POSITIONIER- UND BAHNSTEUERUNG APCI-8001 und APCI-8008

Ressourcen-Interface

Stand: 23.12.2016, CD-ROM V2.53VK
Rev. 14/112018

www.addi-data.de

1 Einführung

Das Ressourcen-Interface ermöglicht es, direkt auf interne Systemgrößen der RWMOS-Betriebssystem-Software zuzugreifen. Weiterhin können Ressourcen (Systemgrößen) definiert werden, die dann mit der Scanner-Funktion aufgezeichnet werden können.
Der Zugriff auf das Ressourcen-Interface erfolgt mit den Methoden, die im Handbuch „Universelles Objekt-Interface“ beschrieben sind. Zur Nutzung der entsprechenden Funktionalitäten, sind entsprechende Optionen in der Betriebssystemsoftware RWMOS.ELF erforderlich. Die aktuell vorhandenen Optionen, können nach einem Bootvorgang in fwsetup.exe eingesehen werden.

2 Verwendung des Ressourcen-Interface

2.1 Initialisierung

Der Zugriff auf das Ressourcen-Interface ist nur verfügbar, wenn in der Betriebssystemsoftware RWMOS.ELF die Option „optionRESOURCE“ enthalten ist. Für Zugriffe auf den PCI-Bereich des PCs muss außerdem die Option „optionPCI“ in RWMOS.ELF enthalten sein. Folgende Werte für das "Universelle-Object-Interface" sind für die Verwendung des Ressourcen-Interface zu verwenden:

Tabelle 1: Object-Descriptor-Elemente

Object-Descriptor Element	Wert
Handle	Muss beim Start der Applikation oder nach Reboot der Steuerung mit 0 initialisiert sein und wird dann vom System geführt / verwendet. **Bei PCAP-Programmierung:** Nach einem Clean der Ressource-Funktionalität **müssen** die Handles aller Elemente genullt werden.
BusNumber	1000
DeviceNumber	1, 2, ... Funktionsnummer lt. Tabelle 2
Index	0, 1, ... Parameter zur jeweiligen Funktion lt. Tabelle 2
SubIndex	Parameter zur jeweiligen Funktion lt. Tabelle 2 falls nicht anders angegeben = 0

Weitere Informationen zu den Object-Descriptor-Elementen sind im Dokument „Universelles Objekt-Interface“ beschrieben.

Hinweis zur PCAP-Programmierung:

- Wenn die Funktion Clear aufgerufen wird, müssen die Handles alle Option-Descriptor-Elemente des Ressourcen-Interface (BusNumber = 1000) genullt werden.
- Die Funktion Clear darf nicht aufgerufen werden, solange Ressourcenelemente verwendet werden (z.B. bei der Scanner-Funktionalität)
- Der Zugriffstye r/w bedeutet, dass auf die entsprechende Variable lesend oder schreibend zugegriffen werden kann. Hierbei ist zu beachten, dass für jede Zugriffsart ein eigenes ObjectDescriptor Element definiert werden muss, in welchem die Zugriffsart hinterlegt wird. **Falsch** ist hier die Zugriffsart „ATAccessInputOutput“ (# 3) zu verwenden.
- Beim Zugriff auf Variablen vom Type float ist die Zugriffsfunktion wrOptionInt bzw. rdOptionInt zu verwenden. Ggf. muss der Zeiger auf den Parameter in einen Integerzeiger konvertiert werden.

2.2 Funktionen des Ressourcen-Interface

Tabelle 2: Funktionen Ressourcen-Interface

Dev. Nr.	Bezeichnung	Typ	Erläuterung	Parameter Index [Subindex]
0	clear	integer w	vorhanden Ressourcen löschen Dieser Schreibzugriff muss vor der Definition einer Ressourcengruppe, z.B. nach dem Neustart einer Applikation aufgerufen werden. Als Parameterwert (in value) muss 1 übergeben werden. **Bei PCAP-Programmierung:** Nach dem Aufruf von clear müssen die Handles aller Object-Descriptor-Elemente genullt werden.	1 [0]
1	dp	double r	desired position – Sollposition	Achsnummer (0, 1, ...)
2	rp	double r	real position - Istposition	Achsnummer (0, 1, ...)
3	axst	integer r	Axis Status Register - Achsenstatus (siehe PCAP Befehl rdaxst)	Achsnummer (0, 1, ...)
4	digi	integer r	Digital Inputs – Digitale Eingänge (siehe PCAP-Befehl rddigi)	Achsnummer (0, 1, ...)
5	scntr	integer r	sample time counter Zähler, der in jedem Abtastintervall um 1 erhöht wird.	0 [0]
6	digo	integer r	Digital Outputs– Digitale Ausgänge	Achsnummer (0, 1, ...)
7	poserr	double r	position error - Schleppfehler	Achsnummer (0, 1, ...)
8	trvl	double r	Trajectory Velocity – Bahngeschwindigkeit des aktuellen Spooler-Kommandos Rückgabe erfolgt in den aktuell angewählten Trajectorie-Einheiten	Achsnummer (0, 1, ...)
9	dv	double r	desired velocity - Sollgeschwindigkeit	Achsnummer (0, 1, ...)
10	rv	double r	real velocity - Istgeschwindigkeit	Achsnummer (0, 1, ...)
11	aux	double r	aux – Auxiliary Register	Achsnummer (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	Achsnummer (0, 1, ...)
15	lpndx	double r	Index Latched Position	Achsnummer (0, 1, ...)
16	RefOffset	double r/w	Nullpunkt-Verschiebung für G-Code Interface	Achsnummer (0, 1, ...) [Satz Nr] (0..5)
17	Mirror	integer r/w	Achs-Spiegelung für G-Code Interface 1 = Spiegelung ein 0 = Spiegelung aus	Achsnummer (0, 1, ...)
18	Position Factor	double r/w	Positionsfaktor bei Achs-Spiegelung (Standard = -1) Der Wert 0 ist nicht erlaubt.	Achsnummer (0, 1, ...)

