
POSITIONIER- UND BAHNSTEUERUNG MCU-G3

G3-Ressourcen-Interface

1 Einführung	5
2 Verwendung des G3-Ressourcen-Interface	5
2.1 Initialisierung	5
2.2 Funktionen des G3-Ressourcen-Interface.....	7
3 Die Funktion SyncMode	12
3.1 Einführung	12
3.2 Ressourcen des SyncMode	13
3.3 Hinweise zur Verwendung	13
4 Busmaster Zugriffe auf den Arbeitsspeicher des Host-Systems	15
5 Die GEAR-Funktionalität der MCU-G3 Produkte	16
6 ENDAT-Interface	18
6.1 Einführung	18
6.2 Initialisierung der ENDAT-Schnittstelle	18
6.3 ENDAT Objekte und Funktionen.....	18
6.4 Hinweis zum Zugriff auf das Endat-Interface.....	21
7 DMA Latch mit der MCU-3000.....	22
7.1 Versionshinweise	22
7.2 DMA RTS mit DMA Scan von Analogen Eingängen	22
7.3 Ressourcen für DMA-RTS Handling	23
7.4 Die Ressource RTS_DATABLOCK	24
7.4.1 Das Element Index der Ressource RTS_DATABLOCK	24
7.4.2 Das Element SubIndex der Ressource RTS_DATABLOCK.....	25
7.4.3 Die Handhabung der Ressource RTS_DATABLOCK	25
7.5 Die Ressource RTS_DATAANALOGBLOCK	26
7.6 Hinweis zur Verwendung des DMA-RTS	27

1 Einführung

Das G3-Ressourcen-Interface ermöglicht es, direkt auf interne Systemgrößen der RWMOS-Betriebssystemsoftware 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 File „G3-Universelles-Objekt-Interface.pdf“ 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 G3-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-Objekt-Interface" sind für die Verwendung des G3-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 Resource-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 G3-Universelles-Objekt-Interface.pdf beschrieben.

Hinweis zur PCAP-Programmierung:

- Wenn die Funktion Clear aufgerufen wird, müssen die Handles alle Option-Descriptor-Elemente des RessourcenInterface (BusNumber = 1000) genullt werden.
- Die Funktion Clear darf nicht aufgerufen werden, solange Ressourcenelemente verwendet werden (z.B. bei der Scanner-Funktionalität)
- Der Zugriffstyp 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 Typ 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 G3-Ressourcen-Interface

Tabelle 2: Funktionen G3-Ressourcen-Interface

Dev. Nr.	Bez.	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 Trajectory-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 (Default = -1) Der Wert 0 ist nicht erlaubt	Achsnummer (0, 1, ...)

Tabelle 2: Funktionen G3-Ressourcen-Interface (Fortsetzung)

Dev. Nr.	Bez.	Typ	Erläuterung	Parameter Index [Subindex]
19	LookAhead Deep	integer r/w	Tiefe der LookAhead Berechnung wenn die Funktion AutoSpool in MODEREG gesetzt ist (nur für SAP-Programmierung) (Default = 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, Defaultwert 0)	Achsnummer (0, 1, ...)
28	MCP_MAX	float r/w	Maximalwert des Sollwert-Ausgabeports (bei Servoachsen Analogwert in digits mit Vorzeichen - 10V entspr. 32768 dig.)	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, Defaultwert 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	PosKorrRot Axis	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	Positionsoffset dpoffset (siehe PCAP Kommando wrdppoffset)	Achsnummer (0, 1, ...)
38	mcpmax	double r/w	Maximalwert des Stellgrößenausgangs in Volt (Defaultwert steht in system.dat)	Achsnummer (0, 1, ...)
39	mcpmin	double r/w	Minimalwert des Stellgrößenausgangs in Volt (Defaultwert steht in system.dat)	Achsnummer (0, 1, ...)
40	backlash	double r/w	Hysterese der Getriebespielkompensation in der Achsspezifischen Einheit	Achsnummer (0, 1, ...)
41	mcpcp	double r/w	Positive Nullpunkt-Kompensations-Spannung in Volt (Defaultwert steht in system.dat)	Achsnummer (0, 1, ...)
42	mcpcn	double r/w	Negative Nullpunkt-Kompensations-Spannung in Volt (Defaultwert wie 41)	Achsnummer (0, 1, ...)

