller Params field (see PCAP command scp, Programming Manual) of the SLAVE axis in the line that corresponds to the respective MASTER axis (always in column 0). In closed control loops, the track mode is activated by setting the gcr variable (gear control register) for the MASTER axis (or axes): 1 corresponds to setpoint value tracking, 2 corresponds to actual value tracking. This function can be used for example for gantry axes. The user can check if this option is available when the abbreviation "GEAR" is displayed in fwsetup. For this, see also the PCAP commands scp and wrGCR / rdGCR.

When the master axis is traversed, the slave axis then follows with the set gear factor. Here, a difference between the setpoint value (dp) and the actual value (rp) is generated on the slave axis. This difference is below referred to as "internal past value" and generated or calculated by gear tracking.

### Important notes:

- For axes that are tracked to the actual position, the quantisation noises of the actual value position may be increased by a pilot control in such a way that the tracked axis (slave axis) becomes unsteady (rough run). In this case, the pilot control of the slave axis must be reduced accordingly.
- By writing -1 at the gcr variable, the internal past values of the GEAR tracking will be deleted. This value must only be written if the control loops for all SLAVE axes are opened, as these will otherwise skip a position. Moreover, tracking must be switched off on the master axis/axes (gcr = 0) to write the value -1.
- By opening a control loop, the gcr value of the corresponding axis is reset to zero for setpoint value tracking.
- If setpoint value tracking must be enabled by setting the gcr register of the MASTER axis to 1, the control loop of the MASTER axis must be first be closed. To avoid position skips, the control loop of the SLAVE axis must also be closed.
- A dynamic modification of the gear factor is not allowed.

**Caution:** Please operate very carefully with gantry axes. You can easily damage the machine. Please note the following points:

- Do not perform motion processes from mcfg without activated tracking, and accordingly, no open-loop motion processes on the slave axes
- Each time the application is to be started, check the controller against any configuration errors. If a direction inversion is set wrong for example, because the controller has been exchanged and not configured correctly, the machine can be damaged at the first motion.
- Cabling, ground and shield connections must be done very carefully and according to the current regulations in electronics.
- Limit switch and reference switch concept must be perfect.
- Error sources must be carefully monitored in the application program; especially position errors must be continuously controlled.
- Before operating the gantry axes, the reliability of the drives must be controlled in a long-time test.
- By operating / parameterising the gantry axes, the motors must be separated from the machine.
- Be cautious when other axes are to be parameterised. A gantry axis can be accidentally switched through wrong selection of an axis. For the gantry axes, avoid traversing by taking off the enable signal.
- Do not modify parameters subsequently.
- Traverse movements must under no circumstance be executed with the SLAVE axis.
- Check the application software very carefully, particularly the initialisation and handling of gear tracking.

## 6 ENDAT-Interface

### 6.1 Introduction

The ENDAT-Interface of the company HEIDENHAIN is a digital, bidirectional interface for measurement devices. This interface can give position values of incremental and absolute measurement devices and can also read out and update information that is saved in the measurement device or store new information.

Four signal lines are sufficient because of the serial data transfer. The data is transferred synchronously to clock signal that is given by the sequence-electronic (in this case APCI-8001 / CPCI-8004). The selection of the transfer manner (position values, parameter, diagnosis, etc.) occurs with mode commands, that are sent by the sequence-electronic (APCI-8001 / CPCI-8004) to the measurement device. Currently, the Endat versions 2.1 and 2.2 are supported.

The functionality of the ENDAT-interface is realised in the loadable FPGA-logic of the APCI-8001 / CPCI-8004 control and gives the user an additional hardware option. This implementation method has the advantage that the interface is handled nearly without any additional on-load of the control software and in hard real time.

### 6.2 Initialisation of the ENDAT interface

When starting the `rwmos.elf` operating system software or after a software reset `rs()` the drive axes, which were projected with ENDAT-interfaces are set nearly automatically on the specific connected encoder type. Hereto the parameters of the encoder type, e.g. incremental or rotatory measurement system, resolution of the measurement system, the resolution of the measurement system (number of databits for the absolute position value) and the measurement steps or measurement steps/turn are read out. This procedure allows a nearly automatic setup of the ENDAT-interface, independently of the used encoder type.