Dev. Nr.	Bezeichnung	Typ	Erläuterung	Parameter Index [Subindex]
19	LookAheadDeep	integer r/w	Tiefe der LookAhead-Berechung wenn die Funktion AutoSpool in MODEREG gesetzt ist (nur für SAP-Programmierung) (Standard = 0) Mit dem Wert 0 ist die LookAhead Berechnungstiefe nur durch die Spoolergröße begrenzt.	keine Bedeutung
20	DTV[0]	double r	Desired Trajectory Velocity, programmierter Wert des gerade in Ausführung befindlichen Spoolerkommandos	keine Bedeutung
21	DTV[1]	double r	Desired Trajectory Velocity, begrenzter Wert des gerade in Ausführung befindlichen Spoolerkommandos	keine Bedeutung
22	PIR	integer r	Profile Info Register	keine Bedeutung
23	PTP	double r	Profile Target Position	Achsnummer (0, 1, ...)
24	TaskLineNr	integer r	Cnc-Task-Line Nr.	Tasknummer (0, 1, 2, 3)
26	mcp	integer r	Motor-Command-Port	Achsnummer (0, 1, ...)
27	Backlash	double r/w	Getriebespielkompensation (in achs-spezifischer Einheit, Standardwert 0)	Achsnummer (0, 1, ...)
28	MCP_MAX	float r/w	Maximalwert des Sollwert-Ausgabeports (bei Servoachsen Analogwert in digits mit Vorzeichen – 10 V entsprechen 32768 digits)	Achsnummer (0, 1, ...)
29	MCP_MIN	float r/w	Minimalwert des Sollwert-Ausgabeports (bei Servoachsen Analogwert in digits wie bei 28)	Achsnummer (0, 1, ...)
30	Actual Backlash Value	double r	aktuell verwendeter Wert der Getriebespielkompensation (in achs-spezifischer Einheit, Standardwert 0)	Achsnummer (0, 1, ...)
31	PosErrAux	double r	Schleppfehler AUX	Achsnummer (0, 1, ...)
32	PcapIndex	integer r	Profil-Index aus Spooler	Achsnummer (0, 1, ...)
33	PosKorrRotAxis	double r	aufgelaufene Umdrehungen bei rotatorischen Systemen	Achsnummer (0, 1, ...)
36	ZP Offset	double r	Zero Position Offset	Achsnummer (0, 1, ...)
37	DpOffset	double r	Positions-Offset dpoffset (siehe PCAP Kommando wrdpoffset)	Achsnummer (0, 1, ...)
38	mcpmax	double r/w	Maximalwert des Stellgrößenausgangs in Volt (Standardwert steht in system.dat)	Achsnummer (0, 1, ...)
39	mcpmin	double r/w	Minimalwert des Stellgrößenausgangs in Volt (Standardwert steht in system.dat)	Achsnummer (0, 1, ...)
40	backlash	double r/w	Hysterese der Getriebespiel-kompensation in der achsspezifischen Einheit	Achsnummer (0, 1, ...)
41	mcpcp	double r/w	Positive Nullpunkt-Kompensations-spannung in Volt (Standardwert steht in system.dat)	Achsnummer (0, 1, ...)
42	mcpcn	double r/w	Negative Nullpunkt-Kompensations-spannung in Volt (Standardwert wie bei 41)	Achsnummer (0, 1, ...)

Dev. Nr.	Bezeichnung	Typ	Erläuterung	Parameter Index [Subindex]
61	Expand Sample Time	integer r/w	Dehnung der Regler-Abtastzeit (nur bei Optionen)	Max. Wert der Verzögerung in Mikrosekunden
62	EpmRev SdiCh0	integer r	Rev.Nr. von U23 auf APCI-8001	
63	EpmRev SdiCh1	integer r	Rev.Nr. von U29 auf OPMF-3001	
64	FAST PULSE OUT	WORD r/w (16 bit)	**Nur bei spezieller Hardware-Variante:** Durch Zugriff auf diese Ressource kann per PCAP oder SAP-Anweisung ein schneller Hardware-Ausgang (RS422 oder 24V Digital Out) geschaltet werden (siehe auch Abschnitt Scan-Trigger-Output im Handbuch „Scanner-Interface“).	Parameter: bitcodierter Wert, in dem die zu setzenden Ausgänge mit 1 und die rückzusetzenden Ausgänge mit 0 gekennzeichnet sind. Jeder Achse ist ein Bit zugeordnet, d.h., mit dem Wert 4 ist die 3. Achse gemeint.
70	TASK STATUS	integer r	Funktionswert, der auch mit der PCAP-Funktion gettskinfo() zurückgeliefert wird	Tasknummer (0, 1, 2, 3)
73	Compen sation Position	double r	Wirksamer Korrekturwert aller Achs-Kompensationstabellen (ELCAM-Modul) in UserUnit	Achsnummer (0, 1, ...)
79	INV_DIGI	integer r	Invertierungsflag für digitale Eingänge	Achsnummer (0, 1, ...)
83	AOUT03	integer w	**Nur bei APCI-8008:** 4. Analog-Ausgang (16-Bit) auf Basiskarte schreiben (X2 – Pin 7)	0
87	AOUT	integer w	**Nur bei APCI-8008:** Analog-Ausgang (16-Bit) auf Basiskarte schreiben (X2 – Pin 1/3/5/7)	Analog-Kanal-Nummer (0, 1, ...) Vorsicht: nicht Achsnummer
91	DISTSUM INDEX	integer w	Mit Hilfe dieser Ressource können max. 2 Achsgruppen gebildet werden, deren Wege sich addieren oder subtrahieren.	Achsnummer (0, 1, ...)
100	ain_CH	integer r	Analog Input Channel Analog-Eingangskanal	Kanal Nr. (0..7)
101	WTLSTRB	integer r	Wait Latch Strobe Warten bei Scanner, bis Latch-Strobe aktiv ist. Ggf. wird Latch-Strobe beim Lesen zurückgesetzt; wenn Latch-Strobe beim Lesen nicht gesetzt ist, wird Busy (2) zurückgeliefert. (siehe Handbuch „Scanner-Interface“)	Kanal Nr. (0..7)
105	EVLSS	integer r/w	Freigabe/Sperren von Software-Endschaltern alternativ zu SHP (insbesondere für Absolutwert-Messsysteme)	Achsnummer (0, 1, 2, ...) aktivieren mit Wert 1, deaktivieren mit Wert 0
200	cp[][]	double r/w	Controller-Params Column 0 z.B. für GEAR (Kapitel 4)	Achsnummer (0, 1, ...) [Line] (0..14)
...	cp[][]	double r/w	Controller-Params Column 1..13 z.B. für GEAR	Achsnummer (0, 1, ...) [Line] (0..14)

