POSITIONIER- UND BAHNSTEUERUNG MCU-G3

PROGRAMMIER- UND REFERENZ-HANDBUCH / PHB

Stand: 16.01.2024, ab Disk V2.53VP

1 Einführung

Wozu dient dieses Handbuch?

Dieses Handbuch enthält alle notwendigen Angaben zur Programmierung der **MCU-G3** Positionier- und Bahnsteuerungen. Das komplette Handbuch besteht aus drei Teilen: BHB (Bedienungs-Handbuch), IHB (Inbetriebnahme-Handbuch) und PHB (Programmierhandbuch).

Welche Geräte gehören zur MCU-G3-Familie?

Bei der MCU-G3-Familie handelt sich um Positionier- und Bahnsteuerungen der dritten Generation. Hierzu gehören z.Zt. die Positionier- und Bahnsteuerungen MCU-3000, MCU-3100, MCU-3100e, MCU3100e-EC, MCU-6000 und MCU-3400C. Weitere Geräte sind in Planung.

Weitere Anmerkungen

Sofern die in diesem Handbuch beschriebenen Funktionen nicht für alle Geräte der MCU-G3-Familie übereinstimmen, sind diese besonders gekennzeichnet. In diesem Fall gilt die entsprechende Funktion nur für das jeweils gekennzeichnete Gerät! Bevor die verschiedenen Programmiermethoden und Betriebsarten vorgestellt werden können, ist es notwendig verschiedene Funktionen der *rw_MOS*-Betriebssystem-Software zu beschreiben. Weitere Informationen zu *rw_MOS* sind im [BHB / Kapitel 4.1] enthalten.

2 Interne Details der *rw_MOS*-Betriebssystem-Software

Wie schon im Bedienungs-Handbuch erwähnt, beruht die Leistungsfähigkeit der MCU-G3 unter anderem auf der Betriebssystemsoftware *rw_MOS*. In den nachfolgenden Kapiteln werden die in *rw_MOS* implementierten Funktionen wie z.B. die Profilgenerierung oder Endschalterbehandlung beschrieben.

2.1 Der MCU-G3 Lageregler

Die Grundbetriebsart der MCU-G3 ist die Lageregelung. In dieser Betriebsart versucht die MCU-G3 die Motorposition auf der Sollposition festzuhalten. Der Regelkreis besteht gewöhnlich aus den Komponenten Digitaler Regler - Digital/Analog Wandlung - Leistungsteil - Motor - Enkoder - Impulserfassung. Der Enkoder ist in den meisten Fällen direkt am Motor angebaut, d.h. starr mit der Motorachse verbunden. Falls dies nicht der Fall ist, sind die Übertragungsglieder zwischen Motorachse und Enkoderachse zusätzlich im Regelkreis enthalten. An der Motorachse ist weiterhin die Last angeschlossen. Das Verhalten der Regelung wird durch alle im Regelkreis enthaltenen Glieder und die Last bestimmt. Bei einem gegebenen System lässt sich das Regelverhalten nur durch die des digitalen Filters beeinflussen. Hierbei müssen alle möglichen Betriebsfälle, z.B. Änderungen der Last berücksichtigt werden.

2.1.1 Regelkreis geöffnet / geschlossen

Nach dem Einschalten ist der Regelkreis zunächst geöffnet. Auf den Stellgrößenausgang (Motor-Command-Port) wird der Wert 0 ausgegeben. Durch Ausgabe eines Wertes kann die angeschlossene Achse ungeregelt verfahren werden. Mit dem PCAP-Befehl *cl()* (close loop) wird der Regelkreis geschlossen. Hierbei wird die aktuelle Position als Sollposition übernommen, um ein unbeabsichtigtes Verfahren der Motorachse zu verhindern. Nur nach Aktivierung der Lageregelung können Verfahrprofile durchgeführt werden. Dies gilt auch für Schritt-Motoren.

2.1.2 PIDF-Filter

Das digitale Filter hat die Struktur eines realen PIDF-Filters. Mit diesem Reglertyp lassen sich nahezu alle in der Praxis auftretenden Regelstrecken stabil einstellen.

2.1.2.1 Die Filterparameter K_D, K_I, K_P

Die Einstellung erfolgt über die Filterparameter K_D, K_I und K_P. Die Bedeutung dieser Parameter lassen sich sehr einfach auf die in der Literatur gängigen Parameter Proportionalverstärkung K_P, Vorhaltezeit T_V (Differenzierzeit) und Nachstellzeit T_N (Integrationszeitkonstante) zurückführen.

- K_P - Proportionalverstärkung
- K_I - Integrierbeiwert
- K_D - Differenzierbeiwert
- T_V - Vorhaltezeit
- T_N - Nachstellzeit

$K_I = K_P / T_N$
$K_D = K_P * T_V$

Falls ein Regler mit anderer Struktur realisiert werden soll, so lassen sich die einzelnen Anteile einfach durch Nullsetzen disaktivieren.

2.1.2.2 Zusätzliches Phasenglied

Das gegebene digitale PIDF-Filter ist standardmäßig in Kette geschaltet mit einem Verzögerungsglied erster Ordnung mit der Zeitkonstante $T_{A/2}$ (halbe Abtastzeit). Daher kommt die Bezeichnung reales PIDF-Filter. Mit dem Filterparameter K_{PL} kann nun diese Verzögerungszeit noch verkleinert werden. Dadurch ist eine härtere Einstellung des Regler möglich. Der Parameter K_{PL} darf prinzipiell Werte zwischen 0 und 1 annehmen. In der Praxis ist jedoch ein Wert größer als ca. 0.95 nicht mehr sinnvoll.
Der Zusammenhang zwischen K_{PL} und der Verzögerungszeit läßt sich einfach darstellen:

K_{PL} - Filterparameter
T_{VERZ} - reale Verzögerungszeit des PIDF-Filters
T_A - Abtastzeit

$$T_{VERZ} = (1 - K_{PL}) * T_A / 2$$

2.1.2.3 Abtastzeit

Im obigen Abschnitt wurde die Abtastzeit T_A verwendet. Diese ist eine charakteristische Größe für den digitalen Regler. Die Abtastzeit ist die Zeit, nach der jeweils Soll- und Istwert abgetastet werden und über den Regelalgorithmus die Stellgröße berechnet wird. Wenn die Abtastzeit klein ist gegenüber den vorkommenden Streckenzeitkonstanten kann die Dimensionierung des Reglers wie bei einem kontinuierlichen Regler erfolgen. Dadurch sind zur Einstellung keine speziellen Kenntnisse der digitalen Regelungstechnik erforderlich.
Anmerkung: In der MCU-G3-Standardversion ist die Abtastzeit auf **1,28ms** eingestellt.

2.2 Der MCU-G3 Profilgenerator

Beim Verfahren mit den einzelnen Achsen werden die angegebenen Wege mit einem Trapez-Drehzahl-Profil angefahren. Für ein derartiges Trapez-Drehzahl-Profil sind die Größen Anfangsgeschwindigkeit, Anfangsposition, Beschleunigung, Maximalgeschwindigkeit, Zielposition und Zielgeschwindigkeit maßgebend. Die vorliegende Profilgenerierung erzeugt für den Lageregler [Kapitel 2.1] abtastsynchron die entsprechenden Sollwerte, damit von der augenblicklichen Position ausgehend von der Momentangeschwindigkeit zur Maximalgeschwindigkeit beschleunigt wird. Die Anfangsgeschwindigkeit und Anfangsposition sind Momentanwerte und werden nicht als Parameter für ein Bewegungsprofil angegeben. Bevor die Zielposition erreicht wird, bremst der Profilgenerator rechtzeitig mit der vorgegebenen Beschleunigung ab, damit im vorgegebenen Zielpunkt die Zielgeschwindigkeit erreicht wird.

2.2.1 Profilgenerierung für JOG-Befehle

Beim Abfahren eines Trapez-Drehzahlprofils mit JOG-Verfahrbefehlen (Einzelachsbewegungen) sind nun einige Sonderfälle möglich:

- Die Anfangsgeschwindigkeit ist negativ gegenüber der Verfahrrichtung. Die Achse verfährt also zunächst in die falsche Richtung, bremst jedoch ab, kehrt um und beschleunigt nun in die richtige Richtung.
- Die Endgeschwindigkeit ist negativ gegenüber der Verfahrrichtung. Die Achse fährt zunächst über den Zielpunkt hinaus, bremst ab, kehrt die Richtung um und hat beim nochmaligen Erreichen des Zielpunktes die Zielgeschwindigkeit.
- Die Anfangsgeschwindigkeit ist gleich der Maximalgeschwindigkeit.
- Die Anfangsgeschwindigkeit ist höher als die Maximalgeschwindigkeit. In diesem Fall wird automatisch auf die Maximalgeschwindigkeit abgebremst.
- Die Endgeschwindigkeit ist gleich der Maximalgeschwindigkeit.
- Die Maximalgeschwindigkeit wird nicht erreicht, da vorher abgebremst werden muss um die Zielgeschwindigkeit bis zur Zielposition zu erreichen. In diesem Fall wird ein Dreieck-Drehzahl-Profil abgefahren.

Alle diese Fälle werden richtig behandelt, falls der abzufahrende Weg jeweils ausreicht. Weiterhin muss stets eine positive Maximalgeschwindigkeit und Beschleunigung angegeben werden und die Endgeschwindigkeit muss kleiner oder gleich groß sein wie die Maximalgeschwindigkeit. Bei Angabe einer negativen Beschleunigung wird diese für die Bremsrampe des Profils herangezogen. Bei JOG-Verfahrbefehlen ist es also möglich Beschleunigungs- und Bremsrampen mit unterschiedlicher Steilheit zu programmieren.

Bei nachfolgend aufgeführten Fällen treten Geschwindigkeitssprünge, also unerwünscht hohe Beschleunigungen auf. Falls diese vom System nicht realisiert werden können, tritt ein Schleppfehler auf, der jedoch i.a. nach einer begrenzten Zeit wieder ausgeregelt wird. Beim Einsatz von Schrittmotoren können diese Fälle i.a. nicht zugelassen werden.

- Der angegebene Verfahrweg reicht nicht zum Abbremsen aus.
- Die Zielgeschwindigkeit ist höher als die Maximalgeschwindigkeit. In diesem Fall wird am Profilende die Verfahrgeschwindigkeit auf die Zielgeschwindigkeit gesetzt.

2.2.2 Profilgenerierung für MOVE-Befehle

Beim Abfahren eines Trapez-Drehzahlprofils mit MOVE-Verfahrbefehlen (Mehrachsbewegungen mit Interpolation) mit einer oder mehreren Achsen sind folgende Sonderfälle möglich:

- Die Endgeschwindigkeit ist negativ gegenüber der Verfahrrichtung. Das System fährt zunächst über den Zielpunkt hinaus, bremst ab, kehrt die Richtung um und hat beim nochmaligen Erreichen des Zielpunktes die Zielgeschwindigkeit.
- Die Anfangsgeschwindigkeit ist gleich der Maximalgeschwindigkeit.
- Die Anfangsgeschwindigkeit ist höher als die Maximalgeschwindigkeit. Bei direkten MOVE-Befehlen wird in diesem Fall automatisch auf die Maximalgeschwindigkeit abgebremst. Bei Spooler-Befehlen wird die Anfangsgeschwindigkeit auf die Maximalgeschwindigkeit gesetzt. Dies entspricht einem Geschwindigkeitssprung.
- Die Endgeschwindigkeit ist gleich der Maximalgeschwindigkeit.
- Die Maximalgeschwindigkeit wird nicht erreicht, da vorher abgebremst werden muss um die Zielgeschwindigkeit bis zur Zielposition zu erreichen. In diesem Fall wird ein Dreieck-Drehzahl-Profil abgefahren.

Alle diese Fälle werden richtig behandelt, falls der abzufahrende Weg jeweils ausreicht. Weiterhin muss stets eine positive Maximalgeschwindigkeit und Beschleunigung angegeben werden und die Endgeschwindigkeit muss kleiner oder gleich groß sein wie die Maximalgeschwindigkeit. Bei Angabe einer negativen Beschleunigung oder Maximalgeschwindigkeit wird das Profil verworfen. Bei MOVE-Verfahrbefehlen ist es nicht möglich Beschleunigungs- und Bremsrampen mit unterschiedlicher Steilheit in einem Verfahrbefehl zu programmieren. Falls dies erforderlich ist können jedoch mehrere MOVE-Verfahrbefehle hintereinander programmiert werden.

Bei nachfolgend aufgeführten Fällen treten Geschwindigkeitssprünge, also unerwünscht hohe Beschleunigungen auf. Falls diese vom System nicht realisiert werden können, tritt ein Schleppfehler auf, der jedoch i.a. nach einer begrenzten Zeit wieder ausgeregelt wird. Im Zusammenhang mit Schrittmotoren dürfen diese Fälle i.a. nicht zugelassen werden.

- Der angegebene Verfahrweg reicht nicht zum Beschleunigen auf die Zielgeschwindigkeit aus. In diesem Fall wird die Zielgeschwindigkeit auf einen Wert gesetzt der im angegebenen Profil erreicht werden kann. In diesem Fall tritt jedoch kein Geschwindigkeitssprung auf.
- Der angegebene Verfahrweg reicht nicht zum Abbremsen aus. In diesem Fall wird die Profil-Anfangsgeschwindigkeit auf einen Wert gesetzt der es erlaubt im angegebenen Profil auf die Endgeschwindigkeit abzubremsen.
- Die Zielgeschwindigkeit ist höher als die Maximalgeschwindigkeit. In diesem Fall wird am Profilende die Verfahrgeschwindigkeit auf die Zielgeschwindigkeit gesetzt.
- Die Richtung des Verfahrprofils wird geändert. In diesem Fall wird der Betrag des Geschwindigkeitsvektors aus der bisherigen Richtung in die nun abzufahrende Richtung gelegt. In diesem Fall treten bei den beteiligten Achsen mehr oder weniger große Geschwindigkeitssprünge auf. Besondere Vorsicht ist hier beim Einsatz von Schrittmotorsystemen geboten.

Diese Art der Profilgenerierung wird nicht nur beim Abfahren von linearen Bewegungskommandos ausgeführt. Auch beim Abfahren von Kreisbewegungen mit zwei Achsen wird die Bahngeschwindigkeit nach diesem Schema generiert.

2.2.3 Beschleunigung

Falls eine Beschleunigung kleiner als Null angegeben wird, wird bei MOVE-Befehlen der Datensatz verworfen. Bei JOG-Befehlen gibt eine negative Beschleunigung die Steilheit der Bremsrampe vor. Standardmäßig hat die Bremsrampe die gleiche Steilheit wie die Beschleunigungsrampe. Die Einheiten für die Beschleunigung können im Hilfsprogramm *mcfg.exe* achsspezifisch festgelegt werden. Für die Interpolationsbefehle (MOVE-Befehle) gibt es verschiedene Möglichkeiten zur Auswahl der Einheiten. Die Wertangabe für die Beschleunigung erfolgt als Gleitpunktzahl, der Wertebereich ist also nahezu unbegrenzt. Bei Angabe einer höheren Beschleunigung als vom System realisiert werden kann, entsteht während der Beschleunigungsphase ein vergrößerter Schleppfehler.

2.2.4 Maximal-Geschwindigkeit

Die Maximalgeschwindigkeit muss immer größer als Null angegeben werden, andernfalls wird der Datensatz verworfen (MOVE) bzw. wird ein Endlosprofil in die falsche Richtung abgefahren (JOG). Die Einheiten für die Maximalgeschwindigkeit können im Hilfsprogramm *mcfg.exe* achsspezifisch festgelegt werden. Für die Interpolationsbefehle gibt es verschiedene Möglichkeiten zur Auswahl der Einheiten. Die Wertangabe für die Maximalgeschwindigkeit erfolgt als Gleitpunktzahl, der Wertebereich ist also nahezu unbegrenzt. Bei Angabe einer höheren Geschwindigkeit als vom System realisiert werden kann, entsteht während des Verfahrens ein vergrößerter Schleppfehler. Falls die angegebene Maximalgeschwindigkeit kleiner ist als die Anfangsgeschwindigkeit, gelten die oben genannten Bedingungen abhängig von der Befehlsart.

2.2.5 Zielgeschwindigkeit

Die Zielgeschwindigkeit kann positiv, negativ oder natürlich mit 0 angegeben werden. Die Richtung der Zielgeschwindigkeit bezieht sich immer auf die Verfahrrichtung. Beim Verfahren in negativer Richtung und positiver Zielgeschwindigkeit wird also in negativer Richtung weitergefahren. Die Zielgeschwindigkeit hat dieselbe Einheit wie die Maximalgeschwindigkeit. Die Wertangabe erfolgt als Gleitpunktzahl, der Wertebereich ist also nahezu unbegrenzt. Bei Angabe einer höheren Geschwindigkeit als vom System realisiert werden kann, entsteht während des Verfahrens ein vergrößerter Schleppfehler. Falls die angegebene Zielgeschwindigkeit größer ist als die Maximalgeschwindigkeit, wird das Verfahrprofil mit einem Geschwindigkeitssprung beendet. Die Momentangeschwindigkeit wird in diesem Fall am Profilende auf die Zielgeschwindigkeit gesetzt.

2.2.6 Geschwindigkeitskorrektur

In bestimmten Fällen ist es erwünscht, die Achs- oder Bahngeschwindigkeit während der Ausführung eines Trapez- Drehzahlprofils zu verändern. Ein typisches Beispiel hierfür ist die manuelle Geschwindigkeitskorrektur (Override). Hierzu stehen dem Anwender verschiedene SAP- und PCAP-Befehle zur Verfügung.
Der Geschwindigkeitskorrekturfaktor, dessen Defaultwert = 1.0 ist, wirkt sich gleichermaßen auf Geschwindigkeiten und Beschleunigungen aus.
Je nach Betriebsart muss bei der Programmierung des Overrides streng unterschieden werden, wie dieser zu verwenden ist: Bei Einachs-Verfahrbefehlen (JOG-Befehle) kann der JOG-Override für jede Achse getrennt programmiert werden. Dies darf jedoch nicht bei Interpolationsfahrten gemacht werden. Bei Interpolationsbefehlen (MOV-Befehle) muss der Trajektorie-Override gesetzt und mit dem Kommanod utrovr synchron für alle Achsen übernommen werden, die an der Interpolationsfahrt beteiligt sind. Nur so ist die Synchronität der interpolierenden Achsen jederzeit gewährleistet. Bei der Übernahme des Trajektorie-Override, wird dieser Wert automatisch bei den beteiligten Achsen auch als JOG-Override übernommen (abschaltbar). Um Geschwindigkeitssprünge bei der Programmierung des Override zu verhindern, kann für den Trajektorie-Override eine Anpasszeit programmiert werden.

2.2.7 Zielposition / Verfahrweg

Die Angabe des Zieles kann als Relativ- oder Absolutwert erfolgen. Bei Angabe eines Relativwertes wird um den angegebenen Weg verfahren, d.h. es wird ein Verfahrweg programmiert. Bei Angabe eines Absolutwertes wird auf die angegebene Position verfahren, d.h. es wird eine Zielposition programmiert. Der Bezugspunkt für absolute Zielpositionen ist der Maschinennullpunkt.

2.2.8 Betriebsarten zur Befehlsabarbeitung

Verfahrbefehle und sonstige Befehle können in zwei unterschiedlichen Betriebsarten, dem sogenannten Direkten-Modus und dem Spool-Modus ausgeführt werden. Die zur Ausführung kommende Betriebsart wird durch die Syntax des jeweiligen Befehls automatisch festgelegt.

Anmerkung: Die Kommandokürzel der *spool*-Befehle sind im Gegensatz zu den Direkt-Befehlen durch das Zeichen 's' als ersten Buchstaben im Befehlswort gekennzeichnet. Es stehen für beide Programmiermethoden also SAP- und PCAP-Programmierung identische *spool*-Befehle zur Verfügung.

2.2.8.1 Direkter Modus

Der direkte Modus wird durch den Aufruf von speziellen *move*- und *jog*-Befehlen automatisch aktiviert.
Beim Programmieren eines Verfahrbefehls im direkten Modus, wird mit der Ausführung des angegebenen Befehls nach einer systembedingten Verzögerungszeit (ca. 2 - 3 Abtastintervalle) begonnen. Ein bereits ablaufendes Profil wird nicht bis zum Ende abgefahren, die Momentanwerte von Geschwindigkeit und Position werden als Anfangswerte für den aktuellen Verfahrbefehl übernommen. Falls die Profildaten und die Anfangswerte konsistent sind, d.h. den obigen Anforderungen entsprechen, wird ein gerade ablaufendes Profil nahtlos fortgesetzt. So ist es z.B. möglich den Zielpunkt eines ablaufenden Profils zu verändern, die Geschwindigkeit nochmals zu erhöhen, oder die Beschleunigung der Bremsrampe nachträglich zu verändern, ja sogar die Beschleunigung während einer Beschleunigungsrampe zu verändern. Falls verschiedene Profile nacheinander abgefahren werden sollen, muss jeweils das Profilende abgewartet werden.