### 6.3 ENDAT objects and functions

The ENDAT-interface is shown in the resource interface of the RWMOS.ELF operating system software and in parts of the FPGA hardware logic and contains all important software and hardware functions for the complete use and operation of the customary ENDAT-encoder.

The ENDAT-functionality is only available when the option "optionENDAT" is contained in the operating system software. Furthermore, this option is only possible if the used hardware is adjusted for each case and if the necessary environmental variables

Furthermore, this option is only possible if the used hardware is adjusted for each case and if the necessary environmental variables are set for the corresponding axes ( $MT? = 11$  for 2.1 and  $MT? = 16$  for 2.2). For more detailed information about the environmental variables, see the commissioning manual.

The functions that are necessary for the operation of the interface are listed in the following table.

For more detailed information please refer to the document "Bidirectional synchronous serial interface for position measurement systems".



Table 5: ENDAT functions in the G3 resource interface

| Dev. No. | Name      | Type      | Description   | Parameter Index [Subindex]        |
|----------|-----------|-----------|---|-----------------------------------|
| 3000     | ENDAT_TPV | integer r | ENDAT transmit position value,<br>or: measurement system send absolute position value. Hereto APCI-8001 / CPCI-8004 sends the mode command "000111".<br>According to the ENDAT encoder type the position value is available after max 1 mx.<br>At data transfer the CRC and timeout errors are monitored.<br>Additionally, the alarm flag is updated in the alarm register. The position value (return value) is indicated as complete data word, whose length depends on the resolution of the measurement system. | Axis number (0, 1, ...)           |
| 3001     | ENDAT_SMA | integer w | ENDAT selection of memory area,<br>or: Selection of the storage range. Hereto APCI-8001 / CPCI-8004 sends the mode command "00110".<br>Before the transfer of parameters, the respecting storage range and the following MRS (Memory Range Select) code are determined. The possible storage ranges are indicated in the parameters of manufacturer of the measurement device.  | Axis number (0, 1, ...)           |
| 3002     | ENDAT_TP  | integer r | ENDAT transmit parameter,<br>or: Read parameter. After the selection of the storage range (see ENDAT_SMA), APCI-8001 / CPCI-8004 sends a complete transfer protocol, beginning with mode command read parameter „100011“, followed by <u>8 bit address</u> and 16 bit of any contents (0). The measurement device answers with the repetition of the address (will not be evaluated) and a data information of 16 bit, the contents of the parameter. The CRC check is the conclusion of the transfer cycle.        | Axis number (0, 1, ...) [Address] |
| 3003     | ENDAT_RP  | integer w | ENDAT receive parameter,<br>or: Write parameter. After the storage range selection (see ENDAT_SMA), APCI-8001 / CPCI-8004 sends a complete transfer protocol, beginning with the mode command write parameter "011100", followed by <u>8 bit address</u> and <u>16 bit parameter value</u> . The measurement device answers with the repetition of the address (will not be evaluated) and the parameter contents. At the end there is the CRC check.   | Axis number (0, 1, ...) [Address] |

| Dev. No. | Name           | Type        | Description  | Parameter Index [Subindex] |
|----------|----------------|-------------|--|----------------------------|
| 3004     | ENDAT_RR       | integer w   | ENDAT receive reset,<br>Or: send reset. The command begins with the mode command parameter send reset "101010", followed by 24 data bits with value 0. The commands allows the resetting of the measurement device at error functions or storage operations.<br>This function may be called only if the control loop of the corresponding axis is opened. Otherwise, the value 80 hex (STATE_ERR) is returned. | Axis number (0, 1, ...)    |
| 3010     | ENDAT_RA       | integer r   | ENDAT read alarm bit<br>This read register is an internal status flag, which is updated by the command ENDAT_TPV. It is a collective message. The cause for the alarm can be read out from the memory of the measurement system.   | Axis number (0, 1, ...)    |
| 3011     | ENDAT_CRC ERRS | integer r/w | ENDAT crc errors<br>This register contains the sum of all detected CRC errors that are occurred during data transfer. The register can be deleted at any time by writing the value 0.  | Axis number (0, 1, ...)    |
| 3012     | ENDAT_TOE RRS  | integer r/w | ENDAT timeout errors<br>This register contains the sum of all detected time out error that are occurred during data transfer. The register can be deleted at any time by writing the vaule 0.  | Axis number (0, 1, ...)    |
| 3013     | ENDAT_BUS ERR  | integer r/w | ENDAT gobal buserror register<br>This register contains the last detected error that is occured during data transfer. The register can be deleted at any time by writing the value 0. Internally, this register is also used for generating of an "EVENDAT"SAP event.  | Axis number (0, 1, ...)    |