Tabelle 2: Funktionen G3-Ressourcen-Interface (Fortsetzung)

Dev. Nr.	Bez.	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 MCU-3000	
63	EpmRev SdiCh1	integer r	Rev.Nr. von U29 auf OPMF-3001	
64	FAST PULSE OUT	integer r/w	Nur bei spezieller Hardwarevariante: durch Zugriff auf diese Ressource kann per Software ein schneller Hardwareausgang (RS422 oder 24V Digital Out) geschaltet werden per PCAP oder SAP-Anweisung. (siehe auch Abschnitt Scan-Trigger-Output im Handbuch „G3-Scanner-Interface.pdf“)	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.
70	TASK STATUS	integer r	Funktionswert der auch mit der PCAP-Funktion gettskinfo() zurückgeliefert wird	Tasknummer (0, 1, 2, 3)
73	Compensation Position	double r	Wirksamer Korrekturwert aller Achs-Kompensationstabellen (ELCAM-Modul) in UserUnit	Achsnummer (0, 1, ...)
79	INV_DIGI	integer r	Invertierungsflag für Digital Eingänge	Achsnummer (0, 1, ...)
83	AOUT03	integer w	nur bei MCU-3100: 4. Analogausgang 16 bit auf Basiskarte schreiben (X2 – Pin 7)	0
87	AOUT	integer w	nur bei MCU-3100: Analogausgang 16 bit auf Basiskarte schreiben (X2 – Pin 1/3/5/7)	Analog-Kanal-Nummer (0, 1, ...) Vorsicht: nicht Achsnummer

Tabelle 2: Funktionen G3-Ressourcen-Interface (Fortsetzung)

Dev. Nr.	Bez.	Typ	Erläuterung	Parameter Index [Subindex]
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 G3-Scanner-Interface.pdf)	Kanal Nr. (0..7)
200	cp[]	double r/w	Controller-Params Column 0 z.B. für GEAR (Kapitel 0)	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)
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 Speicheradressen verwaltet werden.
301	HostMem Byte	byte r/w	8-Bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)
302	HostMem Word	Word r/w	16bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)
304	HostMem Int	integer r/w	32bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)

Tabelle 2: Funktionen G3-Ressourcen-Interface (Fortsetzung)

Dev. Nr.	Bez.	Typ	Erläuterung	Parameter Index [Subindex]
305	HostMem Float	float r/w	32bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access (Gleitpunkt) Basisadresse def. per Device 300	Offset auf Basisadresse in Byte [SetNr] (siehe Dev.Nr. 300)
308	HostMem Double	double r/w	64bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access (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	[Achsnnummer]
311	IsisHost MemByte	byte r/w	8-Bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnnummer]
312	IsisHost MemWord	Word r/w	16bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnnummer]
314	IsisHost MemInt	integer r/w	32bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnnummer]
315	IsisHost MemFloat	float r/w	32bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access (Gleitpunkt) Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnnummer]
318	IsisHost MemDouble	double r/w	64bit-Zugriff auf Host Arbeitsspeicher per Busmaster Access (Gleitpunkt) Basisadresse def. per Device 310	Offset auf Basisadresse in Byte [Achsnnummer]
320	IsisSensor Frequency Factor	integer r/w	Verhältnis der Abtastfrequenz zwischen Isis-Sensor und APCI-8001	[Achsnnummer]
321	IsisPos Norm Factor	double r/w	Normierungsfaktor für die Sollpositionsübergabe an RayDex Systeme Defaultwert Linearachsen: 20000 Defaultwert Rotationsachsen: 4000	[Achsnnummer]
322	IsisIRQ Enable	integer	Interrupt nach Änderung der RayDex Sollpositionswerte ein/aus	
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 Stellgrößenausgang der jeweiligen Achse addiert wird in der Einheit digits	Achsnnummer (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 auf eine andere 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 Werkstückes 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, aber auch ein gespoolter Verfahrzyklus, bestehend aus mehreren Verfahrkommandos sein. Beim Erreichen der Triggerposition, beginnt die Ausführung des Bewegungsprofils der Slave-Achse, wobei dieses auf die Führungsgröße synchronisiert ist. Das heisst, dass bei einer Synchronisierung auf die Istposition bei einem Blockieren der Master-Achse auch die Bewegung der Slave-Achse gestoppt wird.