Dev. Nr.	Bezeichnung	Typ	Erläuterung	Parameter Index [Subindex]
214	cp[][]	double r/w	Controller-Params Column 14 z.B. für GEAR	Achsnummer (0, 1, ...) [Line] (0..14)
300	HostMem PhysAdr	integer r/w	Physikalische Basisadresse im Host-Arbeitsspeicher für Busmaster-Zugriffe	[SetNr] ab RWMOS V2.5.3.71 muss die SetNr angegeben werden. Dadurch können bis zu 8 physische Speicher-adressen verwaltet werden.
301	HostMem Byte	byte r/w	8-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)
302	HostMem Word	Word r/w	16-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)
304	HostMem Int	integer r/w	32-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)
305	HostMem Float	float r/w	32-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff (Gleitpunkt) Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)
308	HostMem Double	double r/w	64-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff (Gleitpunkt) Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)
310	IsisAxis PhysAdr	integer r/w	Physikalische Basisadresse im Host-Arbeitsspeicher für Busmaster-Zugriffe auf Isis Achse	[Achsnummer]
311	IsisHost MemByte	byte r/w	8-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnummer]
312	IsisHost MemWord	Word r/w	16-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnummer]
314	IsisHost MemInt	integer r/w	32-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnummer]
315	IsisHost MemFloat	float r/w	32-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff (Gleitpunkt) Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnummer]
318	IsisHost MemDouble	double r/w	64-Bit-Zugriff auf Host-Arbeitsspeicher per Busmaster-Zugriff (Gleitpunkt) Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnummer]
320	IsisSensor Frequency Factor	integer r/w	Verhältnis der Abtastfrequenz zwischen Isis-Sensor und APCI-8001	[Achsnummer]
321	IsisPos Norm Factor	double r/w	Normierungsfaktor für die Sollpositionsübergabe an RayDex Systeme Standardwert Linearachsen: 20000 Standardwert Rotationsachsen: 4000	[Achsnummer]
322	IsisIRQ Enable	integer	Interrupt nach Änderung der RayDex Sollpositionswerte ein/aus	

Dev. Nr.	Bezeichnung	Typ	Erläuterung	Parameter Index [Subindex]
323	HwSync Strobe	integer	Sample-Timer-Synchronisation umschalten von Latch-Strobe-Signal auf einen beliebigen schnellen Digital-Eingang (I14, I15, I16 usw. bitcodiert)	
406	MCP OFFSET	double r/w	Offset, welcher auf den Stellgrößen-ausgang der jeweiligen Achse in der Einheit digits addiert wird	Achsnummer (0, 1, ...)
3000 ... 3100	ENDAT_xxx		Funktionsgruppe für das ENDAT-Interface. Eine ausführliche Beschreibung erfolgt im Kapitel 6.	
7000	SyncMode		Funktionsgruppe für Profil-Synchronisation (siehe Kapitel 3)	

Diese Liste kann benutzerspezifisch erweitert werden. Die Treiberebene bleibt bei kundenspezifischen Erweiterungen unberührt, lediglich das Betriebssystemfile RWMOS.ELF muss aktualisiert werden.

3 Die Funktion SyncMode

3.1 Einführung

Mit Hilfe der Funktionalität SyncMode ist es möglich, ein Verfahrprofil einer Führungsgröße nachzuführen („Fliegende Säge“ bzw. „Flying Saw“). Die Führungsgröße sollte immer von einer Achse gewonnen werden, die einen niedrigeren Index als die nachgeführte Achse hat. Ab RWMOS.ELF V2.5.3.99 ist diese Option in der Option „optionRESOURCE“ enthalten; in früheren Versionen war noch „optionFS“ erforderlich.
Diese Funktionalität wird dann angewendet, wenn die Verfahrbewegung einer Achse mit einer anderen Achse synchronisiert werden soll, um z.B. ein Bearbeitungswerkzeug einem Werkstück nachzuführen, welches sich in Bewegung befindet. Die Achse, welche das Bearbeitungswerkzeug führt, wird nachfolgend als "Slave-Achse" oder "nachgeführte Achse" bezeichnet. Die Bewegung des Werkstücks wird mit der "Master-Achse" (oder auch "Führungsachse") gesteuert oder mit einem Positionsmesssystem ermittelt. Die Programmierung der nachfolgend beschriebenen Parameter erfolgt stets bei der Slave-Achse. Nachdem die Nachführparameter programmiert sind, kann die Slave-Achse mit einem Bewegungsprofil beaufschlagt werden. Dies kann z.B. ein Jog-Kommando sein, aber auch ein gespoolter Verfahrzyklus, bestehend aus mehreren Verfahrkommandos. Beim Erreichen der Triggerposition beginnt die Ausführung des Bewegungsprofils der Slave-Achse, wobei dieses mit der Führungsgröße synchronisiert ist. Bei einer Synchronisierung mit der Istposition bedeutet dies, dass auch die Bewegung der Slave-Achse gestoppt wird, falls die Master-Achse blockiert.
Mit dem Ende des Verfahrzyklus der Slave-Achse ist auch der Nachführmodus aufgehoben. Danach kann mit der Slave-Achse wieder normal verfahren werden. Nun kann diese z.B. auf die Anfangsposition für den nächsten Zyklus zurückgestellt werden.

Tabelle 3: Initialisierungen für SyncMode

Object-Descriptor Element	Wert
Handle	siehe oben
BusNumber	1000
DeviceNumber	7000
Index	jeweilige Funktion laut Tabelle 4
SubIndex	Parameter zur jeweiligen Funktion laut Tabelle 4 falls nicht anders angegeben = 0

3.2 Ressourcen des SyncMode

Tabelle 4: Funktionen des SyncMode

Index	Bez.	Typ	Erläuterung	Subindex
1	SYNCMODE	integer r/w	Status der Synchronisations-Betriebsart 0 = Idle 1 = Positionsnachführung aktivieren	Achsnummer (0, 1, ...)
2	MASTER AXIS	integer r/w	Index der Führungsachse (Master-Achse)	Achsnummer (0, 1, ...)
3	SYNC SOURCE	integer w	Gibt die Führungsgröße an 0 = dp 1 = rp 2 = aux	Achsnummer (0, 1, ...)

Index	Bez.	Typ	Erläuterung	Subindex
4	START POSITION	double r/w	Startposition der Führungsachse in Benutzereinheit	Achsnummer (0, 1, ...)
5	POSITION OFFSET	double r/w	Positionsoffset der Führungsachse in Benutzereinheit	Achsnummer (0, 1, ...)
6	MASTER VELOCITY	double r/w	Sollgeschwindigkeit der Führungsachse in Benutzereinheit	Achsnummer (0, 1, ...)
7	GEAR FACTOR	double r/w	Umrechungsfaktor von Benutzereinheit in Counts / UserUnit (z.B. mm) Muss bei Nachführung auf aux vom Anwender beschrieben werden.	Achsnummer (0, 1, ...)
8	AUX FACTOR	double r/w	Umrechungsfaktor von Counts des Aux-Kanals in Counts / des Nachführkanals Muss bei Nachführung auf aux vom Anwender beschrieben werden (Standard 1.0) digits AX = digits AUX * AUXFACTOR	Achsnummer (0, 1, ...)