## 6.4 Note on the use of the Endat-Interface

The return values of the accesses to the resource interface via PCAP programming (rdOptionInt, rdOptionDbl, wrOptionInt, wrOptionDbl) must be monitored. At the return value BUSY (2) it is necessary to repeat the calling until the value OK (4) is returned. It is normal that there must be several calls at the Endat-Interface, because here internal system states of the Endat system must be taken into consideration and must be waited. If a value different from BUSY or OK is returned, there is an error that must be treated separately.

## 7 DMA latch with the APCI-8001 / CPCI-8004

With the DMA latch operating mode of the APCI-8001 / CPCI-8004 it is possible to record position data synchronously with the external trigger signal by DMA access. This can be realized with a frequency that is significantly higher than the sampling frequency of the bearing controller (up to 30 kHz). In the following this module is named as DMA-RTS (DMA-Real-Time-Scan). The external trigger signal is led to the first axis of the system via the hardware latch strobe input. The positions latched through the hardware latch strobe are recorded.

The DMA-RTS module is operated via the resource interface. For this resource numbers from 8000 dez. and higher are foreseen. In order to access to the recorded position data (via the scanner module) the new data type **ATDataBlock** is available. The data type **ATDataBlock** has the ordinal number 6. The data type **ATDataBlock** only can be used if the updated programming language interfaces (mcug3.h, mcug3.bas, mcug3.pas – according to the used programming language) are used.

The DMA latch option can only be used for incremental encoder set value signals. This method cannot be applied for stepper signals or SSI absolute encoders.

### 7.1 Notes on the versions

To be able to use the DMA-RTS functionality, RWMOS.ELF must be equipped with the option optionDMARTS. This version is available from V2.5.3.66 or higher.

A DMA-RTS is only possible with the hardware versions of the APCI-8001 / CPCI-8004 that have the option EP1K50. This is not possible with the version EP1K30. The available version is showed in fwsetup during booting of the system.

To use a resource with the data type **ATDataBlock** in the SAP programming, mcfg must be used from version V2.5.3.59 or higher or ncc.exe (or ncc.dll).

Further improvements have been made in RWMOS.ELF from V2.5.3.75.

## 7.2 Resources for DMA-RTS handling

List of device numbers for DMA-RTS handling

| Dev. No. | Name                  | Type               | Description   | Parameter Index [Subindex]   |
|----------|-----------------------|--------------------|---|--|
| 8000     | <b>RTS_Stop</b>       | <i>integer w</i>   | Stop RTS-DMA module. After the measurement value acquisition the DMA module can be stopped. In this way on the recognition of the latch signal no data are recorded anymore.  | Of no importance   |
| 8001     | <b>RTS_Init</b>       | <i>integer w</i>   | Initialise and start RTS-DMA module. By calling this function on the recognition of a hardware latch strobe position data are recorded. This function must be called before the measurement value acquisition, i.e. directly before the scanner starts. | Of no importance   |
| 8010     | <b>RTS_DIAG</b>       | <i>integer r</i>   | Output diagnostics display in the diagnostics display.<br>Return value BUSY if no data are available<br>(only for diagnostic purposes)  |  |
| 8011     | <b>LPR_RTS</b>        | <i>short int r</i> | Reading of a latch register (16-bit) directly from the counter component  | Axis [0, 1, ..., 7]  |
| 8012     | <b>STROBE RTS</b>     | <i>short int r</i> | Reading of the latch strobes (bit codes) of all axes<br>(only for diagnostic purposes)  |  |
| 8013     | <b>RTS_DATA BLOCK</b> | <i>datablock r</i> | Scan resource<br>Description see below  | Number of axes [0..15] +<br>Axes bit coded [16..31]<br>[Max. number of data records] |

## 7.3 The resource RTS\_DATABLOCK

To record the position data recorded by DMA with the scanner module, as scan object the resource RTS\_DATABLOCK is used. The programming of the scanner is realised in analog mode, for example when scanning a position value.