Anmerkung: Eventuell im Spooler vorliegende Daten werden bei der Ausführung von Befehlen im direkten Modus verworfen.

2.2.8.2 Spool-Modus

Im Spool-Modus kann eine große Anzahl von Verfahr- oder sonstigen Befehlen in eine Warteschlange (Spooler) eingetragen werden. Jede Achse hat ihren eigenen Spooler. Bei Interpolationsbefehlen werden die entsprechenden Spooler synchron geladen und abgearbeitet. Die Abarbeitung der Befehle, welche im Spooler eingetragen sind, wird beispielsweise durch den PCAP-Befehl *ssms()* gestartet. Während der Abarbeitung ist es möglich, den Spooler mit weiteren Befehlen zu beschreiben. Die Abarbeitung der Befehle aus dem Spooler erfolgt nacheinander ohne Verzögerung. Der freie Spoolerbereich wird mit jedem Befehlseintrag kleiner, beim Beginn jeder Befehlsausführung wieder größer. Wenn alle Befehle im Spooler abgearbeitet sind, wird automatisch wieder in den Direkt-Modus geschaltet, d.h. nach erneutem Eintragen von *spool*-Befehlen muss deren Abarbeitung erneut gestartet werden.

Anmerkung: Damit die Spoolereinträge richtig abgearbeitet werden können, müssen folgende Voraussetzungen gelten:

- Alle Achsen, für die Kommandos gespoolt werden sollen, müssen beim ersten Spoolereintrag in der Lageregelung sein.
- Die Geschwindigkeit dieser Achsen muss vor der Ausführung des ersten *spool*-Befehls Null sein, deshalb darf der Befehl Start Spooled Motions Synchronised *ssms()* nur dann ausgeführt werden, wenn alle beteiligten Achsen stehen.
- Verfahrprofile im Spooler müssen eine Ausführungszeit haben. die größer ist als die Abtastzeit der Steuerung (Default 1,28ms). Die Ausführungszeit eines Verfahrprofils ergibt sich ca. aus Weg / Geschwindigkeit. Kürzerer Verfahrprofile müssen vom Anwenderprogramm unterdrückt werden.

2.2.8.3 Weitere Hinweise zum Spoolerbetrieb

Damit eine, per Spoolerbefehlen programmierte Kontur bahntreu abgefahren wird, müssen innerhalb einer Befehlssequenz immer alle Achsen synchron programmiert und gestartet werden. Desweiteren muss ein etwaiger Override-Wert immer synchron für alle Interpolationsachsen übernommen werden (Kommando utrovr). Sobald durch eine asynchrone Operation der Spoolerablauf gestört wird, muß damit gerechnet werden, dass die programmierte Kontur nicht eingehalten wird. Um einen derartigen Fehler zu erkennen kann die automatische Spooler-Synchronisations-Überwachung verwendet werden. Diese ist ab RWMOS V2.5.3.88 verfügbar.
Wenn ein asyncroner Ablauf der Spooler vom Betriebssystem erkannt wird, wird das Bit SAF (#19) gesetzt. Falls im MODEREG-Register das Bit JSatSAF (#28) gesetzt ist, werden in diesem Fall alle Achsen per Jog-Stop mit der programmierten Stop-Deceleration angehalten. Dieses Fehlerereignis wird mit dem entsprechenden saf Flag in axst und mit dem Bit 19 im ErrorReg (siehe auch 4.4.71.1) angezeigt.
Durch Aufruf eines Jog- oder eines direkten Move Befehles werden eventuell gesetzte SAF-Flags wieder gelöscht.

2.3 Interpolation mit der MCU-G3

Das Verfahren einzelner Achsen mit der MCU-G3 erfolgt mit den *jog*-Befehlen. Um mehrere Achsen interpoliert zu verfahren, stehen die *move*-Befehle zur Verfügung. Mit der MCU-G3 ist es möglich Kreis-, Linear- und Helixinterpolationen durchzuführen. Mehrere Interpolationsprofile können gleichzeitig, mit beliebiger Anfangs- und Endgeschwindigkeit abgearbeitet werden. Alle Interpolationsberechnungen erfolgen abtastsynchron (1.28 ms).

2.3.1 Linearinterpolation

Bei der Linearinterpolation wird eine beliebige Anzahl von Achsen auf einer Raumgeraden (n-dimensional) vom Startpunkt zum Zielpunkt (Absolutpositionierung) oder um einen Raumvektor (Relativpositionierung) verfahren. Parameter bei der Linearinterpolation sind die teilnehmenden Achsen, der Verfahrweg bzw. die Zielposition, die Bahnbeschleunigung, die maximale Bahngeschwindigkeit und die Bahn-Zielgeschwindigkeit. Beim Interpolieren mit einer Anfangsgeschwindigkeit sollte gewährleistet sein, dass die Richtungsvektoren der Anfangsgeschwindigkeit und des Interpolationsprofils übereinstimmen. Ansonsten wird die Richtung des Geschwindigkeitsvektors geändert. Dies kann zu Geschwindigkeitssprüngen bei den teilnehmenden Achsen führen. Falls die Interpolationsrichtung von einem Profil zum nächsten geändert werden muss, sollte ein Zwischenstopp erfolgen. Bei Richtungsumkehr ist die Beendigung des ersten Profils mit negativer Zielgeschwindigkeit möglich.

2.3.1.1 Formale Linearinterpolation

Beim Abfahren von Konturen besteht die Möglichkeit, dass eine Achse die momentane Motorposition beibehalten muss, während die anderen Achsen interpoliert verfahren werden. Diese stillstehende Achse kann jedoch formal an dieser Interpolation der anderen Achsen teilnehmen und bleibt somit auf diese synchronisiert. Diese formale Interpolation ist besonders wichtig in der Spool-Betriebsart und wird automatisch für alle Achsen selektiert, bei denen ein Verfahrweg von 0 programmiert wird.

2.3.2 Kreisinterpolation

Die Kreisinterpolation wird mit zwei beliebigen Achsen durchgeführt. Parameter bei der Kreisinterpolation sind die teilnehmenden Achsen, die Koordinaten des Kreismittelpunktes, der Verfahrwinkel (positiv oder negativ), die Bahnbeschleunigung, die Bahn-Maximalgeschwindigkeit und die Bahn-Zielgeschwindigkeit. Die Koordinaten des Kreismittelpunktes können in Absolutkoordinaten oder in Relativkoordinaten angegeben werden.
Beim Interpolieren eines Kreises mit einer Anfangsgeschwindigkeit muss darauf geachtet werden, dass die Anfangsgeschwindigkeit die gewünschte Richtung hat, d.h. die Richtung der Tangente im Kreisstartpunkt. Ansonsten wird die Richtung des Geschwindigkeitsvektors geändert. Dies kann zu Geschwindigkeitssprüngen bei den teilnehmenden Achsen führen. Falls die Interpolationsrichtung von einem Profil zum nächsten geändert werden muss, sollte ein Zwischenstopp erfolgen. Bei Richtungsumkehr ist die Beendigung des ersten Profils mit negativer Zielgeschwindigkeit möglich.

2.3.3 Helixinterpolation

Die Helix-Interpolation wird für zwei beliebige Achsen als Zirkular-Interpolation und mit einer dritten beliebigen Achse als Linear-Interpolation ausgeführt.

2.3.4 Mantelflächenbearbeitung

Bei Interpolationsbefehlen werden die Bahnparameter Geschwindigkeit und Beschleunigung mit Hilfe der Systemparameter PositionUnit (PU) und TimeUnit (TU) angegeben. Dabei ist vorausgesetzt, dass an einer Interpolationsfahrt immer Achsen des gleichen Typs (translatorisch oder rotatorisch) teilnehmen, da nur so

gewährleistet ist, dass die PositionUnit entsprechend verarbeitet werden kann. Wenn nun rotatorische Achsen an einer translatorischen Interpolationsfahrt teilnehmen, wie z.B. bei der Bearbeitung einer Zylinderoberfläche, muss für diese der wirksame Radius definiert werden, über den die Umrechnung der rotatorischen Achsgrößen in die translatorischen Interpolationsgrößen stattfindet.
Hierzu steht der achsspezifische Wert *effradius* zur Verfügung. Somit ist es möglich, die richtige Bahngeschwindigkeit und Beschleunigung einzustellen. Der Radius wird in der bei der Linearinterpolation eingestellten Einheit angegeben. Diese Möglichkeit besteht bei Linear-, Kreis- und Helixinterpolation.

Beispiel:

Auf einer Rohroberfläche soll geschweißt werden. Das Rohr hat einen Durchmesser von 200mm und wird mit einer rotatorisch definierten Achse C gedreht.

```
PU := 0;                // Position Unit = mm
C.effradius := 100;     // Radius angeben
```

Nun kann die Achse C in der Linearinterpolation verwendet werden:

```
mlr (X := 25, C := 60);
```

Der Verfahrweg der rotatorischen Achse wird hier in der translatorischen Positionseinheit (hier mm) angegeben. Falls der Verfahrweg der rotatorischen Achse in der achsspezifischen rotatorischen Einheit (z.B. deg) angegeben werden soll, muss das Bit 10 im Register MODEREG (Kapitel 6.3.1.5) gesetzt werden. Diese Umschaltung ist nur für alle Achsen gleichzeitig möglich.

2.3.5 Synchrone und asynchrone Interpolationen

Die MCU-G3 gestattet unter anderem auch das Abarbeiten mehrerer verschiedener Interpolationen zum gleichen Zeitpunkt. So ist es beispielsweise möglich, zwei Zirkular-Interpolationen mit jeweils zwei verschiedenen Achskanälen auszuführen. Die jeweiligen Interpolationen können dabei synchron als auch asynchron zueinander ausgeführt werden. Die synchrone Betriebsart wird dabei besonders gut durch den Spoolermechanismus unterstützt. Selbstverständlich kann auch neben einer Interpolation jede beliebige andere Achse, die nicht im Interpolationszusammenhang verwendet wird, unabhängig verfahren werden.

2.4 Die MCU-G3 Endschalterbehandlung

Die MCU-G3 bietet eine große Palette von Möglichkeiten der Endschalterbehandlung bzw. Verfahrbereichsbegrenzung. So ist es möglich einen oder mehrere beliebige Digitaleingänge als Hardwareendschalter Links oder Rechts zu konfigurieren. Bei der Konfiguration wird dem Endschaltereingang zusätzlich eine Funktion TOM, SMA oder SMD zugeordnet. Weiterhin kann für jeden Achskanal zusätzlich ein Softwareendschalter links und rechts definiert werden. Die Endschalterpositionen sind frei wählbar. Auch hier kann zwischen den Funktionen TOM, SMA und SMD ausgewählt werden. Der Zustand der Endschalter kann dem Statusflag *axst* entnommen werden.
Ein bestimmter Endschalterzustand wird beim Unterschreiten der Sollposition unter die Endschalterposition gelöscht.
Anmerkung: Alle Endschalterzustände werden beim Schließen des Regelkreises [Kapitel 4.4.6 - *cl()*] gelöscht.

2.4.1 Endschalterfunktion TOM (Turn-Off-Motor)

Bei dieser Endschalterfunktion wird der Motor in die Endschalterrichtung stromlos geschaltet, d.h. die Achse läuft beim Ansprechen des Endschalters ungeregelt aus und kann nicht weiter in den Endschalterbereich, lediglich gegen die Endschalterrichtung verfahren werden. Die Sollposition kann jedoch, z.B. durch ein

gerade ablaufendes Profil weiterhin in den Endschalterbereich laufen. Beim Herausfahren aus dem Endschalterbereich sind unkontrollierte Geschwindigkeitssprünge möglich.

2.4.2 Endschalterfunktion SMA (Stop-Motor-Abruptly)

Bei dieser Endschalterfunktion wird die Sollposition beim Überschreiten der Endschalterposition festgehalten. Der Lageregler hält die Achse in dieser Position fest. Die vom Profilgenerator berechnete Sollposition wird jedoch intern richtig weitergeführt. Beim Ein- und Austritt der Sollposition aus dem Endschalterbereich sind unkontrollierte Geschwindigkeitssprünge möglich.

2.4.3 Endschalterfunktion SMD (Stop-Motor-Decelerate)

Bei dieser Endschalterfunktion wird die betreffende Achse mit der spezifizierten Stop Deceleration {*sdec*} auf Geschwindigkeit 0 abgebremst. Die Achse wird in den direkten Modus geschaltet und etwaige Spoolereinträge werden verworfen. Ein weiteres geregeltes Verfahren in den Endschalterbereich ist nicht mehr möglich. Die Achse kann mit allen Verfahrbefehlen aus dem Endschalterbereich herausgefahren werden. Dies ist die Universal-Endschalterfunktion.
Sonstige Funktionsgruppen

2.4.4 Applikationsspezifische Systemvariable

In der RWMOS Systemsoftware können systemspezifische Variable verfügbar sein. Diese Variable werden anforderungsspezisch angepasst und dokumentiert und können in beliebiger Anzahl verfügbar sein. Für derartige Variable gibt es Zugriffsmechanismen für Ganz- und Gleitpunktwerte:

In der PCAP-Programmierumgebung gibt es hierfür die Lese-Funktionen rdSysVarInt, rdSysVarDbl und die Schreibfunktionen wrSysVarInt und wrSysVarDbl.
In der Standalone-Programmierumgebung erfolgt der Zugriff indiziert auf die Systemvariable SysVarInt[] oder SysVarDbl[].

```
CI0 := SysVarInt[1];
```

2.4.5 Applikationsspezifische Achsvariable

In der RWMOS Systemsoftware können achsspezifische Variable verfügbar sein. Diese Variable werden anforderungsspezisch angepasst und dokumentiert und können in beliebiger Anzahl verfügbar sein. Für derartige Variable gibt es Zugriffsmechanismen für Ganz- und Gleitpunktwerte:

In der PCAP-Programmierumgebung gibt es hierfür die Lese-Funktionen rdAxVarInt, rdAxVarDbl und die Schreibfunktionen wrAxVarInt und wrAxVarDbl.
In der Standalone-Programmierumgebung erfolgt der Zugriff indiziert mit Hilfe der Achsenqualifizierer varint[] oder vardbl[].

```
CI0 := A3.varint[1];
```

3 Die MCU-G3 Programmiermethoden

Ein wichtiger Bestandteil der MCU-G3-Positionier- und Bahn-Steuerung ist das Echtzeit-Multi-Task-Betriebssytem *rw_MOS* (*M*ips *O*perating *S*ystem). Dieses ist in der Datei *rwmos.elf* abgelegt und wird einmalig pro PC-Systemstart mit dem Bootmenü in *mcfg.exe* oder durch ein Anwenderprogramm innerhalb weniger Sekunden in den lokalen Arbeitsspeicher der MCU-G3 geladen. Die Betriebssystemsoftware *rw_MOS* ist in verschiedene Tasks aufgeteilt, welche im wesentlichen zwei unterschiedliche Arten der Anwenderprogrammierung ermöglicht.

Anmerkung: *rwmos.elf* und *mcfg.exe* sind Bestandteil der MCU-G3 TOOLSET Software. Weitere Informationen dazu sind im Bedienungshandbuch enthalten.

3.1 PC-Applikations-Programmierung (PCAP-Programming, auch Direkt-Programmierung)

Die MCU-G3 PC-Applikations-Programmierung (PCAP) wird durch ein auf dem PC ablaufendes Anwenderprogramm erledigt. Die Programmerstellung erfolgt mit Hilfe einer höheren Programmiersprache wie z.B. *Borland C*, *Microsoft C, Borland Delphi* oder *Microsoft Visual Basic*. Mit Hilfe der im Lieferumfang enthaltenen Funktionenbibliotheken für diese Programmiersprachen kann der Anwender auf einen leistungsfähigen Befehlsvorrat zurückgreifen, welcher eine schnelle und effektive Programmerstellung gestattet. Zum Befehlsumfang gehören beispielsweise Verfahrbefehle mit und ohne Interpolation, Ein-Ausgabe-Befehle, Abfrage-Befehle, *spool*-Befehle usw.
Das typische Anwenderprogramm sendet einen oder mehrere dieser Befehle an die MCU-G3 und wartet im Anschluß auf die Abarbeitung dieser Aufträge. Nachdem die entsprechenden Befehle durch die *PC-Task* im *rw_MOS*-Betriebssystem selbsttätig ausgeführt wurden, können neue Befehlsaufträge an die *PC-Task* übermittelt werden. Die Zeit zwischen Befehlsauftrag und Befehlsabarbeitung kann das Anwenderprogramm nutzen, um weitere applikationsspezifische Aufgaben zu erledigen.
Da die Programmierung durch den direkten Zugriff eines PC-Anwenderprogramms erfolgt, wird diese Programmiermethode auch PC-Direkt-Programmierung genannt.
Anmerkung: In den nachfolgenden Kapiteln wird gelegentlich der Begriff PCAP-Befehl verwendet. Dieser Befehlstyp hat die soeben beschriebene Programmiermethode zur Grundlage.

3.2 Standalone-Applikations-Programmierung (SAP-Programming)

Im Gegensatz zur PC-Applikations-Programmierung gestattet die Stand-Alone-Applikations-Programmiermethode eine Programmabarbeitung gänzlich ohne Hilfe eines PC-Anwenderprogrammes. Ein in der Programmiersprache *rw_SymPas* erstelltes Anwenderprogramm wird mit Hilfe des in der Entwicklungsumgebung *mcfg.exe* integrierten Compilers *NCC* oder des Kommandozeilencompilers *ncc.exe* übersetzt und erzeugt ein für die MCU-G3 verständliches Betriebsprogramm. Dieses Betriebsprogramm kann auf die MCU-G3 geladen werden und wird mit Hilfe der *CNC-Task* (CNC = Computerized Numerical Control) in *rw_MOS* selbsttätig ausgeführt. Sofern eine Synchronisation zwischen einem PC-Anwenderprogramm und dem MCU-G3-Standalone-Programm notwendig ist, kann diese mit Hilfe vordefinierter System-Variablen, auf welche beide System-Partner (PC und MCU-G3) zugreifen können, durchgeführt werden.
Anmerkung: In den nachfolgenden Kapiteln wird des öfteren der Begriff SAP-Befehl verwendet. Dieser Befehlstyp hat die soeben beschriebene Programmiermethode zur Grundlage.

3.2.1 SAP-Multitasking

Die Betriebssystemsoftware *rw_MOS* kann bis zu 4 SAP-Programme gleichzeitig abarbeiten. Alle gleichzeitig ausgeführten Tasks haben die gleiche Priorität. Die verschiedenen Task werden über Nummern angesprochen. Die kleinste Tasknummer hat den Wert 0 und die größte somit den Wert 3.
Die Multitasking-Programmierung ermöglicht es, eine komplexe Aufgabe in kleine überschaubare Teilaufgaben zu zerlegen. Beispielsweise könnte eine Task zur Referenzfahrt, eine andere zur Überwachung des Antriebes mit den entsprechenden EVENT-Handlern und wieder eine andere zur reinen SPS-Steuerung mit entsprechenden Zugriffen auf Digital-I/O bzw. PC-Kommunikation mit vordefinierten Registern verwendet werden.
Die verschiedenen SAP-Programme können sich mit Hilfe verschiedener Task-Steuerbefehle selbsttätig stoppen, starten oder auch fortsetzen.
Die Synchronisation der CNC-Tasks untereinander und die Synchronisation auf ein evt. parallel ablaufendes PCAP-Anwenderprogramms bzw. der Datenaustausch zwischen diesen kann durch vordefinierte Register, den sogenannten COMMON-Variablen, ausgeführt werden. Hierzu stehen für alle CNC-Tasks 1000 gemeinsame Ganzzahl- und 1000 gemeinsame Gleitpunkt-Register zur Verfügung.
Jeder CNC-Task steht zusätzlich ein lokaler Speicherbereich mit einer Größe von 1000 Bytes (COMMON BUFFER) zur Verfügung, auf den sowohl der PC als auch die entsprechende CNC-Task sowohl lesend als auch schreibend zugreifen kann. Dieser kann z.B. zum Aufbau eines benutzerspezifischen Befehlssatzes verwendet werden.