3.3 Hinweise zur Verwendung

Zunächst muss der Lageregelkreis der Slave-Achse geschlossen werden. Dann müssen die Werte von MasterAxis, SyncSource, StartPosition, PositionOffset und MasterVelocity initialisiert werden. *MasterVelocity* ist die gewünschte Sollgeschwindigkeit der Führungsachse. *StartPosition* ist die Position der Führungsachse, bei der die Nachführung beginnt. Diese kann mit *PositionOffset* verschoben werden, um einen Weg für die Beschleunigungsphase zu berücksichtigen. Somit ist es möglich, dass sich die Achsen bei Erreichen der *StartPosition* bereits synchron bewegen.
Wenn die Führungsachse in negative Richtung verfahren werden soll, dann muss bei der MasterVelocity auch ein negativer Wert eingetragen werden. StartPosition und PositionOffset müssen ebenfalls vorzeichenrichtig bezogen auf die Verfahrrichtung der Führungsachse angegeben werden. Nun wird die Nachführung „scharf gemacht“ durch Schreiben von 1 an die Variable SyncMode. Danach kann ein Verfahrprofil in die Nachführachse eingetragen werden.
Wenn die Achsen in einem Punkt synchron verfahren sollen, muss die Bahngeschwindigkeit der Nachführachse so groß wie die MasterVelocity sein. Um die Synchronität in genau der angegebenen *StartPosition* der Führungsachse zu gewährleisten, muss der Weg, den die Master-Achse in der Beschleunigungsphase der Slave-Achse zurücklegt, mit negiertem Vorzeichen in PositionOffset eingetragen werden.

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

Bei Nachführung auf die Systemgröße aux (Enkoderposition bei Schrittmotorachsen) muss der Anwender den Umrechungsfaktor auf die User-Einheit in *AuxFactor* eintragen. Dieser Umrechungsfaktor hat die Einheit Counts / UserUnit.
Nun kann die Führungsachse gestartet werden. Während der Nachführung können an die Slave-Achse weitere Verfahrbefehle geschickt werden. Wenn das Profil der Slave-Achse beendet ist, wird der Nachführmodus automatisch deaktiviert, kann aber auch durch Schreiben von 0 an die Variable SyncMode vorzeitig aufgehoben werden. Dann kann die Slave-Achse wieder normal bedient werden.

4 Busmaster Zugriffe auf den Arbeitsspeicher des Host-Systems

Mit Hilfe der Funktionen 300 bis 308 ist es möglich, direkt auf den PC-Arbeitsspeicher lesend und schreibend zuzugreifen. Voraussetzung ist eine RWMOS-Betriebssystemsoftware mit den Optionen **optionRESOURCE** und **optionPCI** ab der Betriebssystem-Version 2.5.3.13. Zunächst muss der Steuerung die Basisadresse für die Zugriffe mitgeteilt werden. Diese Adresse kann mit der Funktion 300 geschrieben werden. Bei der Adressangabe muss es sich um eine physikalische Speicheradresse handeln. Es darf keinesfalls eine normalerweise in Programmen verwendete virtuelle Speicheradresse angegeben werden.

Ein entsprechender Speicherbereich kann z.B. mit der DLL-Funktion

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

allokiert werden. Der Erfolg dieses Funktionsaufrufs muss unbedingt überprüft werden. Im Fehlerfall wird ein Wert ≠ 0 zurückgegeben.
Speicher, der auf diese Weise allokiert wurde, muss vor Beenden der Applikation mit der DLL-Funktion

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

wieder freigegeben werden. Diese Funktionen sind in mcug3.dll ab Version 2.5.3.10 realisiert.

Achtung: Durch fehlerhafte Verwendung dieser Funktionalität kann das PC-System sehr leicht in einen unkontrollierten Zustand gebracht werden.

5 Die GEAR-Funktionalität der xPCI-800x

Mit der GEAR-Funktionalität ist es möglich, eine elektronische Getriebefunktion zu realisieren. Hierbei fungieren ein oder mehrere Achsen als Führungsachse (MASTER) für eine Nachführachse (SLAVE). Der Getriebefaktor muss im Controller-Params-Feld (siehe PCAP-Kommando scp, Handbuch PHB) der SLAVE-Achse in der Zeile eingetragen werden, die der jeweiligen MASTER-Achse entspricht (immer in Spalte 0). Bei geschlossenen Regelkreisen wird der Nachführmodus durch Setzen der Variable gcr (Gear-Control-Register) bei der MASTER-Achse (den MASTER-Achsen) aktiviert: 1 entspricht Sollwertnachführung, 2 entspricht Istwertnachführung. Diese Funktion kann beispielsweise bei der Realisierung von Gantry-Achsen eingesetzt werden. Das Vorhandensein dieser Option wird in fwsetup mit dem Kürzel „GEAR“ angezeigt.
Siehe hierzu auch die PCAP-Kommandos scp und wrGCR / rdGCR.
Beim Verfahren der Masterachse folgt nun die Slaveachse mit dem eingestellten Getriebefaktor. Hierbei entsteht eine Differenz zwischen Sollwert (dp) und Istwert (rp) bei der Slaveachse. Diese Differenz wird nachfolgend als "interner Vergangenheitswert" bezeichnet und durch die Getriebenachführung erzeugt bzw. berechnet.

Wichtige Hinweise:

- Bei Achsen, die auf die Istposition nachgeführt werden, kann das Quantisierungsrauschen der Istwertposition durch eine Vorsteuerung so verstärkt werden, dass die nachgeführte Achse (Slave-Achse) unruhig wird (rauer Lauf). In diesem Fall ist die Vorsteuerung der Slave-Achse entsprechend zu verringern.
- Durch Schreiben von –1 auf gcr werden die internen Vergangenheitswerte (siehe oben) der GEAR-Nachführung gelöscht. Dieser Wert darf nur geschrieben werden, wenn die Regelkreise aller SLAVE-Achsen geöffnet sind, da diese ansonsten einen Positionssprung durchführen. Des Weiteren muss die Nachführung bei der/den Masterachse/n abgeschaltet sein (gcr = 0), um den Wert –1 schreiben zu können.
- Durch Öffnen eines Regelkreises wird der Wert gcr der entsprechenden Achse bei Sollwertnachführung genullt.
- Wenn die Sollwertnachführung aktiviert werden soll, durch Setzen des gcr-Registers der MASTER-Achse auf 1, muss vorher der Regelkreis der MASTER-Achse geschlossen sein. Um Positionssprünge zu verhindern, muss der Regelkreis der SLAVE-Achse ebenfalls geschlossen sein.
- Ein dynamisches Ändern des Getriebefaktors während der Nachführung ist nicht erlaubt.

