POSITIONING AND CONTOURING CONTROL SYSTEM
ADDIPOS APCI-8001

Resource Interface

Rev. 7/012008

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.pdf“. 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

The functionality is only available if the RWMOS.ELF operating system software containts the option „optionRESOURCE“. 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. **For PCAP programming:** After the resource functionality is cleaned, the handles for all elements **must** be reset to zero.
BusNumber	1000
DeviceNumber	1, 2, ... Function number according to table 2.
Index	0, 1, ... Parameters of the respective function, according to table 2.
SubIndex	Parameters of the respective function, according to table 2. Unless otherwise specified = 0

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

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)

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. 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). **For PCAP programming:** After calling 'clear', the handles for all object descriptor elements must be reset to zero.	1 [0]
1	Dp	double r	Setpoint position	Axis number (0, 1, ...)
2	Rp	double r	Actual position	Axis number (0, 1, ...)
3	Axst	integer r	Axis status register (see PCAP command rdaxst)	Axis number (0, 1, ...)
4	Digi	integer r	Digital inputs (see PCAP command rddigi)	Axis number (0, 1, ...)
5	Scntr	integer r	Sample time counter Counter that is increased by 1 in each scan interval.	0 [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 Return takes place in the currently selected trajectory units	Axis number (0, 1, ...)
9	Dv	double r	Desired velocity	Axis number (0, 1, ...)
10	Rv	double r	Actual velocity	Axis number (0, 1, ...)
11	Aux	double r	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, ...) [Line No.] (0..5)
17	Mirror	integer r/w	Axis reflection for G-Code interface 1 = Reflection on 0 = Reflection off	Axis number (0, 1, ...)
18	Position Factor	double r/w	Position factor at axis reflection (Default = -1) The value 0 is not allowed	Axis number (0, 1, ...)
19	LookAhead Deep	integer r/w	Depth of the LookAhead calculation when the function AutoSpool is set in MODEREG (only for SAP programming) (Default = 0) With the value 0, the LookAhead calculation depth is only limeted by the spooler size.	Of no importance
20	DTV[0]	double r	Desired Trajectroy Velocity, programmed value	Of no importance
21	DTV[1]	double r	Desired Trajectroy Velocity, limited value	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)
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-3001	
70	TASK STATUS	integer r	Function value that is also returned with the PCAP-function gettskinfo()	Task number (0, 1, 2, 3)
100	ain_CH	integer r	Analog input channel .	Channel No. (0..7)
101	WTLSTRB	integer r	Wait Latch Strobe Wait by scanner until Latch-Strobe is active. If necessary, the latch strobe is reset during reading; if Latch-Strobe is not set during reading then Busy (2) is returned. (see Scanner-Interface.pdf)	Channel Nr. (0..7)
200	cp[][]	double r/w	Controller parameters column 0 E.g. for GEAR (see chapter 5)	Axis number (0, 1, …) [Line] (0..14)
...	cp[][]	double r/w	Controller parameters column 1.. 13 E.g. for GEAR	Axis number (0, 1, …) [Line] (0..14)
214	cp[][]	double r/w	Controller parameters column 14 E.g. for GEAR	Axis number (0, 1, …) [Line] (0..14)
300	HostMem PhysAdr	integer r/w	Physical base address in the host working memory for busmaster accesses.	[SetNr] from RWMSO 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 Base address def. by Device 300	Offset on base address in Byte [SetNr] (see Dev.No. 300)
302	HostMem Word	Word r/w	16-bit-access to host working memory via busmaster access Base address def. by Device 300	Offset on base address in Byte [SetNr] (see Dev.No. 300)
304	HostMem Int	integer r/w	32bit-access to host working memory via busmaster access Base address def. by Device 300	Offset on base address in Byte [SetNr] (see Dev.No. 300)
305	HostMem Float	float r/w	32bit-access to host working memory via busmaster Access (floating-point) Base address def. by Device 300	Offset on base address Byte [SetNr] (see Dev.No. 300)
308	HostMem Double	double r/w	64bit-access to host working memory via busmaster Access (floating point) Base address def. by Device 300	Offset on base address Byte [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 Base address def. by Device 300	Offset on base address in Byte [Axis number]
312	IsisHost	Word r/w	16-bit-access tp host working memory via	Offset on base address in

	MemWord		busmaster Access Base address def. by Device 310	Byte [Axis number]
314	IsisHost MemInt	integer r/w	32-bit-access to host working memory via Bbsmaster Access Base address def. by Device 310	Offset on base address in Byte [Axis number]
315	IsisHost MemFloat	float r/w	32-bit-access to host working memory via busmaster Access (floating-pointkt) Base address def. via Device 310	Offset on base address in Byte [Axis number]
318	IsisHost MemDouble	double r/w	64-bit-access to host working memory via busmaster Access (floating-point) Base address def. by Device 310	Offset on base address in Byte [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 Default value linear axes: 20000 Default value rotation axes: 4000	[Axis number]
322	IsisIRQ Enable	int	Interrupt after change of the RayDex Desired position values on/off.	
323	HwSync Strobe	int	Switch Sample-Timer-Syncronisation from Latch-Strobe-Signal to any fast digital input (I14, I15, I16, etc., bit coded)	
3000 ... 3100	ENDAT_xxx		Function group for the ENDAT interface. A detailed description can be found in chapter 0.	
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 as the tracked index.