4 PC-Applikations-Programmierung

4.1 Einführung

In der MCU-G3 TOOLSET Software sind Bibliotheksfunktionen für die Programmiersprachen *Borland Delphi*, *C (z.B Borland C++Builder, Microsoft Visual C++)* und *Microsoft Visual Basic* enthalten. Es handelt sich hierbei um Programmierwerkzeuge für die Windows-Plattformen Windows 95, 98, Me, Windows NT 4.0, Windows NT Embedded 4.0, Windows 2000, XP, Vista und Windows 7. Die einzelnen Funktionen dieser Hochsprachenbiblitheken werden unter Zuhilfenahme des Systemtreibers *mcug3.dll* ausgeführt. Die Bedeutung der einzelnen Funktionsparameter und deren Datentypen ist für die oben erwähnten Programmiersprachen identisch.
Das Einbinden der Funktionenbibliotheken in die jeweilige Programmiersprache wird nachfolgend erläutert:

Programmiersprache	Hinweise zur Anwendung
Borland Delphi	Der Name der Funktionenbibliothek ist *mcug3.pas*. Mit Hilfe dieser Funktionen wird die Verbindung zwischen PC-Applikations-Programm und dem Systemtreiber *mcug3.dll* hergestellt. Diese Datei ist als *unit* deklariert und wird mit Hilfe der *uses*-Anweisung zum Anwenderprogramm gebunden. Achtung: Verschiedene Systemparameter besitzen den Datentyp *double*. Dies bedeutet, dass das Anwenderprogramm mit der Option *{$N+}* compiliert werden muss!
C **(Borland *C, Microsoft C u.a.)***	Der Name der Funktionenbibliothek ist *mcug3*.lib. Mit Hilfe der dort enthaltenen Funktionen wird die Verbindung zwischen PC-Applikations-Programm und dem Systemtreiber *mcug3.dll* hergestellt. Die Lib-Dateien werden für verschiedene C-Programmierwerkzeuge bereitgestellt und müssen mit dem Anwenungsprogramm gebunden werden. Die Funktionsdeklarationen sind in der Datei *mcug3.h* enthalten. Diese Datei kann mit Hilfe der *#include*-Anweisung in das Anwenderprogramm miteingebunden werden.
Microsoft Visual Basic	Der Name der Funktionenbibliothek ist *mcug3.bas*. Mit Hilfe der in *mcug3.bas* deklarierten Funktionen wird die Verbindung zwischen PC-Applikations-Programm und dem Systemtreiber *mcug3.dll* hergestellt. Diese Datei ist als Basic-Modul verfügbar und kann in die Projektumgebung des Anwenderprogramms hinzugefügt werden.

4.2 Beispielprogramme zur Anwendung der Funktionenbibliotheken

Die in der MCU-G3 TOOLSET Software enthaltenen Beispielprogramme zeigen die einfache Anwendung nachfolgend beschriebener Funktionen. Die Quelltexte der Beispielprogramme sind durch Kommentare selbsterklärend. Deshalb wird an dieser Stelle auf eine detaillierte Programmbeschreibung dieser Beispiele verzichtet. Die einzelnen Beispielprogramme für die beiden Programmiersprachen sind in den angegebenen Unterverzeichnissen zu finden und haben folgende Namen:

Programmiersprache	Unterverzeichnis	Dateien
Borland Delphi	Delphi	*mcug3.pas*, *ld.pas, move.pas* usw.
Borland C++ Builder	C C/borland	*mcug3.h*, *ld.c, move.c* usw. *mcug3.lib*
Microsoft Visual C++	C C/mvc	*mcug3.h*, *ld.c, move.c* usw. *mcug3.lib*
Microsoft Visual Basic	Vb	*mcug3*.bas, *ld.bas, move*.bas usw.

4.3 Definitionen, Strukturen und Records

Bevor die einzelnen Funktionen erklärt werden, erfolgt die Beschreibung einiger Definitionen, Strukturen bzw. Records die zum Teil als Parameter für diese Funktionen benötigt werden. Die Deklaration der benötigten Struktur- bzw. Record-Datenfelder erfolgt immer im PC-Anwenderprogramm. Dies hat den Vorteil, dass der Systemtreiber nur wenig PC-Arbeitsspeicher belegt, und verschiedene PC-Anwendungen gleichzeitig auf die MCU-G3-Controller zugreifen können.
Alle nachfolgend abgedruckten Struktur- bzw. Record-Typen und Systemkonstanten sind in den oben genannten Programmiersprachen in den Dateien *mcug3.h*, *mcug3.pas* bzw. *mcug3.bas* definiert.

4.3.1 Definitionen

Tabelle 1: Systemkonstanten

Name	Typ	Funktion
MAXAXIS	integer	Maximale Anzahl der möglichen Achsen. Derzeit unterstützt die TOOLSET Software bis zu 18 Achsen. **Achtung:** Dieser Wert darf nicht verändert werden!
LONGINT	integer int long	Synonym für den Datentyp int bzw. integer in der Programmiersprache *C* bzw. *DELPHI Pascal* und longint in der Programmiersprache *Microsoft Visual Basic*.

4.3.2 Strukturen, Records und Typen

Je nach Programmiersprache spricht man entweder von Strukturen *(C)*, Records *(Pascal*) oder Typen *(Visual Basic)*. Der Aufbau und die Funktionsweise dieser Datentypen ist für alle Programmiersprachen identisch. Im weiteren Verlauf wird der Begriff Struktur - und Record-Typ verwendet, der diese Datentypen umfasst. Zur Steigerung der Übersichtlichkeit sind alle Struktur- bzw. Record-Typen groß und deren Komponenten klein geschrieben.

4.3.2.1 Struktur- und Record-Typ AS

Tabelle 2: Struktur- und Record-Typ AS

Element	Typ	(Kurzwortbedeutung), Funktion
unoa	LONGINT	(used number of axis) Anzahl der anzuwählenden Achsen bei verschiedenen Funktionsaufrufen.
san	Feld mit MAXAXIS LONGINT	(selected axis number) Feld der anzuwählenden Achsen. Dieses Feld ist mit Index 0 beginnend je nach Anzahl der verwendeten Achsen zu initialisieren.

Anmerkung: Die Zählweise der Achskanäle beginnt bei dem Wert 0.

```
Beispiel: Anwahl der ersten und dritten Achse

as.unoa = 2;     // Anzahl der Achsen
as.san[0] = 0;   // erste Achse
as.san[1] = 2;   // dritte Achse
```

4.3.2.2 Struktur- und Record-Typ TSRP

Um mit den einzelnen Achssystemen arbeiten zu können, muss für alle Achsen je ein Struktur- bzw. Record-Typ *TSRP* deklariert werden. Mit Hilfe der in TSRP enthaltenen Struktur- bzw. Record-Elemente erfolgt bei verschiedenen PCAP-Befehlen der Datenaustausch mit der MCU-G3. So können achsspezifische Systemgrößen wie Beschleunigungen, Geschwindigkeiten und Positionen mit Hilfe spezieller Lese- und Schreibbefehle abgefragt bzw. gesetzt werden.
Achtung: Die einzelnen Elemente der Struktur TSRP werden nicht automatisch initialisiert, d.h. der Anwender muss diese durch direktes Setzen bzw. vorheriges Lesen aktualisieren.

Anmerkung: Es muss darauf geachtet werden, dass bei der Verwendung von mehr als einem Achskanal die Strukturen bzw. Records *TSRP* im Speicher direkt hintereinander angeordnet werden, da der Systemtreiber *mcug3.dll* zum Teil mit Hilfe von Adreßberechnungen auf die verschiedenen Achsparameter zugreift. Deshalb muss ggf. die Datenausrichtung auf 4 Bytes eingestellt werden. Die korrekte Anordnung im PC-Arbeitsspeicher wird erzwungen, indem TSRP als Feld-Variable deklariert wird. Die Größe des Feldes muß für MAXAXIS Achsen definiert werden.
Vor der Verwendung muß diese Datenstruktur initialisiert sein. Die Initalisierung erfolgt z.B. mit den Befehlen InitMcuSystem, InitMcuSystem2 oder InitMcuSystem3. In den meisten Fällen ist eine Instanz dieser Datenstruktur für jede Steuerung im System global definiert und wird beim Programmaufruf, bzw. nach dem Booten der Steuerung initialisiert. Eine Verwendung lokal deklarierter Instanzen ohne vorherige Initialisierung ist nicht erlaubt und kann zu unverhersehbaren Fehlfunktionen führen.

Tabelle 3: Struktur- und Record-Typ TSRP (achsspezifische Parameter)

Element	Typ	(Kurzwortbedeutung), Funktion
an	LONGINT	(axis number)
kp	double	(PIDF filter parameter kp)
ki	double	(PIDF filter parameter ki)
kd	double	(PIDF filter parameter kd)
kpl	double	(PIDF filter parameter kpl)
kfca	double	(PIDF forward compensation acceleration)
kfcv	double	(PIDF forward compensation velocity)
jac	double	(jog acceleration)
jvl	double	(jog velocity)
jtvl	double	(jog target velocity)
jovr	double	(jog override)
hac	double	(home acceleration)
hvl	double	(home velocity)
rp	double	(real position)
dp	double	(desired position)
tp	double	(target position)
sll	double	(software limit left)
slr	double	(software limit right)
ipw	double	(in position window)
mpe	double	(maximum position error)
gf	double	(gear factor)
mcp	LONGINT	(motor command port)
axst	LONGINT	(axis status)
lsm	LONGINT	(left spool memory)
epc	LONGINT	(eeprom programming cycle)
digi	LONGINT	(digital inputs)

Tabelle 3 (Fortsetzung): Struktur- und Record-Typ TSRP

Element	Typ	(Kurzwortbedeutung), Funktion
digo	LONGINT	(digital outputs)
ifs	LONGINT	(interface status)
scratch	Feld mit 4 mal LONGINT	(scratch field) Platzhalter zum nächsten TSRP-Satz

4.3.2.3 Struktur- und Record-Typ TRU (Trajectory Units)

Dieser Struktur- bzw. Recordtyp ist Parameter für den PCAP-Befehl *ctru()*.

Tabelle 4: Struktur- und Record-Typ TRU

Element	Typ	(Kurzwortbedeutung), Funktion
pu	LONGINT	(position unit) Positions-Einheit
tu	LONGINT	(time unit) Zeit-Einheit

4.3.2.4 Struktur- und Record-Typ LMP (Linear Motion Parameters)

Dieser Struktur- bzw. Recordtyp ist Parameter bei allen linearen Interpolationsbefehlen.

Tabelle 5: Struktur- und Record-Typ LMP

Element	Typ	(Kurzwortbedeutung), Funktion
ac	double	(acceleration) Bahnbeschleunigung
vl	double	(velocity) Bahngeschwindigkeit
tvl	double	(target velocity) Bahnzielgeschwindigkeit
dtm	Feld mit MAXAXIS double	(distance to move) Dieses Feld ist je nach Index der verwendeten Achsen zu initialisieren. Die Zählweise des Index beginnt bei 0. In die einzelnen Elemente werden die gewünschten Verfahrwege je nach Positionierart (absolut bzw. relativ) eingetragen. Die Eintragungen in diesem Datenfeld müssen mit den selektierten Achsen im Struktur- bzw. Record-Typ AS übereinstimmen. Der Verfahrweg z.B. der 5. Achse (Achsindex 4) muß also immer im Element dtm[4] eingetragen werden.

4.3.2.5 Struktur- und Record-Typ CMP (Circular Motion Parameters)

Dieser Struktur- bzw. Recordtyp ist Parameter bei allen zirkularen Interpolationsbefehlen.

Tabelle 6: Struktur- und Record-Typ CMP

Element	Typ	(Kurzwortbedeutung), Funktion
ac	double	(acceleration) Bahnbeschleunigung
vl	double	(velocity) Bahngeschwindigkeit
tvl	double	(target velocity) Bahnzielgeschwindigkeit
phi	double	Verfahrwinkel in Grad
dtca1	double	(distance to center x-Achse)
dtca2	double	(distance to center y-Achse) Die Zuordnung von dtca1 und dtca2 an die gewünschten Achskanäle wird mit dem Struktur- bzw. Record-Typ AS hergestellt. Der dort im Feld 0 eingetragene Achskanal ist die x-Achse. Im Feld 1 wird entsprechend die y-Achse eingetragen.

4.3.2.6 Struktur- und Record-Typ HMP (Helical Motion Parameters)

Dieser Struktur- bzw. Recordtyp ist Parameter bei allen Schraubenlinien-Interpolationsbefehlen.

Tabelle 7: Struktur- und Record-Typ HMP

Element	Typ	(Kurzwortbedeutung), Funktion
ac	double	(acceleration) Bahnbeschleunigung
vl	double	(velocity) Bahngeschwindigkeit
tvl	double	(target velocity) Bahnzielgeschwindigkeit
phi	double	Verfahrwinkel in Grad Das Vorzeichen bestimmt die Kreis-Richtung. Mit einem Betrag des Verfahrwinkels <= 1e-100 wird ein Kreis mit Zielpunktsangabe abgefahren.
dtca1	double	(distance to center x-Achse)
dtca2	double	(distance to center y-Achse)
dtm	Feld mit MAXAXIS double	(distance to move z-Achse und höher) Dieses Feld ist je nach Index der verwendeten Achsen zu initialisieren. Die Zählweise des Index beginnt bei 0. (Siehe auch LMP). In die einzelnen Elemente werden die gewünschten Verfahrwege je nach Positionierart (absolut bzw. relativ) eingetragen.. Bei Abfahren einer Helix mit Angabe des Kreiswinkels werden hier die Verfahrwege / Zielpunkte der linear interpolierenden Achsen ab Index 2 eingetragen. Bei Abfahren einer Helix unter Angabe des Zielpunktes, werden hier ebenfalls die Zielpunkte der Zirkularachsen eingetragen.

4.3.2.7 Struktur- und Record-Typ HMP3D (Helical Motion Parameters 3Dimensional)

Dieser Struktur- bzw. Recordtyp ist Parameter bei allen 3D-Interpolationsbefehlen.

Tabelle 8: Struktur- und Record-Typ HMP3D

Element	Typ	(Kurzwortbedeutung), Funktion
ac	double	(acceleration) Bahnbeschleunigung
vl	double	(velocity) Bahngeschwindigkeit
tvl	double	(target velocity) Bahnzielgeschwindigkeit
phi	double	Verfahrwinkel in Grad Das Vorzeichen bestimmt die Kreis-Richtung. Das Abfahren mit Zielpunktsvorgabe ist hier nicht möglich.
dtca1	double	(distance to center x-Achse)
dtca2	double	(distance to center y-Achse)
dtca3	double	(distance to center z-Achse)
pn1	double	Flächen-Normale X-Vektor
pn2	double	Flächen-Normale Y-Vektor
pn3	double	Flächen-Normale Z-Vektor
dtm	Feld mit MAXAXIS double	reserviert für zukünftige Erweiterungen

4.3.2.8 Struktur- und Record-Typ ROSI (Risc Operating System Informations)

Dieser Struktur- bzw. Recordtyp ist Parameter für den PCAP-Initialisierungsbefehl *mcuinit()*. Nach erfolgreicher Initialisierung der MCU-G3 werden folgende *rw_MOS*-Informationen (*rwmos.elf*) in die Struktur ROSI eingetragen:

Tabelle 9: Struktur- und Record-Typ ROSI

Element	Typ	(Kurzwortbedeutung), Funktion
revision	Feld mit SIZE_STRREV characters	Momentane Software-Revision der rw_MOS-Betriebssystemsoftware.
number_axis	LONGINT	Anzahl der vorhandenen Achskanäle
sysfile_loaded	LONGINT	Diese Status-Variable zeigt mit dem Wert 1 an, ob die Systemdatei bereits auf die MCU-G3 übertragen wurde.

Anmerkung: Mit Hilfe des PCAP-Initialisierungsbefehls *InitMcuSystem2() oder InitMcuSystem3()* wird die Systemdatei system.dat, welche hauptsächlich mit Hilfe des TOOLSET-Programms mcfg.exe verändert wird, auf die MCU-G3 übertragen und bewirkt dort die Initialisierung systeminterner Parameter wie z.B. Beschleunigungen, Geschwindigkeiten, Filterkoeffizienten, Grenzwerte usw. Dieser Ladevorgang muss einmalig pro Systemstart erfolgen.

4.3.2.9 Struktur- und Record-Typ CBCNT (Common Buffer CNC-Task)

Jeder CNC-Task steht ein lokaler Speicherbereich mit einer Größe von 1000 Bytes (COMMON BUFFER) zur Verfügung, auf den sowohl der PC als auch die entsprechende CNC-Task sowohl lesend als auch schreibend zugreifen kann. Dieser kann z.B. zum Aufbau eines benutzerspezifischen Befehlssatzes verwendet werden.

Der Struktur- bzw. Recordtyp *CBCNCT* ist Parameter für die PCAP-Befehle *rdcbcnct()* und *wrcbcnct()*, mit deren Hilfe die COMMON BUFFER gelesen bzw. beschrieben werden können.

Tabelle 10: Struktur- und Record-Typ CBCNCT

Element	Typ	(Kurzwortbedeutung), Funktion
TaskNr	LONGINT	Task-Nummer (0..3)
size	LONGINT	Größe des Buffers [Bytes]
Buffer	Pointer	Zeiger auf einen Puffer, welcher zur MCU-G3 übertragen werden soll, bzw. von der MCU-G3 gelesen werden soll. Der Puffer muss mindestens size Bytes groß sein!

4.3.2.10 Struktur- und Record-Typ CNCTS (Computerized Numerical Control Task Status)

Dieser Struktur- bzw. Recordtyp ist Parameter für den PCAP-Statusabfragebefehl *rdcncts()*.

Tabelle 11: Struktur- und Record-Typ CNCTS

Element	Typ	(Kurzwortbedeutung), Funktion
errnum	LONGINT	Interne CNC-Task Fehlernummer. Sofern kein Fehler aufgetreten ist, hat errnum den Wert 0. Informationen über Laufzeitfehler sind zu finden in 6.8
errline	LONGINT	Im Zusammenhang mit errnum wird mit diesem Element die fehlerverursachende Quelltextzeile des CNC Stand- Alone-Applikations-Programmes angezeigt.
stackfree	LONGINT	Momentan freier Stackbereich [Bytes] für die CNC- Task.
running	LONGINT	Dieses Status-Wort zeigt im Bit 0 an, ob die CNC-Task gerade ein Programm abarbeitet. Bit 1 zeigt an, dass sich die Task in der Einzelschritt-Betriebsart befindet, das System wartet auf einen Schritt- (stepcnct) oder Fortsetzungsbefehl (contcnct). Wenn im Halt-Modus und Bit 2 gesetzt ist, wird angezeigt, daß der Task-Stopp durch das SAP-Kommando writeln verursacht wurde. (Siehe hierzu auch die Anmerkungen zum Register MODEREG Bit 26 Kapitel 6.3.1.5). Bit 3 zeigt an, dass sich die Task gerade im Wartezustand befindet (wt oder warten auf "End of Profile".
csrcline	LONGINT	aktuell in Ausführung befindliche Zeilennummer im Source-Quelltext

4.4 PCAP-Hochsprachen-Funktionenreferenzliste

4.4.1 Aufbau der Referenzliste

Die Funktionen- und Befehls-Referenzliste ist alphabetisch sortiert. Die Beschreibung der einzelnen Befehle und Funktionen hat dabei folgenden Aufbau:

Merkmal	Beschreibung
FUNKTIONSNAME:	Dies ist der Name, mit dessen Hilfe die nachfolgend beschriebene Funktion aufgerufen wird.
KURZWORTBEDEUTUNG:	Hier steht die ausführliche Beschreibung des entsprechenden Funktionsnamens.
BORLAND DELPHI :	Hier stehen die Prototypdefinitionen für die Programmiersprache *Borland Delphi (Pascal-Programmiersprache)*. Es wird ersichtlich, welche Parameter für den entsprechenden Funktionsaufruf benötigt werden.
C:	Prototypdefinition für die Programmiersprache *C*, wie z.B. *Microsoft Visual C++* oder *Borland C++Builder* sonst wie *Borland Delphi*.
VISUAL BASIC:	Prototypdefinition für die Programmiersprache *Microsoft Visual Basic* sonst wie *Borland Delphi*.
TSRP-KOMPONENTEN:	Verschiedene Funktionen benötigen als Parameter Komponenten der Struktur bzw. des Records TSRP. Diese werden hier aufgeführt.
BESCHREIBUNG:	Klartextbeschreibung des Befehls.
RÜCKGABEWERT:	Sofern die Funktion einen Rückgabewert zurückliefert, wird dieser hier beschrieben.
ANMERKUNG:	Bei immer wiederkehrenden Anmerkungen und Erläuterungen wird hier auf die entsprechenden Kapitel verwiesen.
BEISPIEL:	Gelegentlich werden Beispiele für die entsprechenden Funktionsaufrufe gegeben.