Achtung: Bei der Realisierung von Gantry-Achsen muss mit allergrößter Sorgfalt vorgegangen werden. Ansonsten muss mit Schäden an der Maschine gerechnet werden. Folgende Dinge sind insbesondere zu beachten:

- Keine Bewegungsvorgänge aus mcfg ohne aktivierte Nachführung durchführen bzw. keine Open-Loop-Bewegungsvorgänge bei den Slave-Achsen
- Bei jedem Start der Applikation die Steuerung auf Konfigurationsfehler überprüfen. Wenn z.B. eine Richtungs-Invertierung falsch gesetzt ist, weil die Steuerungsbaugruppe getauscht und nicht richtig konfiguriert wurde, kann die Maschine bei der ersten Bewegung beschädigt werden.
- Sorgfältigste Verdrahtung, Masse und Schirmverbindungen müssen nach den anerkannten Regeln der Elektrotechnik ausgeführt werden.
- Einwandfreies End- und Referenzschalterkonzept.
- Sorgfältige Fehlerüberwachung im Applikationsprogramm, insbesondere der Schleppfehler muss ständig überwacht werden.
- Vor der Inbetriebnahme der Gantry-Achsen die Antriebe in einem Dauertest auf Zuverlässigkeit prüfen.
- Bei der Inbetriebnahme / Parametrierung der Gantry-Achsen die Motoren von der Maschine abkoppeln.

- Vorsicht, wenn andere Achsen parametriert werden. Durch falsche Achsauswahl kann hier versehentlich eine der Gantry-Achsen angesprochen werden. Bei diesen gegebenenfalls durch Wegnahme des Freigabesignals oder Ähnlichem ein Verfahren verhindern.
- Keine nachträglichen Parameteränderungen mehr durchführen.
- Keinesfalls Verfahrbewegungen mit der SLAVE-Achse durchführen.
- Sorgfältigste Prüfung der Applikationssoftware, insbesondere die Initialisierung und Handhabung der Getriebenachführung.

6 ENDAT-Interface

6.1 Einführung

Das ENDat-Interface der Firma HEIDENHAIN ist eine digitale, bidirektionale Schnittstelle für Messgeräte. Sie ist in der Lage, sowohl Positionswerte von inkrementalen und absoluten Messgeräten auszugeben, als auch im Messgerät gespeicherte Informationen auszulesen, zu aktualisieren oder neue Informationen abzulegen. Aufgrund der seriellen Datenübertragung sind 4 Signalleitungen ausreichend. Die Daten werden synchron zu dem von der Folge-Elektronik (hier xPCI-800x) vorgegebenen Taktsignal übertragen. Die Auswahl der Übertragungsart (Positionswerte, Parameter, Diagnose ...) erfolgt mit Mode-Befehlen, welche die Folge-Elektronik (xPCI-800x) an das Messgerät sendet. Derzeit werden die Endat-Versionen 2.1 und 2.2 unterstützt.

Die Funktionalität der ENDAT-Schnittstelle wird in der ladbaren FPGA-Logik der xPCI-800x-Steuerung realisiert und stellt aus Anwendersicht eine zusätzliche Hardwareoption dar. Diese Implementierungsmethode hat den Vorteil, dass die Schnittstelle weitestgehend ohne Zusatzbelastung der Steuerungssoftware und in harter Echtzeit bedient wird.

6.2 Initialisierung der ENDAT-Schnittstelle

Beim Start der rwmos.elf-Betriebssystem-Software bzw. nach einem Software-Reset rs() werden die mit ENDAT-Schnittstellen projektierten Antriebsachsen weitestgehend automatisch auf den jeweils angeschlossenen Gebertyp eingestellt. Dazu werden die Kennwerte des Gebertyps wie z.B. inkrementelles oder rotatorisches Messsystem, die Auflösung des Messsystems (Anzahl der Datenbits für den absoluten Positionswert) und die Messschritte bzw. Messschritte / Umdrehung ausgelesen. Diese Prozedur gestattet einen weitestgehend automatischen Setup der ENDAT-Schnittstelle unabhängig vom eingesetzten Encodertyp.

6.3 ENDAT-Objekte und -Funktionen

Die ENDAT-Schnittstelle wird im Ressourcen-Interface der RWMOS.ELF-Betriebssystem-Software und Teile der FPGA-Hardware-Logik abgebildet und beinhaltet alle wesentlichen Software- und Hardwarefunktionen zur vollständigen Nutzung und Inbetriebnahme der handelsüblichen ENDAT-Encoder.
Die ENDAT-Funktionalität ist nur verfügbar, wenn in der Betriebssystemsoftware RWMOS.ELF die Option „optionENDAT“ enthalten ist. Des Weiteren ist diese Option nur möglich, wenn die verwendete Hardware für diesen Einsatzfall angepasst ist, und wenn die notwendigen Umgebungsvariablen für die entsprechenden Achsen gesetzt sind (MT? = 11 für 2.1 und MT? = 16 für 2.2). Nähere Informationen zu den Umgebungsvariablen sind im Handbuch IHB zu finden.
In nachfolgend aufgeführter Tabelle stehen die zur Bedienung der Schnittstelle notwendigen Funktionen.
Für weitere und detailliertere Informationen über das ENDAT-Interface sollte das Dokument „Bidirektionales synchron-serielles Interface für Positionsmesssysteme“ herangezogen werden.