However, the particularity of this resource is the design of the recorded data block that represents a data structure (Record). This data structure is structured as follows:

|  |                                 |               |                                  |
|--|---------------------------------|---------------|----------------------------------|
| integer<br>Number                        | integer<br>Status               |               |                                  |
| integer<br>Reserved                      | integer<br>Reserved             |               |                                  |
| Line 0: double<br>Position value axis 1  | double<br>Position value axis 2 | double<br>... | Double<br>Position value axis on |
| Line 1: double<br>Position value axis 1  | double<br>Position value axis 2 | double<br>... | Double<br>Position value axis on |
| ....                                     |                                 |               |                                  |
| Line zn: double<br>Position value axis 1 | double<br>Position value axis 2 | double<br>... | Double<br>Position value axis on |

The size of a data block in a scan is always fix. The number of columns of this data structure (an) is indicated in the Object-Descriptor-Element Index in low-value 16-bit. The number of lines (zn) is indicated in the Object-Descriptor-Element SubIndex. The contents of Index and SubIndex is described below in detail.

The number of lines that contain valid data can vary from scan element to scan element and is always indicated in the first element "number" of the data structure. For each active edge at the latch-input a data line is recorded during a sampling interval.

The second element in the data block „Status“ shows in bit 2 (4 hex) a possible data overflow. This is the case if not all recorded data can be entered in the above described data block. If this bit is set you may assume that DMA scan data records will get lost because the input frequency at the latch strobe input was too high or because the RTS DMA module was started earlier than the scanner.

### 7.3.1 The Index element of the resource RTS\_DATABLOCK

In this element information about the axis to be recorded is indicated. In the least significant 16-bit the number of axes to be recorded is to be entered as numeric value (max. 8). This number indicates also the number of columns (an) in the above described data record. In the more significant 16-bit of Index the axes to be recorded are specified bit coded (bit 0=1. axis; bit 1 = 2.axis; etc.).

If here are less axes than indicated in the axis number, the axes are recorded up to the indicated number.

Element Index: 32-bit

|                              |                           |
|------------------------------|---------------------------|
| MSB 16-bit<br>Axes bit coded | LSB 16-bit<br>Axes number |
|------------------------------|---------------------------|

### 7.3.2 The SubIndex element of the resource RTS\_DATABLOCK

In the Subindex element the number of lines with position data in the data structure described above is indicated (max. 128). The number of actually described lines in RWMOS.ELF is shown in the first element of the data structure "number".

Through the entered values in Index and SubIndex the size of the scan data record is specified significantly.

In order to avoid unnecessary memory space in the scanner data record and unnecessary data transfer, these values should be set only as high as necessary.

### 7.3.3 Handling of the resource RTS\_DATABLOCK

Before the use of this resource as scan object, there must be also a read process. To do this, the resource must be treated like a 32-bit whole number object, i.e. the read process is done in the PCAP programming with the function rdOptionInt. In the Value parameter of this function is indicated if already data was recorded. If the RTS-DMA module has not been initialised (Ressource # 8001 - RTS\_Init), the return value of the functions = BUSY (2). This return value is a permitted case and must not be treated like an error. However, before the scan there must be an allocation to RTS\_init.

If during the reading process, the value 8 is returned, the hardware version of the controller is not suitable for DMA-RTS.

The handle of the resource RTS\_DATABLOCK that was received during the reading, can be used as scan resource.

## 7.4 Note on the use of DMA-RTS

The record of data and the transfer of recorded real-time data into the scanner requires computing time in the real-time task of RWMOS.ELF. Therefore the sampling time should be not set under the default value of 1.28 ms. See also CM commissioning manual, keyword "Sample Time".

The use of the module DMA-RTS is realised according to the following procedure:

- Initialisation and read access to the resource RTS\_DMABLOCK
- Initialisation of the scanner using the resource RTS\_DMABLOCK
- Write access to resource RTS\_Init: thereby the DMA channel is initialised and activated
- Realise a scan with scanner module (as usual)
- Write access to the resource RTS\_Stop. Thereby the DMA cycle is stopped