4.4.2 Generelle Informationen zu den PCAP-Befehlen

Alle Befehle und Funktionen, außer den *spool*-Befehlen, werden unmittelbar nach dem Aufruf ausgeführt. Bei allen *move*- und *jog*-Befehlen muss vor ihrer Ausführung sichergestellt werden, dass die beteiligten Achsen zuvor in Lageregelung geschaltet wurden (PCAP-Befehl *cl()*). Weiterhin wird bei den Bewegungsfunktionen z.T. zwischen Absolut- und Relativ-Verfahrbefehlen unterschieden. Die absoluten Verfahrbefehle werden im Absolutmaßsystem, also auf den Maschinennullpunkt bezogen, ausgeführt. Die relativen Verfahrbefehle werden inkremental, also von der momentanen Motorposition ausgehend, ausgeführt.
Das Ende der Profilabarbeitung wird sowohl im Direkt-Modus als auch im Spool-Modus durch das *pe*-Flag im *axst*-Register der Struktur bzw. dem Record TSRP angezeigt [Kapitel 4.4.53 - *rdaxst()*].
Bei den achsspezifischen Bewegungskommandos (*jog*-Befehle) werden alle Systemparameter wie Positionen, Verfahrwege, Beschleunigungen und Geschwindigkeiten in den im TOOLSET-Programm *mcfg.exe* festgelegten achsspezifischen Einheiten angegeben. Bei den Interpolationsbefehlen (*move*-Befehle) werden die in der Struktur (Record) TRU angewählten Einheiten herangezogen. Das bedeutet, daß vor der Ausführung von *move*-Befehlen zunächst ein PCAP-Funktionsaufruf *ctru()* erfolgen muß.
Die Umrechnung zwischen anwendungsspezifischen und systeminternen Einheiten erfolgt automatisch mit Hilfe der in *mcfg.exe* festgelegten Faktoren. Maßgeblich für die Umrechnung sind die Enkoderauflösung bzw. Schrittzahl, der Getriebefaktor und die angewählten Weg- und Zeiteinheiten.

4.4.2.1 Funktionswerte und Funktions-Rückgabewerte

Der Funkionswert ist der Wert, der bei einem Lesebefehl von der Steuerung gelesen werden soll. Der Funkions-Rückgabewert ist der Wert, den ein Befehlsaufruf zurückliefert, dieser ist in vielen Fällen nicht der Funktionswert, sondern ein Wert, welcher den Erfolg oder eine Fehlerinformation zum Aufruf einer DLL-Funktion anzeigt. Auch Schreibbefehle können einen Funktionsrückgabewert liefern aber nicht alle Funktionen liefern einen Funktionsrückgabewert.
Falls durch ein unvorhergesehenes Ereigniss auf der Karte z.B. eine Exception oder durch einen Benutzereingriff in einem parallel ablaufenden Programm das Betriebssystem der Steuerungsbaugruppe angehalten wird, dann passiert in einem Anwenderprogramm, welches DLL Funktionen aufruft folgendes:
In dem Aufruf, welcher während oder nach dem Ereignis aufgerufen wird, wird die Funktion nach einer Timeoutzeit von mehreren Sekunden erfolglos beendet. Der Erfolg des Aufrufs kann aber nur bei den Funktionen ermittelt werden, welche eine Statusinformation über den Erfolg zurückliefern.
Wenn die Kommunikation über die DLL erst einmal unterbrochen wurde, kann diese nur durch eine Reinitialsierung z.B. per InitMcuSystem3() wiederhergestellt werden. Das ist z.B. dann möglich wenn die Karte von einem Program oder durch sich selbst neu gebootet wurde. In diesem Fall ist aber sowieso in den meisten Fällen eine komplette Neuinitialisierung der Anwendung erforderlich.

4.4.3 azo, activate zero offsets

BESCHREIBUNG:	Jedem Achskanal können fünf unterschiedliche Nullpunktverschiebungen *(zero offsets)* zugeordnet werden. Mit Hilfe des Befehls *azo()* können die gewünschten achsspezifischen Verschiebungsparameter aktiviert werden. Im Parameter *set* (bzw. *set_*) wird spezifiziert, welcher Satz von Nullpunktverschiebungen aktiviert werden soll. Diese Variable wählt mit dem Wert 0..4 den gewünschten Satz von Nullpunktverschiebungen an. Sofern die Variable jedoch einen Wert größer als 4 hat, werden keine Nullpunktverschiebungen mehr berücksichtigt.
BORLAND DELPHI:	procedure azo(set_: integer);
C:	void azo(int set);
VISUAL BASIC:	Sub azo(ByVal set_ As Long)
ANMERKUNG:	Nullpunktverschiebungen dienen zur Festlegung eines neuen Koordinatensystems, ohne dabei den tatsächlichen Maschinennullpunkt beeinflussen (neu setzen) zu müssen. Der aktuell gesetzte Positionswert der Nullpunktverschiebung kann mit dem Kommando rdZeroOffset (Kapitel 4.4.118) gelesen werden.
RÜCKGABEWERT:	keiner

4.4.4 BootErrorReport, initialision error report

BESCHREIBUNG:	Mit dieser Funktion können die Fehlerrückgabwerte der nachfolgend beschriebenen Funktion BootFile() im Klartext angezeigt werden. Hierbei wird eine Message-Box am Bildschirm eröffnet, welche wiederum durch den Anwender quittiert werden muß.
BORLAND DELPHI:	procedure BootErrorReport(filename:PChar; error:integer);
C:	void BootErrorReport(char *filename, int error);
VISUAL BASIC:	Sub BootErrorReport (ByVal filename As String, ByVal error As Long)
ANMERKUNG:	PCAP-Befehle BootFile()
BEISPIEL:	*booterror = BootFile(...); // Bootsequenz ausführen* *BootErrorReport(..., booterror); // Im Fehlerfall Fehlerrückgabewert // anzeigen*
RÜCKGABEWERT:	keiner

4.4.5 BootFile, boot operating system file

BESCHREIBUNG:	Diese Funktion dient zum Übertragen der Betriebssystemsoftware (rwmos.elf) auf die Steuerung. Das System wird zunächst zurückgesetzt. Anschließend wird die in *BootFileName* spezifizierte Datei (normalerweise rwmos.elf) auf die Steuerung geladen.
BORLAND DELPHI:	function BootFile(var BootFileName:string; TpuBaseAddress: integer):integer;
C:	int BootFile(char* BootFileName, int TpuBaseAddress);
VISUAL BASIC:	Function BootFile(ByVal filename As String, ByVal TpuBaseAddress As Long) As Long
ANMERKUNG:	Nach erfolgreichem Bootvorgang muss noch die Funktion InitMcuSystem2() oder InitMcuSystem3() aufgerufen werden, damit die Steuerung komplett initialisiert wird. *TpuBaseAddress* existiert zur Kompatibilität mit den MCU-G2-Controllern und sollte mit dem Wert 0 initialisiert werden.
RÜCKGABEWERT:	Die Funktion liefert folgende Rückgabewerte:

Rückgabewert	Fehler-Beschreibung
0	kein Fehler, Bootvorgang erfolgreich abgeschlossen.
10	Der in *BootFileName* spezifizierte Dateiname ist ungültig.
11	Die in *BootFileName* spezifizierte Datei konnte nicht geöffnet werden.
12	Unerkanntes Dateiformat. Zur Zeit sind nur Dateien mit dem ELF-Dateiformat zulässig.
13	Ungültiges ELF-Dateiformat oder Übertragungsfehler.
14	Ungültige Startadresse in RWMOS.ELF ermittelt RWMOS.ELF ist u.U. fehlerhaft
15	Ungültige Plattform für RWMOS.ELF Das verwendete RWMOS.ELF ist nicht für die vorliegende Hardwareplattform geeignet.
16	Verify bei Übertragung des Bootfiles schlug fehl, die Datei wurde fehlerhaft übertragen.
200	Kein Zugriff auf Flash-Memory-System des Zielsystems möglich.
201	Ungültige Flash-Speichergröße auf Zielsystem.
202	Auf die benötigten Geräteadressen des Zielsystems ist kein Zugriff möglich.

4.4.6 CardSelect

BESCHREIBUNG:	Mit dieser Funktion kann ein MCU-G3 Controller selektiert werden, falls Mehrere im PC installiert sind. Die Selektion ist so lange wirksam bis die Funktion CardSelect für ein anderes Gerät aufgerufen wird oder bis die Applikation beendet wird. Nach der Selektion beziehen sich alle Befehle der mcug3.dll, die innerhalb der Applikation aufgerufen werden, auf das selektierte Gerät.
BORLAND DELPHI:	functionCardSelect (CardNum: integer): integer;
C:	int CardSelect (int CardNumber);
VISUAL BASIC:	Function CardSelect (ByVal CardNr As Long) As Long
PARAMETER:	Index der Karte im PC (0, 1, ...)
RÜCKGABEWERT:	Index der Karte die erfolgreich selektiert wurde. –1 wenn das angewählte Gerät nicht im PC existiert, in diesem Fall ist das Gerät mit Index 0 angewählt). **Vor der Verwendung dieses Kommandos muss eines der InitMcuSystem-Kommandos aufgerufen werden**, damit die interne Liste der verfügbaren Geräte aktuell ist.
ANMERKUNG:	siehe auch [IHB / Kapitel 5.3]

4.4.7 ClearCI99

BESCHREIBUNG:	Mit dieser Funktion wird die Common-Integer Variable CI99 synchron zur Betriebssystemsoftware RWMOS.ELF abgenullt.
BORLAND DELPHI:	procedure ClearCI99 ();
C:	void ClearCI99 (void);
VISUAL BASIC:	Sub ClearCI99 ()
PARAMETER:	keiner

RÜCKGABEWERT:	keiner, die Variable CI99 wird auf 0 gesetzt.
ANMERKUNG:	Diese Funktion muss verwendet werden, wenn die ssf-Funktionen 1005 – 1025 zur Synchronisation von Spoolerkommandos eingesetzt werden.

4.4.8 cl, close loop

BESCHREIBUNG: **MCU-3000 / APCI-8001:** **MCU-3100 / APCI-8008:** **MCU-3400C/CPCI-8004:**	Alle in AS spezifizierten Achskanäle werden mit diesem Befehl in die Lageregelung gebracht. Dabei werden die Istpositionen der beteiligten Achsen als Sollpositionen übernommen, um große Regelabweichungen zu vermeiden. Zusätzlich werden alle mit PAE-projektierten Digital-Ausgänge gesetzt. Diese Ausgänge können beispielsweise zur Ansteuerung von Relais verwendet werden, mit welchen wiederum die Freigabe der Leistungsverstärkereinheiten erfolgt. Je nach selektiertem Achskanal werden die Freigabe-Relais des zugeordneten Achskanals eingeschaltet [IHB / Kapitel 5.2.10].
BORLAND DELPHI:	procedure cl(var as:AS);
C:	void cl(struct AS far *as);
VISUAL BASIC:	Sub cl(DASEL As ASEL) 'close loop
ANMERKUNG: **MCU-6000 / APCI-8401:**	Die Lageregelung bewirkt das Abarbeiten des PIDF-Filters mit den entsprechend eingestellten Filterkoeffizienten. Beim Schließen des Lageregelkreises werden alle Spoolerdaten der spezifizierten Achskanäle verworfen! Siehe hierzu auch PCAP-Kommando clv(). Das Kommmando cl ist nur erfolgreich, wenn die Bits TOASM in AXST und dc in asms nicht gesetzt sind. Deshalb muss der Erfolg von cl bei diesen Systemen überprüft werden.

4.4.9 clv, close loop velocity

BESCHREIBUNG: **MCU-3000 / APCI-8001:** **MCU-3100 / APCI-8008:** **MCU-3400C/CPCI-8004:**	Alle in AS spezifizierten Achskanäle werden mit diesem Befehl in die Lageregelung gebracht. Dabei werden die Istpositionen der beteiligten Achsen als Sollpositionen und die Istgeschwindigkeiten als Sollgeschwindigkeiten übernommen, um große Regelabweichungen zu vermeiden. Zusätzlich werden alle mit PAE-projektierten Digital-Ausgänge gesetzt. Dieses Kommando sollte verwendet werden, wenn die Achsen vor dem Schließen des Regelkreises in Bewegung sind. Die entsprechenden Achsen erhalten dann beim Schließen des Regelkreises die aktuelle Geschwindigkeit und fahren somit geregelt weiter. Diese können nun z.B. per js() geregelt abgebremst werden. Dadurch wird ein harter Stopp der Achsen beim Schließen des Regelkreises verhindert. Je nach selektiertem Achskanal werden die Freigabe-Relais des zugeordneten Achskanals eingeschaltet [IHB / Kapitel 5.2.10].
BORLAND DELPHI:	procedure clv(var as:AS);
C:	void clv(struct AS far *as);
VISUAL BASIC:	Sub clv(DASEL As ASEL) 'close loop velocity
ANMERKUNG:	siehe auch PCAP-Kommando cl()

4.4.10 contcnct, continue numeric controller task

BESCHREIBUNG:	Mit diesem Befehl kann ein SAP-Programm, welches zuvor mit dem SAP-Befehl *STOP*, *STOPCNCT()* oder mit dem PCAP-Befehl *stopcnct()* angehalten wurde, wieder fortgesetzt werden. Die in *TaskNr* (Werte 0..3) angewählte Task wird fortgesetzt.
BORLAND DELPHI:	procedure contcnct(TaskNr:integer);
C:	void contcnct(int TaskNr);
VISUAL BASIC:	Sub contcnct(ByVal TaskNr As Long)
ANMERKUNG:	Ein SAP-Programm, welches mit dem SAP-Befehl *ABORT* angehalten wurde, kann nur mit dem SAP-Befehl *STARTCNCT()* oder dem PCAP-Befehl *startcnct()* neu gestartet, also nicht fortgesetzt, werden.

4.4.11 ctru, change trajectory units

BESCHREIBUNG:	Mit diesem Befehl können die Einheiten für die Geschwindigkeits-, Beschleunigungs- und Positionsparameter aller Interpolationsbefehle (*move*-Befehle) umgeschaltet werden. Die Parameter werden in den gewählten Einheiten angegeben.
BORLAND DELPHI:	procedure ctru(var tru:TRU);
C:	void ctru(struct TRU far *tru);
VISUAL BASIC:	Sub ctru(DTRU As tru)
ALLE SPRACHEN:	Für die TRU-Strukturkomponente *pu* (position unit) sind folgende Werte erlaubt:

Index	Einheit	Beschreibung
0	mm	Millimeter
1	inch	Inch
2	m	Meter
3	rev	Revolution
4	deg	Degree
5	rad	Radiant
6	counts	Counts
7	steps	Steps

Für die TRU-Strukturkomponente *tu* (time unit) sind folgende Werte erlaubt:

Index	Einheit	Beschreibung
0	sec	Seconds
1	min	Minutes
2	tsample	Sampling Time

ANMERKUNG:	Der Defaultwert für *pu* und *tu* ist 0. Somit wird für alle Wegangaben die Einheit [mm], für Geschwindigkeiten die Einheit [mm/s] und Beschleunigungen [mm/s^2] angenommen. Die gewählten Einheiten werden nur für Interpolationsbefehle (alle *move*-Befehle) herangezogen! Sofern es sich um achsspezifische Bewegungskommandos handelt (alle *jog*-Befehle), werden die in *mcfg.exe* spezifizierten Achs-Einheiten berücksichtigt. Die angewählten Einheiten sind auch maßgebend für ein evtl. parallel ablaufendes SAP-Programm. In der rw_SymPas Programmierumgebung werden diese Parameter über die System-Parameter PU und TU angesprochen (Tabelle 34).

4.4.12 getEnvStr, get Environment String

BESCHREIBUNG:	Mit dieser Funktion wird die im String- bzw. Zeichen-Parameter spezifizierte Umgebungsvariabel von der Steuerung ausgelesen und der Wert in den aufrufenden Parameter eingetragen.
BORLAND DELPHI:	function getEnvStr (var EnvStr:CppString):integer;
C:	int getEnvStr (char far * EnvStr);
VISUAL BASIC:	Function getEnvStr (ByVal EnvStr As String) As Long
RÜCKGABEWERT:	Die Funktion kann folgende Werte zurückliefern: **Rückgabewert** / **Fehler-Beschreibung** **-1** Fehler: z.B. RWMOS stellt die Funktion nicht zur Verfügung **-4** Fehler: Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen **0** Der Parameter wurde nicht gefunden oder ist ein leerer String **> 0** gibt die Stringlänge der gefundenen Zeichenkette an (ohne abschließendes Nullbyte). D.h. also ein Rückgabewert >= 0 zeigt die erfolgreiche Ausführung des Kommandos an.
ANMERKUNGEN:	Mit Hilfe dieser Funktion kann ein Applikationsprogramm das Vorhandensein von Umgebungsvariablen prüfen, welche für die Anwendung von wichtiger Bedeutung sind. Auf diese Weise kann eine Anwendung auch dann kontrolliert reagieren, wenn z.B. durch einen Hardwaretausch wichtige Eigenschaften der Steuerung nicht mehr vorhanden sind. Das Beschreiben von Umgebungsvariablen ist nur bei ungebootetem System in fwsetup möglich. Diese Funktion existiert erst in RWMOS.ELF ab V2.5.3.37 und mcug3.dll ab V2.5.3.25. Der Datentyp CppString für Delphi ist in mcug3.pas Versionsabhängig definiert.
DELPHI BEISPIEL:	EnvString : CppString; ..... EnvString := allocmem (1024); StrPCopy (EnvString, 'SerialNumber'); getEnvStr (EnvString);

4.4.13 gettskinfo, Get Task Informations

BESCHREIBUNG:	Mit diesem Befehl kann eine Task abgefragt werden, ob ein noch nicht ausgelesener String vorliegt.
BORLAND DELPHI:	function gettskinfo (TaskNr: integer; var tskinfo: integer): integer;
C:	int gettskinfo (int TaskNr, int *tskinfo);
VISUAL BASIC:	Function gettskinfo (ByVal tasknr As Long, tskinfo As Long) As Long
Parameter:	TaskNr: Tasknummer (0..3) tskinfo: In dieser Variablen wird der Funktionswert zurückgeliefert
Rückgabewert:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung

ANMERKUNG:	Der Funktionswert wird in tskinfo zurückgeliefert. Bit 0 zeigt an, daß ein noch nicht abgeschlossener String (write) vorliegt. Bit 1 zeigt an, daß ein abgeschlossener String vorliegt (writeln). Die entsprechenden Bits werden durch das Auslesen des String per gettskstr() automatisch zurückgesetzt. Task Message Strings können in der Programmierumgebung der Stand-Alone-Tasks per WRITE oder WRITELN erzeugt werden (Kapitel 6.6.80 und 6.6.81).

4.4.14 gettskstr, Get Task Message String

BESCHREIBUNG:	Mit diesem Befehl kann der taskspezifische Ausgabestring gelesen werden.
BORLAND DELPHI:	function gettskstr (TaskNr: integer; buffer: PChar, szbuffer: integer): integer;
C:	int gettskstr (int TaskNr, char * buffer, int szbuffer);
VISUAL BASIC:	Function gettskstr (ByVal tasknr As Long, ByVal buffer As String, ByVal szbuffer) As Long
Parameter:	TaskNr: Tasknummer (0..3) buffer: In dieser Variablen wird die gelesene Zeichenkette zurückgeliefert szbuffer: Maximale Größe des zu lesenden Strings
Rückgabewert:	Anzahl der gelesenen Zeichen
ANMERKUNG:	Dieser Aufruf setzt die entsprechenden Zustandsbits in tskinfo zurück. Der Speicherbereich von TskStr muß groß genug sein, um die zurückgelieferte Zeichenkette aufzunehmen. Maximal werden 512 bytes zurückgeliefert. Task Message Strings können in der Programmierumgebung der Stand-Alone-Tasks per WRITE oder WRITELN erzeugt werden (Kapitel 6.6.80 und 6.6.81).