Tabelle 5: ENDAT-Funktionen im Ressourcen-Interface

Dev. Nr.	Bez.	Typ	Erläuterung	Parameter Index [Subindex]
3000	ENDAT_TPV	integer r	ENDAT transmit position value, oder: Messsystem sende absoluten Positionswert. xPCI-800x sendet hierzu den Mode-Befehl “000111”. Je nach ENDAT-Gebertyp steht der Positionswert nach max. 1ms zur Verfügung. Bei der Datenübertragung erfolgt die Überwachung von CRC- und Timeoutfehlern. Zusätzlich wird das Alarmflag im Alarmregister aktualisiert. Der Positionswert (Rückgabewert) wird als komplettes Datenwort ausgegeben, dessen Länge von der Auflösung des Messgeräts abhängt.	Achsnummer (0, 1, ...)
3001	ENDAT_SMA	integer w	ENDAT selection of memory area, oder: Auswahl des Speicherbereiches. XPCI-800X sendet hierzu den Mode-Befehl “001110”. Vor der Übertragung von Parametern wird der entsprechende Speicherbereich mit dem Mode-Befehl Auswahl des Speicherbereichs und einem anschließenden MRS (Memory Range Select)-Code bestimmt. Die möglichen Speicherbereiche sind in den Parametern des Messgeräte-Herstellers abgelegt.	Achsnummer (0, 1, ...)
3002	ENDAT_TP	integer r	ENDAT transmit parameter, oder: Parameter lesen. Nach der Speicherbereichsauswahl (siehe ENDAT_SMA) sendet XPCI-800X ein vollständiges Übertragungsprotokoll beginnend mit dem Mode-Befehl Parameter lesen “100011”, gefolgt von 8-Bit Adresse und 16-Bit beliebigen Inhalts (0). Das Messgerät antwortet mit der Wiederholung der Adresse (wird nicht ausgewertet) und einer 16 Bit langen Dateninformation, dem Inhalt des Parameters. Den Abschluss des Übertragungszyklus bildet der CRC-Check.	Achsnummer (0, 1, ...) [Adresse]
3003	ENDAT_RP	integer w	ENDAT receive parameter, oder: Parameter schreiben. Nach der Speicherbereichsauswahl (siehe ENDAT_SMA) sendet die xPCI-800x ein vollständiges Übertragungsprotokoll beginnend mit dem Mode-Befehl Parameter schreiben „011100“, gefolgt von 8-Bit-Adresse und 16-Bit-Parameterwert. Das Messgerät antwortet mit der Wiederholung der Adresse (wird nicht ausgewertet) und des Parameterinhalts. Zum Abschluss folgt der CRC-Check.	Achsnummer (0, 1, ...) [Adresse]

Dev. Nr.	Bez.	Typ	Erläuterung	Parameter Index [Subindex]
3004	ENDAT_RR	integer w	ENDAT receive reset, oder: sende Reset. Das Kommando beginnt mit dem Mode-Befehl Parameter Reset senden „101010“, gefolgt von 24 Datenbits mit dem Wert 0. Das Kommando dient zum Rücksetzen des Messgeräts bei Fehlfunktionen oder Speicheroperationen. Diese Funktion darf nur aufgerufen werden, wenn der Regelkreis der entsprechenden Achse geöffnet ist, ansonsten wird der Wert 80 hex (STATE_ERR) zurückgeliefert.	Achsnummer (0, 1, ...)
3010	ENDAT_RA	integer r	ENDAT read alarm bit Bei diesem Leseregister handelt es sich um eine internes Statusflag, welches durch das Kommando ENDAT_TPV aktualisiert wird. Es handelt sich um eine Sammelmeldung. Die Ursache für den Alarm kann aus dem Speicher des Messsystems ausgelesen werden.	Achsnummer (0, 1, ...)
3011	ENDAT_CRC ERRS	integer r/w	ENDAT crc errors Dieses Register beinhaltet die Summe aller detektierten CRC-Fehlern die während der Datenübertragung aufgetaucht sind. Das Register kann jederzeit durch Beschreiben mit dem Wert 0 gelöscht werden.	Achsnummer (0, 1, ...)
3012	ENDAT_TOE RRS	integer r/w	ENDAT timeout errors Dieses Register beinhaltet die Summe aller detektierten Timeout-Fehlern die während der Datenübertragung aufgetaucht sind. Das Register kann jederzeit durch Beschreiben mit dem Wert 0 gelöscht werden.	Achsnummer (0, 1, ...)
3013	ENDAT_BUS ERR	integer r/w	ENDAT gobal buserror register Dieses Register beinhaltet den zuletzt detektierten Fehler, der während der Datenübertragung aufgetreten ist. Das Register kann jederzeit durch Beschreiben mit dem Wert 0 gelöscht werden. Intern wird dieses Register auch beim Auslösen eines SAP-Events „EVENDAT“ herangezogen.	Achsnummer (0, 1, ...)

6.4 Hinweis zum Zugriff auf das Endat-Interface

Die Rückgabewerte der Zugriffe auf das Ressourceninterface per PCAP-Programmierung (rdOptionInt, rdOptionDbl, wrOptionInt, wrOptionDbl) müssen auf jeden Fall überwacht werden. Beim Rückgabewert BUSY (2) ist es erforderlich den Aufruf zu wiederholen, bis der Wert OK (4) zurückgeliefert wird. Beim Endat-Interface ist es normal, dass die Aufrufe mehrfach durchgeführt werden müssen, weil hier interne Systemzustände des Endat-Systems berücksichtigt und abgewartet werden müssen.
Wenn ein anderer Wert als BUSY oder OK zurückgeliefert wird, liegt ein Fehler vor, der gesondert behandelt werden muss.

7 DMA-Latch mit der APCI-8001 / APCI-8008

Als DMA-Latch wird eine Betriebsart der APCI-8001 / APCI-8008 bezeichnet, mit welcher es möglich ist, Positionsdaten synchron mit einem externen Triggersignal per DMA-Zugriffen aufzuzeichnen. Dies kann mit einer Frequenz erfolgen, die wesentlich höher ist als die Abtastfrequenz des Lagereglers (bis ca. 30 kHz). Dieses Modul wird nachfolgend auch mit dem Kürzel DMA-RTS (DMA-Real-Time-Scan) bezeichnet. Das externe Triggersignal wird über den Hardware-Latch-Strobe-Eingang der ersten Achse des Systems zugeführt. Aufgezeichnet werden die per Hardware-Latch-Strobe gelatchten Positionen.
Das DMA-RTS Modul wird über das Ressourcen-Interface bedient. Hierzu sind Ressourcen-Nummern ab 8000 dez. vorgesehen. Um auf die aufgezeichneten Positionsdaten (per Scannermodul) zugreifen zu können, steht der Datentyp ***ATDataBlock*** zur Verfügung. Der Datentyp ***ATDataBlock*** hat die Ordnungszahl 6. Damit der Datentyp ***ATDataBlock*** verwendet werden kann, müssen unbedingt die aktuellen Hochspracheninterfaces (mcug3.h, mcug3.bas, mcug3.pas – je nach eingesetzter Programmiersprache) verwendet werden.
Die DMA-Latch Option ist nur verwendbar für Inkrementalgeber-Istwertsignale. Bei Schrittsignalen oder SSI-Absolutwertgebern kann diese Methode nicht angewendet werden.