Mit dem Ende des Verfahrzyklus der Slave-Achse ist auch der Nachführmodus wieder aufgehoben. Danach kann mit der Slave-Achse wieder normal verfahren werden. Nun kann diese z.B. wieder auf die Anfangsposition für den nächsten Zyklus zurückgestellt werden.

Tabelle 3: Initialisierungen für SyncMode

Object-Descriptor Element	Wert
Handle	s.o.
BusNumber	1000
DeviceNumber	7000
Index	jeweilige Funktion lt. Tabelle 4
SubIndex	Parameter zur jeweiligen Funktion lt. 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 = Positions Nachführung 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, ...)
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	Umrechnungsfaktor 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	Umrechnungsfaktor von Counts des Aux-Kanals in Counts / des Nachführkanals Muss bei Nachführung auf aux vom Anwender beschrieben werden (default 1.0) $\text{digits AX} = \text{digits AUX} * \text{AUXFACTOR}$	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 Fahrerrichtung 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 gleich gross sein wie *MasterVelocity*. Um die Synchronität in genau an 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.

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

Bei Nachführung auf die Systemgrösse *aux* (Enkoderposition bei Schrittmotorachsen) muss der Umrechnungsfaktor auf die Userinheit vom Anwender in *AuxFactor* eingetragen werden. Dieser Umrechnungsfaktor 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 $\neq 0$ 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 realisiert ab Version 2.5.3.10.

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

5 Die GEAR-Funktionalität der MCU-G3 Produkte

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 jew. 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 nachfolgende als "interner Vergangenheitswert" bezeichnet und durch die Getriebennachführung 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 (rauhes Lauf). In diesem Fall ist die Vorsteuerung der Slaveachse entsprechend zu verringern.
- Durch Schreiben von -1 auf gcr werden die internen Vergangenheitswerte (s.o.) 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 Masterachsen 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.

Vorsicht: 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 durchführen ohne aktivierte Nachführung bzw. bei den Slave-Achsen, keine Open-Loop Bewegungsvorgänge.
- 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 parametrieren werden. Hier kann leicht, durch falsche Achsauswahl versehentlich eine der Gantry-Achsen angesprochen werden. Bei diesen ggf. durch Wegnahme des Freigabesignals o.ä. 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 MCU-G3) vorgegebenen Taktsignal übertragen. Die Auswahl der Übertragungsart (Positionswerte, Parameter, Diagnose ...) erfolgt mit Mode-Befehlen, welche die Folge-Elektronik (MCU-G3) 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 MCU-G3-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`-Betriebssystemsoftware bzw. nach einem Softwarereset `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 Ressourceninterface der `RWMOS.ELF`-Betriebssystemsoftware und Teile der FPGA-Hardwarelogik 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 G3-Ressourcen-Interface

Dev. Nr.	Bez.	Typ	Erläuterung	Parameter Index [Subindex]
3000	ENDAT_TPV	integer r	ENDAT transmit position value, oder: Meßsystem sende absoluten Positionswert. Die MCU-G3 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. Die MCU-G3 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 die MCU-G3 ein vollständiges Übertragungsprotokoll beginnend mit dem Mode-Befehl Parameter lesen "100011", gefolgt von <u>8 Bit Adresse</u> 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]