4.4.15 InitMcuErrorReport, initialision error report

BESCHREIBUNG:	Mit dieser Funktion können die Fehlerrückgabwerte der nachfolgend beschriebenen Funktionen InitMcuSystem(), InitMcuSystem2() und InitMcuSystem3() im Klartext angezeigt werden. Hierbei wird eine Message-Box am Bildschirm eröffnet, welche wiederum durch den Anwender quittiert werden muß.
BORLAND DELPHI:	procedure InitMcuErrorReport(error:integer);
C:	void InitMcuErrorReport (int error);
VISUAL BASIC:	Sub InitMcuErrorReport (ByVal error As Long)
ANMERKUNG:	PCAP-Befehle InitMcuSystem(), InitMcuSystem2() und InitMcuSystem3()
BEISPIEL:	*initerror = InitMcuSystem3(...); // Initialisierung ausführen* *InitMcuErrorReport(initerror); // Im Fehlerfall Fehlerrückgabewert* *// anzeigen*

4.4.16 InitMcuSystem, initialise mcu system

BESCHREIBUNG:	Diese Funktion führt die komplette Software-Initialisierung des Antriebssystems durch. Der Funktionsaufruf muß zu Beginn (in jedem Fall vor anderen PCAP-Aufrufen) in jedem PCAP-Anwenderprogramms ausgeführt werden. Innerhalb dieser Funktion werden verschiedene PCAP-Basis-Funktionen aufgerufen. Unter anderem werden die Achsennummern {an} in der Struktur *tsrp* initialisiert. Sofern die Systemdatei *system.dat* noch nicht auf die MCU-G3 übertragen wurde, wird dies hier getan. Am Ende der Funktion werden die Achsparameter von allen Achsen in die Struktur *tsrp* eingelesen.
BORLAND DELPHI:	function InitMcuSystem(var tsrp:TSRP):integer;
C:	int InitMcuSystem(var TSRP far *tsrp);
VISUAL BASIC:	Function InitMcuSystem(DTSRP As TSRP) As Long
ANMERKUNG:	PCAP-Befehle *txbf2()*, *mcuinit()*, Struktur bzw. Record-Typ ROSI **Wichtig:** Diese Funktion besteht aus Kompatibilität zu den MCU-G2-Controllern. Anstelle dieser sollten die Funktionen InitMcuSystem2() oder besser noch InitMcuSystem3() verwendet werden.
RÜCKGABEWERT:	Die Funktion kann folgende Werte zurückliefern:

Rück-gabewert	Fehler-Beschreibung
0	Kein Fehler.
31	Es wurde kein MCU-G3-Controller gefunden.
32	Die rw_MOS-Betriebssystemsoftware ist nicht geladen oder wurde gestoppt. Siehe hierzu PCAP-Kommando *BootFile()* oder Dienstprogramm *mcfg.exe*
33	Falsche Betriebssystemsoftware. Die Dateiversionen der *mcug3.dll* und *rwmos.elf*-Files haben inkompatible Revisionsstände und können nicht aufeinander abgestimmt werden.
34	Der Gerätetreiber rnwmc.sys (Windows NT 4.0, 2000) oder rnwmc.vxd (Windows 95/98/Me) konnte nicht geöffnet werden.
35	Fehler beim Einmappen des physischen MCU-G3-Speichers.
36	Fehler beim Einmappen des physischen MCU-G3-Speichers.
37	Fehler beim Ausmappen des physischen MCU-G3-Speichers.
38	Zugriff auf die MCU-G3 nicht möglich.
39	Kein Zugriff auf MCU-G3 Mail-Box-Interface möglich.
lderr	Fehler-Rückgabewert von PCAP-Befehl txbf2()

4.4.17 InitMcuSystem2, initialise mcu system (2nd method)

BESCHREIBUNG:	Diese Funktion hat die gleiche Bedeutung wie *InitMcuSystem()*, mit der Ausnahme, dass die Parameter *SystemFileName* und *TpuBaseAddress* spezifiziert werden. Dabei enthält *SystemFileName* den Dateinamen der Systemdatei (normalerweise system.dat) inkl. Pfad- und Laufwerksangabe.
BORLAND DELPHI:	function InitMcuSystem2(var tsrp:TSRP; TpuBaseAddress: integer, var SystemFileName: string):integer;
C:	int InitMcuSystem2(struct TSRP *tsrp, int TpuBaseAddress, char *SystemFileName)
VISUAL BASIC:	Function InitMcuSystem2(DTSRP As TSRP, ByVal TpuBaseAddress As Long, ByVal filename As String) As Long
RÜCKGABEWERT:	Die Funktion hat die gleichen Rückgabewerte wie die Funktion *InitMcuSystem()*. Weitere, dort nicht beschriebene Rückgabewerte können von der implizit aufgerufenen Funktion txbf2 zurückgegeben werden.
ANMERKUNG:	siehe *InitMcuSystem(),* TpuBaseAddress hat keine Bedeutung und sollte mit dem Wert 0 übergeben werden.

4.4.18 InitMcuSystem3, initialise mcu system (3rd method)

BESCHREIBUNG:	Diese Funktion hat die gleiche Bedeutung wie *InitMcuSystem()*, mit der Ausnahme, dass die Parameter *SystemFileName, rosi, TpuBaseAddress* und *BoardType* spezifiziert werden. Dabei enthält *SystemFileName* den Dateinamen der Systemdatei (normalerweise system.dat) inkl. Pfad und Laufwerksangabe.
BORLAND DELPHI:	function InitMcuSystem3(var tsrp:TSRP; var rosi:ROSI, TpuBaseAddress: integer, var SystemFileName: string; var BoardType: integer):integer;
C:	int InitMcuSystem3(struct TSRP *tsrp, struct ROSI *rosi, int TpuBaseAddress, char *SystemFileName, int *BoardType)
VISUAL BASIC:	Function InitMcuSystem3(DTSRP As TSRP, DROSI As ROSI, ByVal TpuBaseAddress As Long, ByVal filename As String, BoardType As Long) As Long
RÜCKGABEWERT:	Die Funktion hat die gleichen Rückgabewerte wie die Funktion *InitMcuSystem()*.Weitere, dort nicht beschriebene Rückgabewerte können von der implizit aufgerufenen Funktion txbf2 zurückgegeben werden. Zusätzlich wird die Struktur *rosi* anhand der von der Steuerung zurückgelieferten Systeminformationen aktualisiert. Der Wert von *BoardType* gibt Aufschluss über den Steuerungstyp. *BoardType* kannfolgende Werten enthalten: 1 = MCU-3T oder PA-8000 (ISA Gerät) 2 = MCU-6 oder PA-840 (ISA-Gerät) 4 = MCU-3000 oder APCI-8001 8 = MCU-6000 oder APCI-8401 16 (10 hex) = MCU-3400C oder CPCI-8004 32 (20 hex) = MCU-3100 oder APCI-8008 0 = unbekanntes Board oder RWMOS veraltet andere Werte = neuere Produkte
ANMERKUNG:	siehe *InitMcuSystem()* TpuBaseAddress hat keine Bedeutung und sollte mit dem Wert 0 übergeben werden. Da diese Initialisierungsfunktion derzeit die grösste Funktionalität aufweist wird die Verwendung dieser Funktion empfohlen.

4.4.19 ja, jog absolute

BESCHREIBUNG:	Die in *AS* gewählten Achskanäle werden auf die in *TSRP[n].tp* angegebenen Zielpositionen mit Hilfe eines Trapez-Drehzahl-Profils absolut verfahren. Zur Profilgenerierung werden die achsspezifischen Systemparameter *jac* (*jog*-Beschleunigung), *jvl* (*jog*-Geschwindigkeit) und *jtvl* (*jog*-Zielgeschwindigkeit) herangezogen. Diese Parameter können mit Hilfe von Schreib- und Lese-Befehlen jederzeit gesetzt und abgefragt werden. Die Defaultwerte werden im Hilfsprogramm *mcfg.exe* spezifiziert. Die Bahnparameter werden in den in *mcfg.exe* festgelegten achsspezifischen Einheiten (Weg, Zeit) angegeben.
BORLAND DELPHI:	procedure ja(var as:AS; var tsrp:TSRP);
C:	void ja(struct AS far *as, struct TSRP far *tsrp);
VISUAL BASIC:	Sub ja(DASEL As ASEL, DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].tp n = 0 .. Anzahl der vorhandenen Achsen-1
ANMERKUNG:	Sofern dieser Befehl gleichzeitig für mehrere Achsen ausgeführt wird, können diese aufgrund der achsspezifischen Systemparameter zu unterschiedlichen Zeitpunkten die Zielpositionen erreichen [Kapitel 2.2.7] Die achsspezifischen Parameter wie z.B. Beschleunigungen und Geschwindigkeiten können jederzeit mit Hilfe von Lese- und Schreibbefehlen abgefragt bzw. gesetzt werden. Diese werden nicht automatisch mit ja übertragen. **Hinweis:** Beim Aufruf von ja muss immer das Element 0 der globalen Datenstruktur TSRP angegeben werden, da ja() sich den Index der verwendeten TSRP-Strukturelemente aus der AS-Struktur entnimmt.

4.4.20 jhi, jog home index

BESCHREIBUNG:	Mit Hilfe dieses Befehls wird der Indexsuchlauf aller in *AS* angewählten Achskanäle gestartet. Der Suchlauf wird entweder beim Aktivieren des Index- (Nullspur) Signals vom Inkrementalenkoder oder nach Überschreiten der in *tp* spezifizierten Weg- bzw. Winkelangabe beendet. Der Suchlauf wird mit Hilfe eines Trapez-Drehzahl-Profils durchgeführt. Die Parameter für den Profilgenerator sind dabei die Systemdaten *hac* und *hvl*, welche mit Hilfe von *mcfg.exe* bzw. den entsprechenden Schreibbefehlen gesetzt werden können. Beim Erkennen des Indexsignals (Nullspur), wird der Motor mit der Beschleunigung *hac* auf Geschwindigkeit 0 abgebremst. Der Parameter *tp* wird als relativer Verfahrweg in der achsspezifischen Positionseinheit angegeben. Die Suchrichtung wird durch das Vorzeichen von *tp* bestimmt. Im allgemeinen wird das Achssystem zuerst auf einen Referenzschalter (Nocken) gefahren. Um die mechanische Ungenauigkeit dieses Nockens zu eliminieren, bietet es sich an, im Anschluss den Index-Suchlauf durchzuführen. **MCU-3000 / APCI-8001:** **MCU-3100 / APCI-8008:** **MCU-3400C/CPCI-8004:** Die Befehlsausführung kann mit Hilfe des Profil-Ende-Flags (PE) im *axst*-Register und der Zustand des Index-Signals mit dem *digi*-Register [Kapitel 4.4.60.1] abgefragt werden. Das Profil-Ende-Flag bleibt bis zum Ende des Suchlaufs auf 0 gesetzt. **MCU-6000 / APCI-8401:** Die Befehlsausführung kann mit Hilfe des Profil-Ende-Flags im *axst*-Register und der Zustand des Index-Signals mit dem *asmi*-Register [Kapitel 4.4.46.1] abgefragt werden. Das Profil-Ende-Flag bleibt bis zum Ende des Suchlaufs auf 0 gesetzt.
BORLAND DELPHI:	procedure jhi(var as:AS; var tsrp:TSRP);
C:	void jhi(struct AS far *as, struct TSRP far *tsrp);
VISUAL BASIC:	Sub jhi(DASEL As ASEL, TSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].tp n = 0 .. Anzahl der vorhandenen Achsen-1
ANMERKUNG	Um eine möglichst genaue Index-Positionierung zu realisieren, sollte der Suchlauf mit möglichst kleiner Verfahrgeschwindigkeit ausgeführt werden. Es gibt jedoch auch die Möglichkeit, den Suchlauf in zwei Schritten durchzuführen. Im ersten Schritt kann der Suchlauf z.B. in positive Verfahrrichtung mit relativ großer Suchgeschwindigkeit gestartet werden. Im zweiten Schritt wird der Suchlauf dann in die negative Richtung mit kleiner Suchgeschwindigkeit abgeschlossen. Die Suchgeschwindigkeit kann mit den PCAP-Befehlen *rdhvl()* und *wrhvl()* gelesen und geschrieben werden. **Hinweis:** Beim Aufruf von jhi muss immer das Element 0 von TSRP angegeben werden, da jhi() sich den Index der verwendeten TSRP-Strukturelemente aus der AS-Struktur entnimmt.

4.4.21 jhl, jog home left

BESCHREIBUNG:	Dieser Befehl startet den Referenzsuchlauf aller in *AS* spezifizierten Achskanäle in negative Verfahrrichtung. Der Suchlauf wird mit Hilfe eines Endlos-Trapez-Drehzahlprofils ausgeführt. Die achsspezifischen Systemdaten *hac* und *hvl* dienen dabei als Parameter zur Profilgenerierung. Sofern ein mit REF-Funktion projektierter Digital-Eingang der MCU-G3 bei dem gewählten Achskanal aktiviert wird, wird der Suchlauf durch Abbremsen (mit *hac*) der Achse auf Geschwindigkeit 0 beendet. Dieser Zustand kann mit Hilfe des *pe*-Profil-Flags im *axst*-Register abgefragt werden. Das Profilflag bleibt bis zum Ende des Suchlaufes auf 0 gesetzt.
BORLAND DELPHI:	procedure jhl(var as:AS);
C:	void jhl(struct AS far *as);
VISUAL BASIC:	Sub jhl(DASEL As ASEL)

4.4.22 jhr, jog home right

BESCHREIBUNG:	Die Funktionsweise dieses Befehls ist identisch mit dem PCAP-Befehl *jhl()*, jedoch wird der Suchlauf in die positive Verfahrrichtung gestartet.
BORLAND DELPHI:	procedure jhr(var as:AS);
C:	void jhr(struct AS far *as);
VISUAL BASIC:	Sub jhr(DASEL As ASEL)

4.4.23 jr, jog relative

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *ja()*, mit dem Unterschied, dass es sich bei der Wegangabe *tp* um einen relative (inkrementale) Verfahrstrecke handelt. Ausgehend von der momentanen Position wird der Motor um die angegebene Strecke (bzw. Winkel) nach links (negative Werte) bzw. rechts (positive Werte) verfahren.
BORLAND DELPHI:	procedure jr(var as: AS; var tsrp:TSRP);
C:	void jr(struct AS far *as, struct TSRP far *tsrp);
VISUAL BASIC:	Sub jr(DASEL As ASEL, DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].tp n = 0 .. Anzahl der vorhandenen Achsen-1
HINWEIS:	Beim Aufruf von jr muss immer das Element 0 von TSRP angegeben werden, da jr() sich den Index der verwendeten TSRP-Strukturelemente aus der AS-Struktur entnimmt.

4.4.24 js, jog stop

BESCHREIBUNG:	Die in AS gewählten Achskanäle werden mit der achsspezifischen Verzögerung *sdec* auf Geschwindigkeit 0 abgebremst und in Lageregelung gehalten. Bis zum Ende des Abbremsvorgangs ist das *pe*-Flag im *axst*-Register rückgesetzt. Die Verzögerung *sdec* kann mit Hilfe von Schreib- und Lese-Befehlen jederzeit gesetzt und abgefragt werden. Der Defaultwert wird im Hilfsprogramm *mcfg.exe* spezifiziert.
BORLAND DELPHI:	procedure js(var as: AS);
C:	void js(struct AS far *as);
VISUAL BASIC:	Sub js(DASEL As ASEL)
ANMERKUNG:	Sofern dieser Befehl gleichzeitig für mehrere Achsen ausgeführt wird, können diese aufgrund der achsspezifischen Systemparameter zu unterschiedlichen Zeitpunkten die Zielpositionen erreichen [Kapitel 2.2.7]. Mit dem Wert sdec = 0 wird ein sofortiger Stop der Achse ohne Bremsrampe erzwungen.

4.4.25 lpr – Latch Position Registers

BESCHREIBUNG:	Mit diesem Kommando kann die Aufzeichnung für die grafische Systemanalyse für eine Achse gestartet werden.
BORLAND DELPHI:	procedure lpr (var latch_infos: LATCH_INFOS);
C:	void lpr (struct LATCH_INFOS *latch_infos);
VISUAL BASIC:	Sub lpr (DLATCH_INFOS As LATCH_INFOS)
RÜCKGABEWERT:	keiner
WIRKUNG:	Nach Ausführung des Befehls lprs wird die Aufzeichnung für die grafische Systemanalyse gestartet. Die Parameter für die Aufzeichnung werden in latch_infos angegeben.
ANMERKUNG:	siehe auch Kommando lprs und grafische Systemanalyse in mcfg **Wichtig:** Die Datenstruktur latch_infos muss 4-byte-weise ausgerichtet sein.

4.4.26 lprs – Latch Position Registers Synchronous

BESCHREIBUNG:	Mit diesem Kommando kann die Aufzeichnung für die grafische Systemanalyse synchron für ein oder mehrere Achsen gestartet werden
BORLAND DELPHI:	procedure lprs (var as: AS; var latch_infos: LATCH_INFOS);
C:	void lprs (struct AS *as, struct LATCH_INFOS *latch_infos);
VISUAL BASIC:	Sub lprs (DASEL As ASEL, DLATCH_INFOS As LATCH_INFOS)
RÜCKGABEWERT:	keiner
WIRKUNG:	Nach Ausführung des Befehls lprs wird die Aufzeichnung für die grafische Systemanalyse gestartet. Die Parameter für die Aufzeichnung werden in latch_infos angegeben. Das Element *san* der Datenstruktur *latch_infos* hat bei diesem Kommando keine Bedeutung, da die Achsen in *as* spezifiziert werden.
ANMERKUNG:	siehe auch Kommando lpr und grafische Systemanalyse in mcfg **Wichtig:** Die Datenstruktur latch_infos muss 4-byte-weise ausgerichtet sein.

4.4.27 lps, latch position synchronous

BESCHREIBUNG:	Mit diesem Befehl kann ein Latch-Vorgang synchron zum Abtastzyklus des in *an* angewählten Achskanals ausgelöst werden. Nach dem Aufruf wird die Ist-Position *{rp}* nach jeweils *mst* Abtastintervallen zwischengespeichert. Sofern ein Latch-Vorgang stattgefunden hat, wird dies im *axst*-Register im Flag *lpsf* (Bit Nr. 16) angezeigt. Mit dem PCAP-Lesebefehl *rdlp()* oder dem SAP-Achsenqualifizierer *lp* kann die zwischengespeicherte Position ausgelesen werden. Das Auslesen löscht auch das *lpsf*-Flag im *axst*-Register.
BORLAND DELPHI:	procedure lps(an: integer; mst: integer);
C:	void lps(int an, int mst);
VISUAL BASIC:	Sub lps(ByVal an As Long, ByVal mst As Long)
ANMERKUNG:	Der Befehl wird hauptsächlich beim Aufzeichnen von Konturen und Teach-In-Anwendungen verwendet, da er das Aufzeichnen von Positionsdaten in Echtzeit von einer oder mehreren Achsen ermöglicht. Typische Werte für *mst* sind 10..100 Abtastintervalle (-> 12.8ms..128.0ms). Der genaue Wert hängt jedoch von der Verarbeitungsgeschwindigkeit der jeweiligen Applikation ab

4.4.28 mca, move circular absolute
smca, spool motion circular absolute

BESCHREIBUNG:	Dieser Befehl bewirkt die zirkulare Interpolation der ersten beiden in AS spezifizierten Achskanäle. Bezüglich der Achsenauswahl gibt es keine Einschränkungen. Die Kreisinterpolation wird auf Basis eines Trapez-Drehzahl-Profiles, d.h. unter Berücksichtigung von Maximalbeschleunigung und Maximalgeschwindigkeit, durchgeführt. Als Interpolationsparameter werden die in CMP spezifizierten Struktur- bzw. Recordkomponenten herangezogen. Dies sind die Bahnbeschleunigung *ac*, die Bahngeschwindigkeit *vl* und die Bahnzielgeschwindigkeit *tvl*. Die in *dtca1* und *dtca2* eingetragenen Koordinaten spezifizieren den Kreismittelpunkt im Absolutmaßsystem. Dabei wird *dtca1* der ersten in AS programmierten Achse und *dtca2* der zweiten in AS spezifizierten Achse zugeordnet. Die Einheiten der Bahnparameter werden mit dem PCAP-Befehl *ctru()* gewählt. Der Winkel *phi* spezifiziert den abzufahrenden Verfahrwinkel mit der Einheit Grad. Die Drehrichtung wird durch das Vorzeichen der Winkelgröße festgelegt. Positive Werte bedeuten, Drehrichtung im Gegenuhrzeigersinn und negative Werte Drehrichtung im Uhrzeigersinn. Der Verfahrwinkelbereich ist nicht auf bestimmte Grenzen fixiert, d.h. es können auch Teil- oder Vielfach-Kreise abgefahren werden.
BORLAND DELPHI:	procedure mca(var as: AS; var cmp: CMP); procedure smca(var as: AS; var cmp: CMP);
C:	void mca(struct AS far *as, struct CMP far *cmp); void smca(struct AS far *as, struct CMP far *cmp);
VISUAL BASIC:	Sub mca(DASEL As ASEL, CMP As CMP) Sub smca(DASEL As ASEL, CMP As CMP)
ANMERKUNG:	Kapitel 2.3 Interpolation mit der MCU-G3.