7.1 Versionshinweise

Um die Funktionalität DMA-RTS verwenden zu können, muss RWMOS.ELF mit der Option optionDMARTS ausgestattet sein. Diese Version ist erst verfügbar ab V2.5.3.66.
Weiterhin ist ein DMA-RTS nur möglich mit Hardwarevarianten der APCI-8001 / APCI-8008, die mit der Option EP1K50 ausgestattet sind. Bei der Variante EP1K30 ist DMA-RTS nicht möglich. Die vorhandene Variante wird in fwsetup beim Booten des Systems angezeigt.
Um eine Ressource mit dem Datentyp ***ATDataBlock*** in der SAP-Programmierung verwenden zu können, muss mcfg ab Version V2.5.3.59 bzw. ncc.exe (oder ncc.dll) ab Version V2.5.3.41 verwendet werden.
Weitere Verbesserungen wurden in RWMOS.ELF ab V2.5.3.75 durchgeführt.

7.2 DMA RTS mit DMA-Scan von analogen Eingängen

Wenn in RWMOS.ELF die Option RTS_ANALOGIN enthalten ist, ist ein DMA-Scan immer ein Scan mit analogen Eingängen. Die Ressource 8013 ist dann ersetzt durch die Ressource 8014.

7.3 Ressourcen für DMA-RTS-Handling

Liste der Device-Nummern für das DMA-RTS Handling

Dev. Nr.	Bez.	Typ	Erläuterung	Parameter Index [Subindex]
8000	***RTS_Stop***	***integer w***	RTS-DMA Modul anhalten. Nach der Messwertaufnahme kann das RTS-DMA Modul gestoppt werden. Dadurch werden bei Erkennung eines Latch-Signales keine Daten mehr aufgezeichnet.	ohne Bedeutung
8001	***RTS_Init***	***integer w***	RTS-DMA Modul initialisieren und starten. Durch den Aufruf dieser Funktion werden beim Erkennen eines Hardware-Latch-Strobes Positionsdaten aufgezeichnet. Diese Funktion muss vor der Messwert-aufnahme, d.h. direkt vor dem Start des Scanners aufgerufen werden.	ohne Bedeutung
8010	***RTS_DIAG***	***integer r***	Diagnose-Anzeige in fwsetup ausgeben Rückgabewert BUSY, wenn keine Daten vorliegen (nur für Diagnose-Zwecke)	
8011	***LPR_RTS***	***short int r***	Lesen eines Latch-Registers (16 Bit) direkt aus dem Zählerbaustein	Achse [0, 1, .., 7]
8012	***STROBE RTS***	***short int r***	Lesen der Latch-Strobes (bitweise codiert) aller Achsen (nur für Diagnose-Zwecke)	
8013	***RTS_DATA BLOCK***	***datablock r***	Scan-Ressource Beschreibung siehe unten	Anzahl der Achsen [0..15] + Achsen bitcodiert [16..31] [Max Anzahl der Datensätze]
8014	***RTS_DATA ANALOG BLOCK***	***datablock r***	Scan-Ressource Beschreibung siehe unten	Anzahl der Achsen [0..15] + Achsen bitcodiert [16..31] [Max Anzahl der Datensätze]

7.4 Die Ressource RTS_DATABLOCK

Um die per DMA aufgezeichneten Positionsdaten mit dem Scannermodul aufzuzeichnen wird, als Scan-Objekt die Ressource RTS_DATABLOCK verwendet. Die Programmierung des Scanners erfolgt analog, wie z.B. beim Scannen eines Positionswerts.
Die Besonderheit bei dieser Ressource ist jedoch der Aufbau des aufgezeichneten Datenblocks, der selbst eine Datenstruktur (Record) darstellt. Der Aufbau dieser Datenstruktur sieht folgendermaßen aus:

integer Anzahl	integer Status		
integer Reserviert	integer Reserviert		
Zeile 0: double Positionswert Achse 1	double Positionswert Achse 2	double ...	double Positionswert Achse an
Zeile 1: double Positionswert Achse 1	double Positionswert Achse 2	double ...	double Positionswert Achse an
....			
Zeile zn: double Positionswert Achse 1	double Positionswert Achse 2	double ...	double Positionswert Achse an

Die Größe des Datenblocks in einem Scan ist immer fest. Die Anzahl der Spalten dieser Datenstruktur (an) wird im Objekt-Descriptor-Element Index in den niederwertigen 16 Bit angegeben. Die Anzahl der Zeilen (zn) wird im Objekt-Descriptor-Element SubIndex angegeben. Der Inhalt von Index und SubIndex wird nachfolgend noch näher erläutert.
Die Anzahl der Zeilen, welche gültige Daten enthalten, kann von Scan-Element zu Scan-Element variieren und ist jeweils im ersten Element „Anzahl“ der Datenstruktur angegeben. Für jede aktive Flanke am Latch-Eingang während eines Abtastintervalls wird eine Datenzeile aufgezeichnet.
Das zweite Element im Datenblock „Status“ zeigt in Bit 2 (4 hex) einen eventuellen Datenüberlauf an. Dies ist dann der Fall, wenn nicht alle aufgezeichneten Daten im oben beschriebenen Datenblock eingetragen werden können. Wenn dieses Bit gesetzt ist, muss damit gerechnet werden, dass DMA-Scan-Datensätze verloren gegangen sind, weil die Eingangsfrequenz an Latch-Strobe-Eingang zu hoch war oder das RTS-DMA-Modul vor dem Scanner gestartet wurde.

7.4.1 Das Element Index der Ressource RTS_DATABLOCK

In diesem Element werden Informationen über die aufzuzeichnenden Achsen angegeben. In den niederwertigsten 16 Bit ist die Anzahl der aufzuzeichnenden Achsen als Zahlenwert anzugeben (max. 8). Diese Anzahl gibt auch die Anzahl der Spalten (an) im oben beschriebenen Aufzeichnungsdatensatz an.
In den höherwertigen 16 Bit von Index werden die aufzuzeichnenden Achsen bitcodiert spezifiziert (Bit 0 = 1. Achse; Bit 1 = 2. Achse usw.).
Wenn hier weniger Achsen als in der Achsanzahl angegeben sind, ist der Inhalt der unbenutzten Spalten undefiniert und darf nicht ausgewertet werden.
Wenn mehr Achsen als in der Achsanzahl angegeben sind, werden nur Achsen, aufsteigend bis zur angegebenen Anzahl aufgezeichnet.