Table 3: Initialisations for SyncMode

Object descriptor element	Value
Handle	see table above.
BusNumber	1000
DeviceNumber	7000
Index	any function according to Table 4
SubIndex	Parameter for any function according to Table 4 if nothing else is entered = 0

3.2 Resources of SynchMode

Table 4: Functions of SyncMode

Index	Name	Type	Explanation	Subindex
1	SYNCMODE	integer r/w	State of the synchronisations operating mode 0 = Idle 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 0 = dp 1 = rp 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) For tracking is to be described on aux by the user.	Axis number (0, 1, ...)
8	AUX FACTOR	double r/w	conversion factor of Counts of the Aux-channel in Counts / of the tracking channel For tracking is to be described on aux by the user.(default 1.0) 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. The start position is the position of the MASTER axis which is to be tracked first. 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. A traverse profile is entered in the tracking axis.
When the axes are synchronously tracked to 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 acceleration path of the slave axis has to be entered with a negative sign in in PositionOffset.
When tracking on system variable aux (encoder position by stepper motor axes) the conversion factor must be entered by the user to the user unit. The unit of the conversion factor is Counts / UserUnit.
The MASTER axis can then be started. During the tracking, other traversing commands can be sent to the slave axis. Whe the tracking is ended, the track mode must be deactivated by writing 0 to the variable SyncMode. 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 tasted in any case the success of this function call. If there was an error, a value 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 ADDIPOS products

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 cp field of the SLAVE axis in the line that corresponds to the 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: 1 corresponds to setpoint 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.

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 becomes unsteady. In this case, pilot control must be reduced accordingly.
- By writing –1 at the gcr variable, the internal past values of the GEAR tracking will be deleted. This must only be written if the control loops for all SLAVE axes are opened, as these will otherwise skip a position. The tracking must be switched off (gcr = 0) to write the value –1
- By opening a control loop, the gcr value of this axis is reset to zero for setpoint tracking.
- If the setpoint 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 skip the control loop of the SLAVE axis should 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 either motion processes from mcfg or open-loop motion processes.
- 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 earthing 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 accidently tripped through wrong selection of an axis. For the gantry axes disable the process by taking off the enable signal.
- Do no operate further parameter modifications.
- Traversing movements must under no circumstance be executed with the SLAVE axis.

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.

4 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 ADDIPOS). The selection of the transfer manner (position values, parameter, diagnosis, etc.) occurs with mode commands, that are sent by the sequence-electronic (ADDIPOS) to the measurement device.

The functionality of the ENDAT-interface is realised in the loadable FPGA-logic of the ADDIPOS 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 paramterers of the encoder type, e.g. incremental or rotatory measurmement 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.
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 synchronuous serial interface for position measurement systems”.

Table 5: ENDAT functions in the G3 resosurce interface

Dev. No.	Name	Type	Description	Parameter Index [Subindex]
3000	ENDAT_TPV	integer r	ENDAT transmit position value, or: measurement system send absolute position value. Hereto ADDIPOS sends the mode command “000111”. According to the ENDAT encoder type the position value is available after max 1 mx. At data transdfer the CRC and timeout errors are monitored. 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, or: Selection of the storage range. Hereto ADDIPOS sends the mode command “00110”. 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, or: Read parameter. After the selectionof the storage range (see ENDAT_SMA), ADDIPOS sends a complete transfer protocol, beginning with mode command read parameter „100011“, followed by 8 bit address and 16 bit of any contents (0). The measurement device answers with the repition 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 cyclus.	Axis number (0, 1, ...) [Address]
3003	ENDAT_RP	integer w	ENDAT receive parameter, or: Write parameter. After the storage range selection (see ENDAT_SMA), ADDIPOS sends a complete transfer protocol, beginning with the mode command write parameter “011100”, followed by 8 bit address and 16 bit parameter value. The measurement device answers with the repititon of the address (will not be evaluated) and the parameter contents. AT the end there is the CRC check.	Axis number (0, 1, ...) [Address]

Table 5: Sequel of the ENDAT-functions in the G3 resource interface

3004	ENDAT_RR	integer w	ENDAT receive reset, 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. 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 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 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 This register contains the sum of all detected time out error that are occured 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 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

With the DMA latch operating mode of the APCI-8001 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 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*** was introduced. 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 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).

7.2 Resources for DMA-RTS handling

List of device numbers for DMA-RTS handling

Dev. No.	Name	Type	Description	Parameter Index [Subindex]
8000	***RTS_Stop***	***integer w***	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	***RTS_Init***	***integer w***	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.	Of no importance
8010	***RTS_DIAG***	***integer r***	Output diagnostics display in the diagnostics display. Return value BUSY if no data are available (only for diagnostic purposes)	
8011	***LPR_RTS***	***short int r***	Reading of a latch register (16-bit) directly from the counter component	Axis [0, 1, .., 7]
8012	***STROBE RTS***	***short int r***	Reading of the latch strobes (bit codes) of all axes (only for diagnostic purposes)	
8013	***RTS_DATA BLOCK***	***datablock r***	Scan resource Description see below	Number of axes [0..15] + Axes bit coded [16..31] [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 Number	integer Status		
integer Reserved	integer Reserved		
Line 0: double Position value axis 1	double Position value axis 2	double ...	Double Position value axis on
Line 1: double Position value axis 1	double Position value axis 2	double ...	Double Position value axis on
....			
Line zn: double Position value axis 1	double Position value axis 2	double ...	Double Position value axis on

The size of a data block in a scan is always fix. The number of lines of this data structure (zn) is indicated in the Object-Descriptor-Element Index in low-value 16-bit. The number of colums (an) 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.

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 alsow the number of colums (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 upt to the indicated number.

Element Index: 32-bit

MSB 16-bit Axes bit coded	LSB 16-bit 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 significanty. 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.
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 realized 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