4.4.29 mcr, move circular relative
smcr, spool motion circular relative

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *mca()* bis auf den Unterschied, dass die in *dtca1* und *dtca2* spezifizierten Koordinaten inkremental, bzw. relativ, auf die aktuelle Motorpositionen bezogen werden.
BORLAND DELPHI:	procedure mcr(var as: AS; var cmp: CMP); procedure smcr(var as: AS; var cmp: CMP);
C:	void mcr(struct AS far *as, struct CMP far *cmp); void smcr(struct AS far *as, struct CMP far *cmp);
VISUAL BASIC:	Sub mcr(DASEL As ASEL, CMP As CMP)
ANMERKUNG:	Kapitel 2.3 Interpolation mit der MCU-G3.

4.4.30 mca3d, move circular absolute three dimensional
smca3d, spool motion circular absolute three dimensional

BESCHREIBUNG:	Dieser Befehl bewirkt die zirkulare Interpolation der drei spezifizierten Achskanäle. Bezüglich der Achsenauswahl gibt es keine Einschränkungen. Die Kreisinterpolation wird auf Basis eines Trapez-Drehzahl-Profiles, d.h. unter Berücksichtigung von Maximalbeschleunigung und Maximalgeschwindigkeit, durchgeführt. Als Interpolationsparameter werden die Bahnbeschleunigung *ac*, die Bahngeschwindigkeit *vl* und die Bahnzielgeschwindigkeit *tvl in hmp3d verwendet*. Die in *dtca1, dtca2 und dtca3* eingetragenen Koordinaten spezifizieren den Kreismittelpunkt im Absolutmaßsystem. Dabei wird *dtca1* der Ersten, dtca2 der Zweiten und dtca3 der dritten in AS spezifizierten Achse zugeordnet. Die Einheiten der Bahnparameter werden mit dem PCAP-Befehl *ctru()* gewählt. Der Kreis kann in einer beliebigen Ebene abgefahren werden, welche durch die Flächen-Normale in PN1, PN2 und PN3 spezifiziert wird. Die aktuellen Startkoordinaten liegen immer in der angegebenen Ebene. Der Winkel *phi* spezifiziert den abzufahrenden Verfahrwinkel mit der Einheit Grad. Die Drehrichtung wird durch das Vorzeichen der Winkelgröße festgelegt. Positive Werte bedeuten, Drehrichtung im Gegenuhrzeigersinn und negative Werte Drehrichtung im Uhrzeigersinn. Der Verfahrwinkelbereich ist nicht auf bestimmte Grenzen fixiert, d.h. es können auch Teil- oder Vielfach-Kreise abgefahren werden. Das Datenfeld dtm[] wird hier nicht verwendet.
BORLAND DELPHI:	procedure mca3d(var as: AS; var hmp3d: HMP3D); procedure smca3d(var as: AS; var hmp3d: HMP3D);
C:	void mca3d(struct AS far *as, struct HMP3D far *hmp3d); void smca3d(struct AS far *as, struct HMP3D far *hmp3d);
VISUAL BASIC:	Sub mca3d(DASEL As ASEL, HMP3D As HMP3D) Sub smca3d(DASEL As ASEL, HMP3D As HMP3D)
ANMERKUNG:	Kapitel 2.3 Interpolation mit der MCU-G3.

4.4.31 mcr3d, move circular relative three dimensional
smcr3d, spool motion circular relative three dimensional

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *mca3d()* bis auf den Unterschied, dass die in *dtca1, dtca2* und *dtca3* spezifizierten Koordinaten inkremental, bzw. relativ, auf die aktuelle Motorpositionen bezogen werden.
BORLAND DELPHI:	procedure mcr3d(var as: AS; var hmp3d: HMP3D); procedure smcr3d(var as: AS; var hmp3d: HMP3D);
C:	void mcr3d(struct AS far *as, struct HMP3D far *hmp3d); void smcr3d(struct AS far *as, struct HMP3D far *hmp3d);
VISUAL BASIC:	Sub mcr3d(DASEL As ASEL, HMP3D As HMP3D) Sub smcr3d(DASEL As ASEL, HMP3D As HMP3D)
ANMERKUNG:	Kapitel 2.3 Interpolation mit der MCU-G3.

4.4.32 mcuinit, motion control unit initialisation

BESCHREIBUNG:	Mit dieser Funktion werden verschiedene Initialisierungen innerhalb des Systemtreibers *mcug3.dll* durchgeführt. Es wird geprüft, ob eine Kommunikation zwischen PC und MCU-G3 möglich ist. Sofern dies der Fall ist, werden die von der MCU-G3 zurückgelieferten *rw_MOS*-spezifischen Systemdaten in die Struktur bzw. im Record ROSI eingetragen. Anhand von ROSI können die *rw_MOS*-spezifischen Systeminformationen auf Gültigkeit abgeprüft werden. Sofern der Kommunikationsaufbau zur MCU-G3 nicht möglich war, enthält die gesamte Struktur ROSI den Wert 0.
BORLAND DELPHI:	procedure mcuinit(var rosi:ROSI);
C:	void mcuinit(struct ROSI far *rosi);
VISUAL BASIC:	Sub mcuinit(DROSI As ROSI)
ANMERKUNG:	Dieser Befehl löst keinen Reset auf der MCU-G3 aus. Dies ist mit den PCAP-Befehlen *ra()* oder *rs()* durchzuführen. Mit dem Rückgabewert ROSI.sysfile_loaded kann festgestellt werden, ob die Systemdatei *system.dat* bereits mit Hilfe des PCAP-Ladebefehls *txbf2()* auf die MCU-G3 übertragen wurde. Ist dieser Wert 0, so muss nach erfolgreichem *mcuinit()*-PCAP-Befehl der PCAP-Befehl *txbf2()* ausgeführt werden, damit ein Arbeiten mit der MCU-G3 möglich ist. Bei den mitgelieferten PCAP-Beispielprogrammen ist dieser Befehl in den Funktion *InitMcuSystem(), InitMcuSystem2() und InitMcuSystem3()* enthalten. Dort wird der Kontrollmechanismus der Systeminitialisierung nochmals verdeutlicht. **Wichtig:** mcuinit() besteht aus Kompatibilität zu den MCU-G2-Controllern und sollte durch den InitMcuSystem3()-Befehl ersetzt werden. Lediglich zur Überprüfung ob die Steuerungsbaugruppe noch online ist kann dieser Befehl verwendet werden.

4.4.33 MCUG3_SetBoardIntRoutine

BESCHREIBUNG:	Mit Hilfe dieser Funktion kann eine benutzerspezifische Interruptbearbeitungsroutine installiert und aktiviert werden.
BORLAND DELPHI:	function MCUG3_SetBoardIntRoutine (func : Pointer): integer;
C:	int MCUG3_SetBoardIntRoutine(InterruptRoutine func);
VISUAL BASIC:	Function MCUG3_SetBoardIntRoutine (ByVal func As Long) As Long
PARAMETER:	func ist ein Funktionszeiger auf die vom Anwender geschriebene Interrupt-Bearbeitungsroutine. Diese wird z.B. in der folgenden Form deklariert (C++): void CALLBACK EventHandler(int IRQLineBits) {}
RÜCKGABEWERT:	keine Bedeutung
ANMERKUNG:	Innerhalb der der Interrupt Bearbeitungsroutine sind die Programmierkonventionen der Windows-Betriebssysteme zu beachten. So ist es z.B. nicht erlaubt, in einem Callback-Handler Fensterobjekte zu erzeugen. Für Visual Basic 6.0 ist das zusätzliche Modul „MCUG3Interrupt.BAS“ für die Verwendung dieser Funktion im Lieferumfang enthalten.

4.4.34 MCUG3_ResetBoardIntRoutine

BESCHREIBUNG:	Mit Hilfe dieser Funktion kann eine zuvor aktivierte benutzerspezifische Interruptbearbeitungsroutine deaktiviert werden.
BORLAND DELPHI:	function MCUG3_ResetBoardIntRoutine (): integer;
C:	int MCUG3_ResetBoardIntRoutine(void);
VISUAL BASIC:	Function MCUG3_ResetBoardIntRoutine () As Long
ANMERKUNG:	Vor dem Beenden der Applikation muß die aktuelle installierte Interrupt-Serviceroutine deinstalliert werden.

4.4.35 mha, move helical absolute
smha, spool motion helical absolute

BESCHREIBUNG:	Mit diesem Kommando wird eine Helix- oder Schraubenlinieninterpolation durchgeführt. Dieser Befehl ist eine Erweiterung der Zirkularinterpolation. Deshalb treffen die beim PCAP-Befehl *mca()* genannten Aussagen für dieses Kommando ebenfalls zu, mit dem Unterschied dass die Bahnparameter in der Struktur bzw. im Record HMP eingetragen werden. Für weitere, in AS spezifizierte Achsen kann zusätzlich der Parameter *dtm* programmiert werden. Dies sind die absoluten Zielpositionen für weitere Achsen. Während die ersten beiden Achsen eine Zirkularinterpolation durchführen, werden weitere Achsen linear verfahren. Alle Achsen erreichen zum gleichen Zeitpunkt ihre Zielpositionen. Im Unterschied zur Zirkularinterpolation kann der Kreiszielpunkt anstatt über den Kreiswinkel per Zielposition definiert werden. Dieser Fall muss vom Benutzer mit einem Betrag des Verfahrwinkels <= 1e-100 angezeigt werden. Das Vorzeichen dieses Winkels gibt die Verfahrrichtung an. Die gewünschten Kreiszielpunkte werden in diesem Fall in dtm [0] und dtm[1] von HMP angegeben. Falls der angegebene Zielpunkt nicht auf dem Kreis liegt, der sich aus Startpunkt und Mittelpunkt ergibt, wird die Zielposition korrigiert.
BORLAND DELPHI:	procedure mha(var as: AS; var hmp: HMP); procedure smca(var as: AS; var hmp: HMP);

C:	void mha(struct AS far *as, struct HMP far *hmp); void smha(struct AS far *as, struct HMP far *hmp);
VISUAL BASIC:	Sub mha(DASEL As ASEL, HMP As HMP) Sub smha(DASEL As ASEL, HMP As HMP)

4.4.36 mhr, move helical relative
smhr, spool motion helical relative

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *mha()* bis auf den Unterschied, dass die in *dtca1*, *dtca2* und *dtm* programmierten Wegangaben inkrementell, bzw. relativ, auf die momentanen Motorpositionen bezogen werden.
BORLAND DELPHI:	procedure mhr(var as: AS; var hmp: HMP); procedure smhr(var as: AS; var hmp: HMP);
C:	void mhr(struct AS far *as, struct HMP far *hmp); void smhr(struct AS far *as, struct HMP far *hmp);
VISUAL BASIC:	Sub mhr(DASEL As ASEL, HMP As HMP) Sub smhr(DASEL As ASEL, HMP As HMP)

4.4.37 mla, move linear absolute
smla, spool motion linear absolute

BESCHREIBUNG:	Mit diesem Befehl wird eine Linear- oder Geradeinterpolation mit absoluten Zielangaben durchgeführt. Zur Interpolation sind alle Achsen im n-dimensionalen Raum zulässig. Welche Achsen an der Interpolation teilnehmen sollen, wird in AS spezifiziert. Mit Hilfe von LMP werden die Bahnbeschleunigung *ac*, Bahngeschwindigkeit *vl* und Bahnzielgeschwindigkeit *tvl* für die Linearinterpolation festgelegt. Die Einheiten der Bahnparameter werden mit dem Befehl *ctru()* gewählt. Je nach Anzahl der Achsen *unoa* werden die gewünschten Achsen im Feld *san* und die entsprechenden Verfahrwege im Feld *dtm* eingetragen. Dabei wird der Verfahrweg im Feld *dtm[n]* der Achsennummer *n+1* zugeordnet. Die Interpolation bezieht sich auf die in AS eingetragenen Achsen. Die Verfahrwege werden als absolute, also auf den Maschinennullpunkt bezogene, Weg- bzw. Winkelinformationen interpretiert.
BORLAND DELPHI:	procedure mla(var as: AS; var lmp: LMP); procedure smla(var as: AS; var lmp: LMP);
C:	void mla(struct AS far *as, struct LMP far *lmp); void smla(struct AS far *as, struct LMP far *lmp);
VISUAL BASIC:	Sub mla(DASEL As ASEL, lmp As lmp) Sub smla(DASEL As ASEL, lmp As lmp)
ANMERKUNG:	Kapitel 2.3 Interpolation mit der MCU-G3.

4.4.38 mlr, move linear relative
smlr, spool motion linear relative

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *mla()*, jedoch werden die im Feld *dtm* spezifizierten Verfahrwege inkremental, bzw. relativ zur momentanen Motorposition, interpretiert.
BORLAND DELPHI:	procedure mlr(var as: AS; var lmp: LMP); procedure smlr(var as: AS; var lmp: LMP);
C:	void mlr(struct AS far *as, struct LMP far *lmp); void smlr(struct AS far *as, struct LMP far *lmp);
VISUAL BASIC:	Sub mlr(DASEL As ASEL, lmp As lmp) Sub smlr(DASEL As ASEL, lmp As lmp)
ANMERKUNG:	Kapitel 2.3 Interpolation mit der MCU-G3.

4.4.39 ms, motion stop

BESCHREIBUNG:	Die in *AS* gewählten Achskanäle werden mit der zur Zeit gültigen Bahnbeschleunigung bzw. Achsenverzögerung auf Geschwindigkeit 0 abgebremst und in Lageregelung gehalten. Bis zum Ende des Abbremsvorgangs ist das *pe*-Flag im *axst*-Register rückgesetzt. Der Richtungsvektor einer evtl. gerade ablaufenden Interpolation wird durch diesen Befehl nicht verändert. Falls die angewählten Achsen gerade einen Kreis abfahren, erfolgt die Abbremsung auf der Kreisbahn mit der angegebenen Bahnbeschleunigung. Achsen, die mit einer Endgeschwindigkeit verfahren, werden mit der achsspezifischen Verzögerung *sdec* auf Geschwindigkeit 0 abgebremst.
BORLAND DELPHI:	procedure ms(var as: AS);
C:	void ms(struct AS far *as);
VISUAL BASIC:	Sub ms(DASEL As ASEL)
ANMERKUNG:	Nicht gemeinsam interpolierende Achsen können den Zielpunkt zu unterschiedlichen Zeitpunkten erreichen.

4.4.40 MsgToScreen

BESCHREIBUNG:	Mit diesem Befehl ist es möglich, Bildschirmmeldungen des DLL-Treibers zu sperren oder zu aktivieren. Ist der Parameter Enable = 0, so werden Bildschirmmeldungen unterdrückt.
BORLAND DELPHI:	procedure MsgToScreen (Enable: integer);
C:	void MsgToScreen (long Enable);
VISUAL BASIC:	Sub MsgToScreen (ByVal Enable As Long)
ANMERKUNG:	Diese Option ist wichtig bei Systemen ohne Benutzerinterface. Wenn Bildschirmmeldungen freigegeben sind, kann das System ansonsten auf eine Eingabe warten, die nicht bedient werden kann. Dieses Kommando ist ab Version 3.5.2.10 verfügbar.

4.4.41 ol, open loop

BESCHREIBUNG:	Dieser Befehl öffnet den Lageregelkreis aller in *AS* angewählten Achsen. Auf den Motor-Command-Ports wird je bei Servo-Achsen 0V Ausgangsspannung und bei Schrittmotorachsen 0Hz Schrittfrequenz ausgegeben. Alle mit PAE-Funktion projektierten MCU-G3 Digitalausgänge werden für die programmierten Achskanäle inaktiv gesetzt. Je nach selektiertem Achskanal werden die Relais K2 (Achskanal 1), K3 (Achskanal 2) und K4 (Achskanal 3) abgeschaltet [IHB / Kapitel 5.2.10].
BORLAND DELPHI:	procedure ol(var as: AS);
C:	void ol(struct AS far *as);
VISUAL BASIC:	Sub ol(DASEL As ASEL)
ANMERKUNG:	Dieser Befehl wird hauptsächlich in Ausnahmesituationen wie Endschalterbegrenzung, Schleppfehlerüberschreitung usw. verwendet.

4.4.42 ra, reset axis

BESCHREIBUNG:	Mit diesem Befehl kann ein achsspezifischer Rücksetzvorgang durchgeführt werden. Dieser bewirkt den Abbruch eines evtl. ablaufenden Profils, das Öffnen des Lageregelkreises, die Sollwertabschaltung, das Verwerfen von evtl. vorhanden Spoolerdaten und das Nullsetzen der Positionsregister. Die Ausgänge werden auf die projektierten Default-Werte gesetzt. Die achsspezifischen Overridefaktoren (PCAP-Befehl *wrjovr()* und *wrtrovr()*) werden auf den Wert 1.0 gesetzt. Die evt. projektierten Softwareendlagen werden für die in *ra()* angewählten Achskanäle nicht mehr überwacht.
BORLAND DELPHI:	procedure ra(var as: AS);
C:	void ra(struct AS far *as);
VISUAL BASIC:	Sub ra(DASEL As ASEL)
ANMERKUNG:	Alle Systemdaten wie Beschleunigungen, Geschwindigkeiten, Filterparameter usw. bleiben gespeichert und brauchen deshalb nicht neu geladen zu werden. Dieser Befehl wird hauptsächlich bei der Systeminitialisierung bzw. in Ausnahmesituationen verwendet. **Vorsicht:** Eventuell gesetzte PAE-Ausgänge anderer Achsen in der gleichen Ausgangsgruppe werden mit diesem Befehl zurückgesetzt.

4.4.43 rasms, reset ASM-2003 status register

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401

BESCHREIBUNG:	Mit diesem Befehl können verschiedene Fehlerflags im ASM-2003-Status-Register *asms* rückgesetzt werden. Im Detail sind dies die Fehlerbits 0, 1, 2, 3, 7 - *lerrtotx*, *lerrtotx*, *lerrd, reserr* und *dc*. Das Rücksetzen sollte nur in Ausnahmesituationen, z.B. in einer Fehlerüberwachungsroutine, ausgeführt werden.
BORLAND DELPHI:	procedure rasms(var as: AS);
C:	void rasms(struct AS far *as);
VISUAL BASIC:	Sub rasms(DTSRP As TSRP)
ANMERKUNG:	Bei verschiedenen ASM-2003-Firmware-Revisionen wird auch die rote LED [IHB / Kapitel 5.5] zurückgesetzt.

4.4.44 rdap, read axis parameters

BESCHREIBUNG:	Mit diesem Befehl können alle achsspezifischen Ein- und Ausgangsgrößen der Struktur bzw. des Records TSRP mit einem Lesebefehl eingelesen werden.
BORLAND DELPHI:	procedure rdap(var tsrp:TSRP);
C:	void rdap(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdap(DTSRP As TSRP)
TSRP-KOMPONENTEN:	alle, d.h. TSRP[n].an .. TSRP[n].ifs
RÜCKGABEWERT:	Nach Ausführung des Befehls stehen die Ein- und Ausgangsgrößen in den jeweiligen Struktur- bzw. Record-Komponenten der Struktur bzw. dem Record TSRP.
ANMERKUNG:	Die einzelnen Struktur- bzw. Record-Komponenten können auch mit speziellen Lesebefehlen abgefragt werden. Im Normalfall werden diese Lesebefehle wegen der kürzeren Zugriffszeit bevorzugt.