Element Index: 32 Bit

MSB 16 Bit Achsen bitcodiert	LSB 16 Bit Achsen Anzahl

7.4.2 Das Element SubIndex der Ressource RTS_DATABLOCK

Im Element Subindex wird die Anzahl der Zeilen mit Positionsdaten in der oben beschriebenen Datenstruktur angegeben (max. 128). Die Anzahl der von RWMOS.ELF tatsächlich beschriebenen Zeilen werden im
1. Element der Datenstruktur „Anzahl“ angezeigt.
Durch die eingetragenen Werte in Index und SubIndex wird die Größe des Scan-Datensatzes in hohem Maße spezifiziert. Um unnötigen Speicherbedarf im Scan-Datensatz und unnötigen Datentransfer zu vermeiden, sollten diese Werte nur so groß wie notwendig eingestellt werden.
Die minimale Anzahl der Zeilen ergibt sich aus der Abtastrate der Steuerung und der maximalen RTS-Triggerfrequenz. In jedem DATABLOCK müssen auf jeden Fall mehr Zeilen aufgezeichnet werden können, als Triggersignale in einem Abtastintervall (= 1 Datenblock) abgearbeitet werden. Hier sollte man ca. 20% Überhang einplanen.

Beispiel:
Standardabtastrate = 1,28 ms
Maximale Triggerfrequenz = 10 kHz

Anzahl der Messwerte = 10 kHz * 1,28 ms = 12,8
Es werden also in jedem Abtastintervall 12 oder 13 Zeilen im Datenblock aufgezeichnet.

7.4.3 Die Handhabung der Ressource RTS_DATABLOCK

Vor der Verwendung dieser Ressource als Scan-Objekt muss mit dieser ein Lesevorgang durchgeführt werden. Hierzu muss die Ressource wie ein 32-Bit Ganzzahlobjekt behandelt werden, d.h. der Lesevorgang wird in der PCAP-Programmierung mit der Funktion rdOptionInt durchgeführt. Allerdings muss bei der Initialisierung des ObjectDescriptorElements als Datentyp ATDataBlock (6) angegeben werden.
Im Parameter Value dieser Funktion wird angezeigt, ob bereits Daten aufgezeichnet wurden. Wenn das RTS-DMA-Modul zuvor noch nicht initialisiert wurde (Ressource # 8001 - RTS_Init), ist der Rückgabewert der Funktion = BUSY (2). Dieser Rückgabewert stellt ein erlaubter Betriebsfall dar und darf nicht als Fehler behandelt werden. Vor dem Scan muss jedoch noch eine Zuweisung an RTS_Init erfolgen.
Wenn beim Lesevorgang der Wert 8 zurückgeliefert wird, ist die Hardware-Version der Steuerung nicht für DMA-RTS geeignet.
Der Rückgabewert 1 zeigt an, dass die Ressource in RWMOS nicht bekannt ist. In diesem Fall wird vermutlich ein ungeeignetes RWMOS.ELF verwendet.
Wenn beim Lesen der Rückgabewert 2 oder 4 zurückgeliefert wird, kann das beim Lesen erhaltene Handle der Ressource RTS_DATABLOCK auf jeden Fall als Scan-Ressource verwendet werden.

7.5 Die Ressource RTS_DATAANALOGBLOCK

Um die per DMA aufgezeichneten Positionsdaten mit analogen Eingangsinformationen mit dem Scanner-Modul aufzuzeichnen, wird als Scan-Objekt die Ressource RTS_DATAANALOGBLOCK verwendet. Die Programmierung des Scanners erfolgt analog, wie z.B. beim Scannen eines Positionswerts bzw. RTS_DATABLOCK.
Der Aufbau dieser Ressource ist um Zeilen mit je 8 Analogwerten (je 16-Bit-Ganzzahl) ergänzt. Der Aufbau dieser Datenstruktur sieht folgendermaßen aus:

integer Anzahl	integer Status		
integer Reserviert	integer Reserviert		
Zeile 0: double Positionswert Achse 1	double Positionswert Achse 2	double ...	double Positionswert Achse an
Zeile 1: double Positionswert Achse 1	double Positionswert Achse 2	double ...	double Positionswert Achse an
....			
Zeile zn: double Positionswert Achse 1	double Positionswert Achse 2	double ...	double Positionswert Achse an
Zeile 0: short integer Analogwert Kanal 1	short integer Analogwert Kanal 2	short integer ...	short integer Analogwert Kanal 8
Zeile 1: short integer Analogwert Kanal 1	short integer Analogwert Kanal 2	short integer ...	short integer Analogwert Kanal 8
....			
Zeile zn: short integer Analogwert Kanal 1	short integer Analogwert Kanal 2	short integer ...	short integer Analogwert Kanal 8

Die Größe des Datenblocks in einem Scan ist immer fest. Die Anzahl der Spalten dieser Datenstruktur (an) wird im Objekt-Descriptor-Element Index in den niederwertigen 16 Bit angegeben. Die Anzahl der Zeilen (zn) wird im Objekt-Descriptor-Element SubIndex angegeben. Der Inhalt von Index und SubIndex wird in den Kapiteln 7.4.1 und 7.4.2 näher erläutert.
Die Anzahl der Zeilen, welche gültige Daten enthalten, kann von Scan-Element zu Scan-Element variieren und ist jeweils im ersten Element „Anzahl“ der Datenstruktur angegeben. Für jede aktive Flanke am Latch-Eingang während eines Abtastintervalls wird eine Datenzeile im Positionsbereich und im Analogbereich aufgezeichnet.
Das zweite Element im Datenblock „Status“ zeigt in Bit 2 (4 hex) einen eventuellen Datenüberlauf an. Dies ist dann der Fall, wenn nicht alle aufgezeichneten Daten im oben beschriebenen Datenblock eingetragen werden können. Wenn dieses Bit gesetzt ist, muss damit gerechnet werden, dass DMA-Scan-Datensätze verloren gegangen sind, weil die Eingangsfrequenz am Latch-Strobe-Eingang zu hoch war oder das RTS-DMA-Modul vor dem Scanner gestartet wurde.

7.6 Hinweis zur Verwendung des DMA-RTS

Das Aufzeichnen der Daten und das Übertragen der aufgezeichneten Echtzeitdaten in den Scanner benötigt Rechenzeit in der Echtzeit-Task von RWMOS.ELF. Deshalb sollte die Abtastzeit möglichst nicht unter den Standardwert von 1,28 ms gesetzt werden (siehe hierzu auch IHB, Stichwort „SampleTime“).

Die Verwendung des Moduls DMA-RTS erfolgt nach folgendem Schema:

- Initialisierung und Lesezugriff auf die Ressource RTS_DMABLOCK bzw. RTS_DMAANALOGBLOCK
- Initialisierung des Scanners unter Verwendung der Ressource RTS_DMABLOCK bzw. RTS_DMAANALOGBLOCK
- Schreibzugriff auf die Ressource RTS_Init: Dadurch wird der DMA-Kanal initialisiert und aktiviert.
- Scan mit Scannermodul durchführen (wie gewohnt)
- Schreibzugriff auf die Ressource RTS_Stop. Dadurch wird der DMA-Zyklus angehalten.