Tabelle 5: Fortsetzung ENDAT-Funktionen im G3-Ressourcen-Interface

Dev. Nr.	Bez.	Typ	Erläuterung	Parameter Index [Subindex]
3003	ENDAT_RP	integer w	ENDAT receive parameter, oder: Parameter schreiben. Nach der Speicherbereichsauswahl (siehe ENDAT_SMA) sendet die MCU-G3 ein vollständiges Übertragungsprotokoll beginnend mit dem Mode-Befehl Parameter schreiben „011100“, gefolgt von <u>8 Bit Adresse</u> und <u>16 Bit Parameterwert</u> . Das Messgerät antwortet mit der Wiederholung der Adresse (wird nicht ausgewertet) und Parameterinhalts. Zum Abschluss folgt der CRC-Check.	Achsnummer (0, 1, ...) [Adresse]
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 Zurücksetzen 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 global 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 MCU-3000

Als DMA-Latch wird eine Betriebsart der MCU-3000 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 MCU-3000, 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. In diesem Fall 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	<i>integer w</i>	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	<i>integer w</i>	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	<i>integer r</i>	Diagnoseanzeige in fwsetup ausgeben Rückgabewert BUSY, wenn keine Daten vorliegen (nur für Diagnosezwecke)	
8011	LPR_RTS	<i>short int r</i>	Lesen eines Latch-Registers (16 bit) direkt aus dem Zählerbaustein	Achse [0, 1, ..., 7]
8012	STROBE RTS	<i>short int r</i>	Lesen der Latch Strobes (bitweise codiert) aller Achsen (nur für Diagnosezwecke)	
8013	RTS_DATA BLOCK	<i>datablock r</i>	Scan-Ressource Beschreibung siehe unten	Anzahl der Achsen [0..15] + Achsen bitcodiert [16..31] [Max Anzahl der Datensätze]
8014	RTS_DATA ANALOG BLOCK	<i>datablock r</i>	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 Positionswertes.

Die Besonderheit bei dieser Ressource ist jedoch der Aufbau des aufgezeichneten Datenblockes, der selber 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 fix. Die Anzahl der Spalten dieser Datenstruktur (an) wird im Objekt-Descriptor-Element Index, in den niederwertigen 16bit angegeben. Die Anzahl der Zeilen (zn) wird im Objekt-Descriptor-Element SubIndex angegeben. Der Inhalt von Index und SubIndex ist 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 Abtastintervalles 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 weil 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 bitcodiert die aufzuzeichnenden Achsen 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 in hohem Maße die Größe des Scan-Datensatzes spezifiziert. Um einen unnötigen Speicherbedarf im Scanner-Datensatz und um unnötigen Datentransfer zu vermeiden, sollten diese Werte deshalb 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:

Standardabtastrage = 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 ebenfalls 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 als Datentyp bei der Initialisierung des ObjectDescriptorElementes der Typ 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 Hardwareversion 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 Analog-Eingangsinformationen mit dem Scannermodul aufzuzeichnen wird als Scan-Objekt die Ressource RTS_DATAANALOGBLOCK verwendet. Die Programmierung des Scanners erfolgt analog wie z.B. beim Scannen eines Positionswertes 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 fix. Die Anzahl der Spalten dieser Datenstruktur (an) wird im Objekt-Descriptor-Element Index, in den niederwertigen 16bit angegeben. Die Anzahl der Zeilen (zn) wird im Objekt-Descriptor-Element SubIndex angegeben. Der Inhalt von Index und SubIndex ist 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 Abtastintervalles 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 an Latch-Strobe-Eingang zu hoch war oder weil 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 1,28ms gesetzt werden. Siehe hierzu auch im Handbuch IHB unter dem 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