4.4.45 rdasmepc, read ASM-2003 EEPROM programming cycle

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401

BESCHREIBUNG:	Mit dieser Funktion kann die momentane Anzahl der ASM-2003 EEPROM Programmierzyklen gelesen werden. Die Zyklusnummer wird bei jedem Speichervorgrang im TOOLSET Programm *mcfg.exe* im EEPROM um eins erhöht. Das EEPROM läßt sich mindestens 10000 mal beschreiben.
BORLAND DELPHI:	procedure rdasmepc(var tsrp:TSRP);
C:	void rdasmepc(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdasmepc (DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].asmepc
RÜCKGABEWERT:	Die momentane Programmier-Zyklusnummer befindet sich nach Ausführung dieses Befehls in der Struktur- bzw. Record-Komponente *asmepc*.

4.4.46 rdasmi, read ASM-2003 inputs

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401

BESCHREIBUNG:	Mit dieser Funktion kann der aktuelle Zustand der ASM-2003 Digital-Eingänge und das Nullspur- (Index) Signal vom Inkrementalenkoder eingelesen werden. Zu beachten ist, daß die Digitaleingänge auf dem ASM-2003 achsspezifisch gruppiert sind. Sofern ein Eingang aktiv ist, wird dies mit dem Wert 1 des jeweiligen Bit angezeigt. Optional können alle Digitaleingänge im TOOLSET Programm *mcfg.exe* mit Invertierung projektiert werden. Ebenso ist es möglich, bei Verwendung eines Inkrementalenkoders mit Index-Signal die gewünschte Polarität zu projektieren.
BORLAND DELPHI:	procedure rdasmi(var tsrp:TSRP);
C:	void rdasmi(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdasmi (DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].asmi n = 0 .. Anzahl der Achsen -1
RÜCKGABEWERT:	Der bitkodierte Rückgabewert befindet sich in der Struktur- bzw. Recordkomponente *asmi* und hat den in nachfolgend abgedruckter Tabelle 12 beschriebenen Aufbau.
ANMERKUNG:	Diese Anweisung wird nur bei der MCU-6000 / APCI-8401 verwendet.

4.4.46.1 Achsenqualifizierer asmi

Nachfolgender Abschnitt gilt nur für die **MCU-6000 / APCI-8401**.
Mit dem Register *asmi* kann der Zustand der Digital-Eingänge des ASM-2003 abgeprüft werden. Sofern die jeweiligen Eingänge aktiv sind, wird dies mit dem Wert 1 an der jeweiligen Bitposition angezeigt.

Tabelle 12: Bitkodierter Aufbau des asmi-Wortes

Bit-Nr.	Achskanal 1, 3, 5 ... Funktion Stecker-PIN Reihe/Reihe (Name/Name)	Achskanal 2, 4, 6 ... Funktion Stecker-PIN Reihe/Reihe (Name/Name)
0	Eingang X1-16 D/B (I11+, I11-)	Eingang X2-06 D/B (I21+, I21-)
1	Eingang X1-18 D/B (I12+, I12-)	Eingang X2-08 D/B (I22+, I22-)
2	Eingang X1-20 D/B (I13+, I13-)	Eingang X2-10 D/B (I23+, I23-)
3	Eingang X1-22 D/B (I14+, I14-)	Eingang X2-12 D/B (I24+, I24-)
4	Eingang X1-24 D/B (I15+, I15-)	Eingang X2-14 D/B (I25+, I25-)
5	Eingang X1-26 D/B (I16+, I16-)	Eingang X2-16 D/B (I26+, I26-)
6	Eingang X2-02 D/B (I17+, I17-)	Eingang X2-18 D/B (I27+, I27-)
7	Eingang X2-04 D/B (I18+, I18-)	Eingang X2-20 D/B (I28+, I28-)
8	Null-Spur vom Inkrementalenkoder Kanal 1, 3, 5 ...	Null-Spur vom Inkrementalenkoder Kanal 2, 4, 6 ...
9..31	Nicht belegt, diese Flags haben einen undefinierten Wert und sind für zukünftige Verwendung reserviert.	

4.4.47 rdasmib, read ASM-2003 input bit

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401

BESCHREIBUNG:	Mit dieser Funktion kann der aktuelle Zustand eines ASM-2003 Digital-Eingangs bzw. des Nullspur- (Index) Signals vom Inkrementalenkoder abgefragt werden. Die Achsnummer muß im Parameter *an* (0, 1, ... *REALAXIS*) spezifiziert werden. Die abzufragende Bitnummer wird in *bitnr* (1..32) angegeben.
BORLAND DELPHI:	function rdasmib(an:integer; bitnr:integer):integer;
C:	int rdasmib(int an, int bitnr);
VISUAL BASIC:	Function rdasmib (ByVal an As Long, ByVal bitnr As Long) As Long
RÜCKGABEWERT:	Die Funktion liefert den Wert 1 bzw. TRUE zurück, sofern der entsprechende Eingang von *bitnr* aktiv ist.
ANMERKUNG:	Siehe auch PCAP-Befehl *rdasmi().* Diese Anweisung wird nur bei der MCU-6000 / APCI-8401 verwendet. **Vorsicht:** Die Zählweise der Bitnummer beginnt bei 1.

Tabelle 13: Zuordnung von *bitnr* zu den jeweiligen ASM-2003 Digitaleingängen

'bitnr'	Achskanal 1, 3, 5 ... Funktion Stecker-PIN Reihe/Reihe (Name/Name)	Achskanal 2, 4, 6 ... Funktion Stecker-PIN Reihe/Reihe (Name/Name)
1	Eingang X1-16 D/B (I11+, I11-)	Eingang X2-06 D/B (I21+, I21-)
2	Eingang X1-18 D/B (I12+, I12-)	Eingang X2-08 D/B (I22+, I22-)
3	Eingang X1-20 D/B (I13+, I13-)	Eingang X2-10 D/B (I23+, I23-)
4	Eingang X1-22 D/B (I14+, I14-)	Eingang X2-12 D/B (I24+, I24-)
5	Eingang X1-24 D/B (I15+, I15-)	Eingang X2-14 D/B (I25+, I25-)
6	Eingang X1-26 D/B (I16+, I16-)	Eingang X2-16 D/B (I26+, I26-)
7	Eingang X2-02 D/B (I17+, I17-)	Eingang X2-18 D/B (I27+, I27-)
8	Eingang X2-04 D/B (I18+, I18-)	Eingang X2-20 D/B (I28+, I28-)
9	Null-Spur vom Inkrementalenkoder Kanal 1, 3, 5 ...	Null-Spur vom Inkrementalenkoder Kanal 2, 4, 6 ...
10..32	Nicht belegt, diese Flags haben einen undefinierten den Wert und sind für zukünftige Verwendung reserviert.	

4.4.48 rdasmo, read ASM-2003 outputs

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401

BESCHREIBUNG:	Mit diesem Befehl wird der aktuelle Ausgabe-Status der ASM-2003-Digital-Ausgänge in die achsspezifische Struktur- bzw. Record-Komponente *.asmo* eingelesen. Die dort gesetzten Bits repräsentieren gesetzte Ausgänge.
BORLAND DELPHI:	procedure rdasmo(var tsrp:TSRP);
C:	void rdasmo(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdasmo (DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].asmo
RÜCKGABEWERT:	Die bitkodierten Rückgabewerte befinden sich nach Ausführung dieses Befehls in der Struktur- bzw. Recordkomponente *asmo*. Diese Komponente hat den beim PCAP-Befehl *wrasmo()* abgedruckten Aufbau.

4.4.49 rdasmob, read ASM-2003 output bit

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401

BESCHREIBUNG:	Mit dieser Funktion kann der aktuelle Zustand eines ASM-2003 Digital-Ausgangs abgefragt werden. Die Achsnummer muß im Parameter *an* (0, 1, ... *REALAXIS*) spezifiziert werden.
BORLAND DELPHI:	function rdasmob(an:integer; bitnr:integer):integer;
C:	int rdasmob(int an, int bitnr);
VISUAL BASIC:	Function rdasmob (ByVal an As Long, ByVal bitnr As Long) As Long
RÜCKGABEWERT:	Die Funktion liefert den Wert 1 bzw. TRUE zurück, sofern der entsprechende Ausgang von *bitnr* aktiv ist. Die Zuordnung von *bitnr* zu den jeweiligen Ausgängen ist beim PCAP-Befehl *wrasmob()* abgedruckt.

4.4.50 rdasms, read ASM-2003 status

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401	
BESCHREIBUNG:	Mit diesem Register können Statusinformationen des ASM-2003 eingelesen werden.
BORLAND DELPHI:	procedure rdasms(var tsrp:TSRP);
C:	void rdasms(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdasms (DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].asms
RÜCKGABEWERT:	Der bitkodierte Rückgabewert befindet sich in der Struktur- bzw. Recordkomponente *asms* und hat den in nachfolgend abgedruckter Tabelle 14 beschriebenen Aufbau.
ANMERKUNG:	Diese Anweisung wird nur bei der MCU-6000 / APCI-8401 verwendet.

4.4.50.1 Achsenqualifizierer asms

Nachfolgender Abschnitt gilt nur für die **MCU-6000 / APCI-8401**.
Mit diesem Register können verschiedene Status-Informationen des ASM-2003 abgefragt werden. Sofern die jeweilige Status-Information gültig ist, wird dies mit dem Wert 1 an der jeweiligen Bitposition angezeigt.

Tabelle 14: Bitkodierter Aufbau des asms-Wortes

Bit-Nr.	Funktion
0	*lerrtotx*: Sende-Timeoutfehler. Das Anschaltmodul ASM-2003 konnte nicht innerhalb einer festgelegten Mindestzeit Daten an die MCU-6000 / APCI-8401 übertragen. Dieses Bit bleibt nach einem Fehler erhalten, bis es per Befehl rasms (siehe Kapitel 4.4.43) oder durch einen Systemreset gelöscht wird.
1	*lerrtorx*: Empfangs-Timeoutfehler. Das Anschaltmodul ASM-2003 wurde nicht innerhalb einer festgelegten Mindestzeit von der MCU-6000 / APCI-8401 adressiert. Dieses Bit bleibt nach einem Fehler erhalten, bis es per Befehl rasms (siehe Kapitel 4.4.43) oder durch einen Systemreset gelöscht wird.
2	lerrd: Daten-Übertragungsfehler. Das von der MCU-6000 / APCI-8401 an das ASM-2003 übertragene Datenpaket wurde nich korrekt übertragen. Dieses Bit bleibt nach einem Fehler erhalten, bis es per Befehl rasms (siehe Kapitel 4.4.43) oder durch einen Systemreset gelöscht wird.
3	*reserr*: nach hardwaremässigem Rücksetzen der ASM-2003 Systemelektronik. Das Rücksetzen erfolgt durch spezielle Befehle, mit Hilfe des Reset- Tasters oder durch Aktivieren des Reset-Eingangs. In diesem Fall wird dieses Flag auf 1 gesetzt. Die ASM-2003 Watchdoglogik kann im Fehlerfall ebenfalls einen Hardware-Rücksetzvorgang bewirken, welches wiederum das Setzen dieses Statusflags bewirkt.
4	*edv*: Die im EEPROM abgelegten Systeminformationen und Daten sind gültig.
5	ope: Das ope-Flag zeigt, daß ein System-Fehler auf dem Options-Print der Anschaltbaugruppe erkannt wurde. Derzeit wird der Fehler nur von der SSI- und OPMF-3001-Optionskarte erzeugt.
6	*options*: Sofern die Firmware der ASM-2003 Baugruppe Optionen unterstützt, ist das Flag options permanent auf 1 gesetzt. Folgende Optionen werden unterstützt: Echtzeitlatchen, SSI-Drehgeberverarbeitung und A/D-Umsetzung.
7	*dc*: Das dc-Flag (disconnect), zeigt an, daß die Kommunikation zwischen MCU-6000 / APCI-8401 und ASM-2003-Anschaltbaugruppe für mehr als 50 Abtastintervalle (i. A. 64ms) ausgefallen war. In diesem Fall durchläuft die *rw_MOS*-Firmware eine interne Reset-Prozedur. Hierbei werden u.a. die anstehenden Bewegungskommandos der betroffenen Achskanäle verworfen und deren Lagelregelkreise geöffnet. Dieses Bit bleibt nach einem Fehler erhalten, bis es per Befehl rasms (siehe Kapitel 4.4.43) oder durch einen Systemreset gelöscht wird. **Anmerkung:** Solange das Bit dc gesetzt ist, kann der Regelkreis dieser Achse nicht geschlossen werden (Kommando cl).
8..31	Nicht belegt, diese Flags haben einen undefinierten Wert und sind für zukünftige Verwendung reserviert.

Anmerkungen: Nach dem zweiten Auftreten eines Fehlers vom Typ *lerrtotx*, *lerrtorx* oder *lerrd* wird der Übertragungsfehler zusätzlich durch die rote LED auf dem ASM-2003 signalisiert.
Das Statusflag *reserr* kann wie folgt gelöscht werden: mit dem Kommando rasms bzw. durch Beschreiben des Achsenqualifizierers asms in der rwSymPas Programmierumgebung oder durch abschalten des ASM-2003, einige Sekunden warten (versch. Spannungsversorgungen haben große Speicherkapzität) und Wiedereinschalten.

4.4.51 rdasmsb, read ASM-2003 status bit

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401	
BESCHREIBUNG:	Mit dieser Funktion kann eine ASM-2003 Statusinformation abgefragt werden. Die Achsnummer muß im Parameter *an* (0, 1, ... *REALAXIS*) spezifiziert werden.
BORLAND DELPHI:	function rdasmsb(an:integer; bitnr:integer):integer;
C:	int rdasmsb(int an, int bitnr);
VISUAL BASIC:	Function rasmsb (ByVal an As Long, ByVal bitnr As Long) As Long
RÜCKGABEWERT:	Die Funktion liefert den Wert 1 bzw. TRUE zurück, sofern der entsprechende Eingang von *bitnr* aktiv ist. Die Zuordnung von *bitnr* zu den jeweiligen Statusinformationen wird in Tabelle 14 beschrieben, jedoch erfolgt bei *bitnr* die Zählweise bei dem Wert 1, d.h. um beispielsweise *lerrtotx* abzufragen, muß *bitnr* den Wert 1 haben!
ANMERKUNG:	Siehe auch PCAP-Befehl *rdasms()* Diese Anweisung wird nur bei der MCU-6000 / APCI-8401 verwendet.

4.4.52 rdaux, read auxiliary register

BESCHREIBUNG:	Diese Funktion liefert das achsspezifische auxiliary Register zurück. [Kapitel 6.3.3]
BORLAND DELPHI:	procedure rdaux (var tsrp:TSRP);
C:	void rdaux (struct TSRP *tsrp);
VISUAL BASIC:	Sub rdaux(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].aux
ANMERKUNG:	siehe auch Kapitel 4.4.143

4.4.53 rdaxst, read axis status

BESCHREIBUNG:	Mit diesem Befehl können verschiedene Status- und Errorflags der Rampen- und Interpolationstask achsspezifisch abgefragt werden. Normalerweise wird dieser Befehl im PCAP-Programm zyklisch wiederholt, um mit dem nachfolgend beschriebenen *pe*-Flag abzuprüfen, ob die Verfahrkommandos der beteiligten Achsen fertig abgearbeitet wurden. Zusätzlich werden mit diesem Befehl eine Reihe von Fehlerflags im *axst*-Register aktualisiert. Diese sollten ebenfalls zyklisch ausgewertet werden, um ein sicheres Betriebsverhalten durch das PCAP-Programm zu garantieren.
BORLAND DELPHI:	procedure rdaxst(var tsrp:TSRP);
C:	void rdaxst(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdaxst(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].axst
RÜCKGABEWERT:	Der bitkodierte Rückgabewert befindet sich nach Ausführung dieses Befehls in der Struktur- bzw. Recordkomponente *axst* und hat den in nachfolgend abgedruckter Tabelle 15 beschriebenen Aufbau.

Tabelle 15: Bitkodierter Aufbau des axst-Wortes

Bit-Nr.	Name	Funktion
0 **0000 0001**	-	**MCU-3000 / MCU-3100 / MCU-3400C:** Nicht belegt, dieses Flag hat einen undefinierten Wert.
0 **0000 0001**	***toasm***	**MCU-6000 / APCI-8401:** Timeout Error-Flag: Hat den Wert 1, wenn ein Timeout-Fehler zwischen MCU-6000 / APCI-8401 und ASM-2003 auftritt. Dieses Flag wird automatisch quittiert, sobald die Kommunikation wiederhergestellt ist. Für weitergehende Überwachungen der ASM Kommunikation ist das ASMS Register (Kapitel 4.4.50.1) zu überwachen. **Anmerkung:** Solange das Bit toasm gesetzt ist, kann der Regelkreis dieser Achse nicht geschlossen werden (Kommando cl).
1 **0000 0002**	***eo***	Emergency-Out Error-Flag: Hat den Wert 1, wenn ein als EO-projektierter Digitaleingang aktiv ist.
2 **0000 0004**	***dnr***	Drive-Not-Ready Error-Flag: Hat den Wert 1, wenn ein als DR-projektierter Digitaleingang inaktiv ist.
3 **0000 0008**	***lslh***	Limit-Switch left Hardware Error-Flag: Hat den Wert 1, wenn ein als LSL_SMD, LSL_TOM oder LSL_SMA projektierter Digitaleingang aktiv ist.
4 **0000 0010**	***lsrh***	Limit-Switch right Hardware Error-Flag: hat den Wert 1, wenn ein als LSR_SMD, LSR_TOM oder LSR_SMA projektierter Digitaleingang aktiv ist.
5 **0000 0020**	***lsls***	Limit-Switch left Software Error-Flag: Hat den Wert 1, wenn die linke Software-Endlage überschritten wird. Die linke Software-Endlage ist im achsspezifischen Systemparameter {sll} abgelegt. Damit dieses Flag aktiv wird müssen zusätzlich zwei Bedingungen gelten: Die Softwareendlage muss mit einer der Funktionen TOM oder SMA projektiert werden und zuvor muss das shp()-Kommando ausgeführt worden sein.
6 **0000 0040**	***lsrs***	Limit-Switch right Software Error-Flag: Hat den Wert 1, wenn die rechte Software-Endlage überschritten wird. Die rechte Software-Endlage ist im achsspezifischen Systemparameter {slr} abgelegt. Damit dieses Flag aktiv wird müssen zusätzlich zwei Bedingungen gelten: Die Softwareendlage muss mit einer der Funktionen TOM oder SMA projektiert werden und zuvor muss das shp()-Kommando ausgeführt worden sein.
7 **0000 0080**	***mpe***	Maximum Position Error-Flag: Hat den Wert 1, wenn der zulässige Schleppfehler überschritten wurde. Der maximal erlaubte Schleppfehler wird im Systemparameter {mpe} spezifiziert. Mit Hilfe der PCAP-Befehle wrmpe()und rdmpe() kann der Parameter auch während der Laufzeit verändert werden
8 **0000 0100**	***dhef***	Data handling error flag: Hat den Wert 1 wenn ein Datenfehler (z.B. Inkonsistente Profildaten) durch das rw_MOS-Betriebssystems festgestellt wird. In bestimmten Fällen, werden beim Auftreten dieses Bitts die Regelkreise der jeweiligen Achse geöffnet. Das Rücksetzen dieses Bits ist nur durch Systemneustart *(BootFile*) oder durch Ausführung der Befehle *ra()* [Kapitel 4.4.42] oder *rs()* [Kapitel 4.4.120] möglich. Ggf. muß in diesem Zusammenhang die Systemvariable ErrorReg beachtet werden.

Tabelle 15 (Fortsetzung): Bitkodierter Aufbau des axst-Wortes

Bit-Nr.	Name	Funktion
9 **0000 0200**	***cef***	Daten-Konfigurations-Fehler. Das cef-Flag wird gesetzt, wenn die Informationen für Betriebsarten, Signalverarbeitung oder CPU-Nummer auf der MCU-G3 nicht mit den Systemdaten (system.dat) übereinstimmen Der Konfigurations-Check wird nach folgenden Ereignissen automatisch durchgeführt: • Nach jeder Rücksetzanweisung (z.B. PCAP-Befehl rs()), • Nach jedem Übertragen der Systemdatei system.dat mit dem PCAP-Befehl txbf2(). Die Behebung der Fehlerursache kann durch Speichern der Systemdaten im Menü [Save Changes] erfolgen.
10..11		Nicht belegt, diese Flags haben einen undefinierten Wert.
12 **0000 1000**	***pe***	Profil-Ende Status-Flag: Hat den Wert 1, wenn das Profilende erreicht ist.
13 **0000 2000**	***cl***	Closed-Loop Status-Flag: Hat den Wert 1, wenn der Achskanal in Lageregelung ist.
14 **0000 4000**	***ip***	In-Position Status-Flag: Hat den Wert 1, wenn das Profilende erreicht wurde und zusätzlich die Differenz von Soll- und Istposition des Achskanals die im achsspezifischen Systemparameter {ipw} enthaltene Positionsdifferenz unterschreitet
15 **0000 8000**	***ui***	User Input Status-Flag: Hat den Wert 1, wenn ein als UI-projktierter Digital Eingang aktiv ist.
16 **0001 0000**	***lpsf***	Das Latch Position Synchronous Flag zeigt an, dass ein Latch-Vorgang synchron zum Abtastzyklus stattgefunden hat [Kapitel 4.4.25], oder ein mit LP-Funktion projektierter Digital-Eingang aktiviert wurde [mcfg / Kapitel 1.7.2.5]. Das Flag wird zurückgesetzt durch das Lesen der gelatchten Position LP, z.B. durch das Kommando rdlp.
17 **0002 0000**	***referenced***	Dieses Flag zeigt an, dass die entsprechende Achse mit dem Kommando shp referenziert wurde. Das Flag wird zurückgesetzt beim booten, mit den Kommandos rs(), ra() oder durch Beschreiben von rp. Bei Schrittmotorachsen wird das Flag weiterhin zurückgesetzt beim Öffnen und Schließen des Regelkreises. Dieses Flag ist erst verfügbar ab RWMOS Version V2.5.3.16.
18 **0004 0000**	***refh***	Ref-Hardware-Input-Flag: Hat den Wert 1, wenn ein als REF projektierter Digitaleingang aktiv ist. Dieses Flag ist erst verfügbar ab RWMOS Version V2.5.3.47.
19 **0008 0000**	***saf***	Spooler-Asynchronous-Flag – zeigt an, dass der Spooler dieser Achse im Interpolationsverbund asynchron ist. Das Flag wird zurückgesetzt durch ResetAxis (ra) oder beim Schließen des Regelkreises (cl). Dieses Flag ist erst verfügbar ab RWMOS Version V2.5.3.88.
18..31		Derzeit nicht belegt, diese Flags haben einen undefinierten Wert und sind für zukünftige Verwendung reserviert..

4.4.54 rdaxstb, read axis status bit

BESCHREIBUNG:	Mit dieser Funktion kann eine MCU-G3 Achsen-Statusinformation abgefragt werden. Die Achsnummer muss im Parameter *an* (0, 1, .. *MAXAXIS-1*) spezifiziert werden.
BORLAND DELPHI:	function rdaxstb(an:integer; bitnr:integer):integer;
C:	int rdaxstb(int an, int bitnr);
VISUAL BASIC:	Function rdaxstb(ByVal an As Long, ByVal bitnr As Long) As Long
RÜCKGABEWERT:	Die Funktion liefert den Wert 1 bzw. TRUE zurück, sofern der entsprechende Eingang von *bitnr* aktiv ist. Die Zuordnung von *bitnr* zu den jeweiligen Achsen-Statusinformationen wird in Tabelle 15 beschrieben, jedoch erfolgt bei *bitnr* die Zählweise bei dem Wert 1, d.h. um beispielsweise *pe* abzufragen, muss *bitnr* den Wert 13 haben!
ANMERKUNG:	Siehe auch PCAP-Befehl *rdaxst()*

4.4.55 rdcbcnct, read common buffer CNC-Task

BESCHREIBUNG:	Jede CNC-Task hat einen lokalen Speicherbereich, den sogenannten Common-Buffer, der sowohl von der jeweiligen CNC-Task als auch durch ein PCAP-Programm gelesen und beschrieben werden kann. Mit dieser Funktion kann der komplette CNC-Task-spezifische Buffer (oder nur ein Teil davon) eingelesen werden. Mit dem Funktionsparameter *cbcnct* erfolgt die Auswahl des CNC-Task-Buffers, die einzulesende Größe in Bytes und die Speicheradresse, wohin dieser Block eingelesen werden soll.
BORLAND DELPHI:	function rdcbcnct(var cbcnct:CBCNCT):integer;
C:	int rdcbcnct(struct CBCNCT far *cbcnct);
VISUAL BASIC:	Sub rdcbcnct(DCBCNCT As CBCNCT)
RÜCKGABEWERT:	Die Funktion rdcbcnct() hat folgenden bitkodierten Rückgabewert:

Bit-Nr	Rück-gabewert	Fehler-Beschreibung
0	0	kein Fehler
0	1	wenn ungültige Task-Nummer
1	0	kein Fehler
1	1	wenn maximal erlaubte Buffergröße überschritten Dies bedeutet, dass die Funktion im Normalfall den Wert 0 zurückliefert
2	0	kein Fehler
2	1	Adressfehler / Speicherfehler

ANMERKUNG:	Die CNC-Task-spezifische Buffergröße beträgt 1000 Bytes. Der Struktur- (Record) Aufbau von CBCNCT ist im Kapitel 4.3.2.9 abgedruckt. PCAP-Befehl *wrcbcnct()*, SAP-Befehle *RDCBx()* und *WRCBx()*

4.4.56 rdcd, read common double

BESCHREIBUNG:	Mit dieser Funktion können vordefinierte Variablen der CNC-Task eingelesen werden. Dies sind die *rw_SymPas*-Variablen CD0 .. CD99. Der erste Parameter gibt dabei die Nummer -*index*- der gewünschten einzulesenden Variablen an. Der Wertebereich von *index* ist dabei 0 bis 999. Der zweite Parameter ist ein Zeiger auf ein Feld mit 1000 double-Variablen.
BORLAND DELPHI:	procedure rdcd(ndx: integer; var cdbuf:CDBUF);
C:	void rdcd(int ndx, struct CDBUF far *cdbuf);
VISUAL BASIC:	Sub rdcd(ByVal ndx As Long, CDBUF As CDBUF)
RÜCKGABEWERT:	Der Befehl *rdcd()* trägt im mit *index* spezifizierten Feld den aktuellen Wert der entsprechenden *CD*-Variable ein.
ANMERKUNG:	Der Inhalt aller Common Variablen bleibt auch nach einem Systemrücksetzvorgang, welcher z.B. durch das *rs()*-Kommando ausgeführt wird, gespeichert. Wenn dies nicht erwünscht ist, sollten die betreffenden Variablen beim Programmstart auf den gewünschten Wert gesetzt werden. **Besonderer Hinweis:** Mit dem Index 100 werden die Variable 0..99 gemeinsam gelesen. Die Variable mit dem Index 100 kann mit rdcd nicht gelesen werden. Mit dem Index 1000 werden die Variable 0..999 gemeinsam gelesen.

4.4.57 rdci, read common integer

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *rdcd()*, jedoch werden hier nicht Werte vom Typ double eingelesen sondern vom Typ LONGINT. Es handelt sich dabei um die *rw_SymPas*-Variablen CI0 .. CI999.
BORLAND DELPHI:	procedure rdci(ndx: integer; var cibuf:CIBUF);
C:	void rdci(int ndx, struct CIBUF far *cibuf);
VISUAL BASIC:	Sub rdci(ByVal ndx As Long, CIBUF As CIBUF)
ANMERKUNG:	**Besonderer Hinweis:** Mit dem Index 100 werden die Variable 0..99 gemeinsam gelesen. Die Variable mit dem Index 100 kann mit rdci nicht gelesen werden. Mit dem Index 1000 werden die Variable 0..999 gemeinsam gelesen.

4.4.58 rdcncts, read computerized numeric controller task status

BESCHREIBUNG:	Mit Hilfe dieses Befehls kann der aktuelle Zustand der in *TaskNr* (Werte 0..3) angewählten CNC-Task abgefragt werden. Die Ergebnisse befinden sich nach Ausführung dieses Befehls in der Struktur bzw. dem Record *CNCTS*.
BORLAND DELPHI:	procedure rdcncts(TaskNr:integer; var cncts:CNCTS):integer;
C:	void rdcncts(int TaskNr, struct CNCTS far *cncts);
VISUAL BASIC:	Sub rdcncts(ByVal TaskNr As Long, CNCTS As CNCTS)
RÜCKGABEWERT:	Die Rückgabewerte, welche sich nach Ausführung von *rdcncts()* in CNCTS befinden, werden im Kapitel 4.3.2.10 beschrieben.

4.4.59 rdControllerFlags, read Controller Flag register

BESCHREIBUNG:	Mit diesem Befehl wird das achsspezifische, bitcodierte Register ControllerFlags der RWMOS-Betriebssystemsoftware gelesen.
BORLAND DELPHI:	procedure rdControllerFlags (an: integer; var value: integer);
C:	void rdControllerFlags (long an, long *value);
VISUAL BASIC:	Sub rdControllerFlags (ByVal an As Long, value As Long)
PARAMETER:	Mit *an* wird der anzusprechende Achskanal angegeben (0, 1, ...). In *value* wird der zu lesende bitcodierte Wert des Registers ControllerFlags übergeben.
ANMERKUNG:	Mit Hilfe von Flags (bits) im achsspezifischen ControllerFlags-Register können unterschiedliche Optionen im Regelalgorithmus von RWMOS.ELF aktiviert bzw. gesteuert werden. (siehe hierzu auch Kapitel 4.4.147 und 6.3.1.4).

4.4.60 rddigi, read digital inputs

BESCHREIBUNG:	Mit dieser Funktion können folgende Signalzustände abgefragt werden: • Der aktuelle Zustand der 16 MCU-G3 Digital-Eingänge • Der aktuelle Zustand der Nullspur- (Index) Signals vom Inkrementalenkoder • Ein zwischengespeicherter Fehler des Meßwerterfassungssystems • Eine zwischengespeicherte Flanke des Nullspur- (Index) Signals vom Inkrementalgeber • Eine zwischengespeicherte Flanke des Hardware-Latchsignals (Strobe) Sofern ein Eingang aktiv ist, wird dies mit dem Wert 1 des jeweiligen Bit angezeigt. Optional können alle Digitaleingänge im TOOLSET Programm *mcfg.exe* mit Invertierung projektiert werden. Ebenso ist es möglich, bei Verwendung eines Inkrementalenkoders mit Index-Signal die gewünschte Polarität zu projektieren.
BORLAND DELPHI:	procedure rddigi(var tsrp:TSRP);
C:	void rddigi(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rddigi(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].digi n = 0 .. Anzahl der Achsen -1
RÜCKGABEWERT:	Der bitkodierte Rückgabewert befindet sich in der Struktur- bzw. Recordkomponente *digi* und hat den in nachfolgend abgedruckter Tabelle 16 beschriebenen Aufbau.
ANMERKUNG:	Es gibt keine festgelegte Achszuordnung der Digital-Eingänge. Bit 16..19 können durch den *rdigi()*-Befehl [Kapitel 4.4.80] zurückgesetzt werden. [mcfg / Kapitel 1.7.2.5] und [mcfg / Kapitel 1.7.2.5.1].

4.4.60.1 Achsenqualifizierer digi

Mit dem Register *digi* kann der Zustand der Digital-Eingänge der MCU-G3 abgeprüft werden. Sofern die jeweiligen Eingänge aktiv sind, wird dies mit dem Wert 1 an der jeweiligen Bitposition angezeigt.

Tabelle 16: Bitkodierter Aufbau des digi-Wortes

Bit-Nr.	Funktion	X1 / Pin MCU-3x00
0	Eingang 1	9
1	Eingang 2	10
2	Eingang 3	11
3	Eingang 4	12
4	Eingang 5	13
5	Eingang 6	14
6	Eingang 7	15
7	Eingang 8	16
8	Eingang 9	42
9	Eingang 10	43
10	Eingang 11	44
11	Eingang 12	45
12	Eingang 13	46
13	Eingang 14 **MCU-3000 / MCU-3100:** Hardware-Strobe-Signal zum Latchen der Istposition Achskanal 1	47
14	Eingang 15 **MCU-3000 / MCU-3100:** Hardware-Strobe-Signal zum Latchen der Istposition Achskanal 2	48
15	Eingang 16 **MCU-3000 / MCU-3100:** Hardware-Strobe-Signal zum Latchen der Istposition Achskanal 3	49
16	Null-Spur vom Inkrementalenkoder, achsspezifisch	--
17	Fehler des Meßwerterfassungssystems, achsspezifisch	--
18	Zwischengespeicherter Wert des Nullspursignals vom Inkrementalgeber, achsspezifisch	--
19	Zwischengespeicherter Wert des Latchsignals (Hardware-Strobe), achsspezifisch	--
20	**MCU-3100 / MCU-3400C:** AEA Alarm Error Enkoder Kanal A	--
21	**MCU-3100 / MCU-3400C:** AEB Alarm Error Enkoder Kanal B	--
22	**MCU-3100 / MCU-3400C:** AEN Alarm Error Enkoder Kanal Index	--
23	**MCU-3100 / MCU-3400C:** AES Alarm Error Enkoder Sammelfehler	--
24	**MCU-3400C:** Eingang 17	--
25	**MCU-3400C:** Eingang 18	--
26	**MCU-3400C:** Eingang 19	--
27	**MCU-3400C:** Eingang 20	--
28	**MCU-3400C:** Eingang 21 u. Hardware-Strobe-Signal zum Latchen Achskanal 1	--
29	**MCU-3400C:** Eingang 22 u. Hardware-Strobe-Signal zum Latchen Achskanal 2	--
30	**MCU-3400C:** Eingang 23 u. Hardware-Strobe-Signal zum Latchen Achskanal 3	--
31	**MCU-3400C:** Eingang 24 u. Hardware-Strobe-Signal zum Latchen Achskanal 4	--

4.4.61 rddigib, read digital input bit

BESCHREIBUNG:	Mit dieser Funktion kann der aktuelle Zustand eines MCU-G3 Digital-Eingangs und diverser anderer Logiksignale abgefragt werden. Die Achsnummer muss im Parameter *an* (0, 1, ... *MAXAXIS-1*) spezifiziert werden. Die abzufragende Bitnummer wird in *bitnr* (1..32) angegeben.
BORLAND DELPHI:	function rddigib(an:integer; bitnr:integer):integer;
C:	int rddigib(int an, int bitnr);
VISUAL BASIC:	Function rddigib(ByVal an As Long, ByVal bitnr As Long) As Long
RÜCKGABEWERT:	Die Funktion liefert den Wert 1 bzw. TRUE zurück, sofern der entsprechende Eingang von *bitnr* aktiv ist.
ANMERKUNG:	Bit-Nr. 17..20 können durch den *rdigi()*-Befehl [Kapitel 4.4.80] zurückgesetzt werden. [mcfg / Kapitel 1.7.2.5] und [mcfg / Kapitel 1.7.2.5.1] und PCAP-Befehl *rddigi()*. **Vorsicht:** Die Zählweise der Bitnummer beginnt bei 1.

Tabelle 17: Zuordnung von *bitnr* zu den jeweiligen MCU-G3 Digitaleingängen

'bitnr'	Funktion	X1 / Pin MCU-3x00
1	Eingang 1	9
2	Eingang 2	10
3	Eingang 3	11
4	Eingang 4	12
5	Eingang 5	13
6	Eingang 6	14
7	Eingang 7	15
8	Eingang 8	16
9	Eingang 9	42
10	Eingang 10	43
11	Eingang 11	44
12	Eingang 12	45
13	Eingang 13	46
14	Eingang 14	47
15	Eingang 15	48
16	Eingang 16	49
17	Null-Spur vom Inkrementalenkoder, achsspezifisch	--
18	Fehler des Meßwerterfassungssystems, achsspezifisch	--
19	Zwischengespeicherter Wert des Nullspursignals vom Inkrementalgeber, achsspezifisch	--
20	Zwischengespeicherter Wert des Latchsignals (Hardware-Strobe), achsspezifisch	--
21	**MCU-3100 / MCU-3400C:** AEA Alarm Error Enkoder Kanal A	--
22	**MCU-3100 / MCU-3400C:** AEB Alarm Error Enkoder Kanal B	--
23	**MCU-3100 / MCU-3400C:** AEN Alarm Error Enkoder Kanal Index	--
24	**MCU-3100 / MCU-3400C:** AES Alarm Error Enkoder Sammelfehler	--
25	**MCU-3400C:** Eingang 17	--
26	**MCU-3400C:** Eingang 18	--
27	**MCU-3400C:** Eingang 19	--
28	**MCU-3400C:** Eingang 20	--
29	**MCU-3400C:** Eingang 21	--
30	**MCU-3400C:** Eingang 22	--
31	**MCU-3400C:** Eingang 23	--
32	**MCU-3400C:** Eingang 24	--
21..32	Die je nach Steuerungstyp nicht belegten Flags haben einen undefinierten Wert und sind für zukünftige Verwendung reserviert.	--

4.4.62 rddigo, read digital outputs

BESCHREIBUNG:	Mit diesem Befehl wird der aktuelle Ausgabe-Status der MCU-G3-Digital-Ausgänge in die achsspezifische Struktur- bzw. Record-Komponente *digo* eingelesen. Die dort gesetzten Bits repräsentieren gesetzte Ausgänge.
BORLAND DELPHI:	procedure rddigo(var tsrp:TSRP);
C:	void rddigo(struct TSRP far *tsrp);
VISUAL BASIC:	TSRP[n].digo
TSRP-KOMPONENTEN:	Sub rddigo(DTSRP As TSRP)
RÜCKGABEWERT:	Die bitkodierten Rückgabewerte befinden sich nach Ausführung dieses Befehls in der Struktur- bzw. Recordkomponente *digo*. Diese Komponente hat den beim PCAP-Befehl *wrdigo()* abgedruckten Aufbau.

4.4.63 rddigob, read digital output bit

BESCHREIBUNG:	Mit dieser Funktion kann der aktuelle Zustand eines MCU-G3 Digital-Ausgangs abgefragt werden. Die Achsnummer muss im Parameter *an* (0, 1, ... *MAXAXIS-1*) spezifiziert werden.
BORLAND DELPHI:	function rddigob(an:integer; bitnr:integer):integer;
C:	int rddigob(int an, int bitnr);
VISUAL BASIC:	Function rddigob(ByVal an As Long, ByVal bitnr As Long) As Long
RÜCKGABEWERT:	Die Funktion liefert den Wert 1 bzw. TRUE zurück, sofern der entsprechende Ausgang von *bitnr* aktiv ist. Die Zuordnung von *bitnr* zu den jeweiligen Ausgängen ist beim PCAP-Befehl *wrdigob()* abgedruckt.

4.4.64 rddp, read desired position

BESCHREIBUNG:	Der MCU-G3-Profilgenerator errechnet eine interne Führungsgröße, die sogenannte Sollposition (= gewünschte Position oder desired position). Diese kann mit diesem Befehl eingelesen werden. Im Normalfall muss in der Betriebsart Lageregelung die Ist-Position [Kapitel 4.4.102 - *rdrp()*] und diese Sollposition bis auf tolerierbare Abweichungen identisch sein.
BORLAND DELPHI:	procedure rddp(var tsrp:TSRP);
C:	void rddp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rddp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].dp
RÜCKGABEWERT:	Nach Ausführung des Befehls steht die Sollposition im Feld *dp* zur Verfügung. Der Wert wird in der achsspezifischen Positionseinheit zurückgeliefert.
ANMERKUNG:	Diese Sollposition wird unter anderem auch zur Soll-Istwert-Differenzbildung für die automatische Schleppfehlerüberwachung herangezogen.

4.4.65 rddpoffset, read desired position offset

BESCHREIBUNG:	Mit dieser Funktion kann der aktuell programmierte Wert des Achsen-Qualifizierers dpoffset gelesen werden.
BORLAND DELPHI:	function rddpoffset (an: integer; var value: double): integer;
C:	int rddpoffset(int an, double *value);
VISUAL BASIC:	Function rddpoffset (ByVal an As Long, value As Double) As Long
PARAMETER:	Mit *an* wird der auszulesende Achskanal angegeben (0, 1, ...). In *value* wird der zu schreibende Positionsoffset in der achsspezifischen Positionseinheit zurückgegeben.
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Siehe hierzu auch Kapitel 4.4.151

4.4.66 rddpd – read desired position in display unit

BESCHREIBUNG:	Der MCU-G3-Profilgenerator errechnet eine interne Führungsgröße, die sogenannte Sollposition (= gewünschte Position oder desired position). Diese kann mit diesem Befehl in der achsspezifischen Anzeigeeinheit (display unit) eingelesen werden.
BORLAND DELPHI:	procedure rddpd(var tsrp:TSRP);
C:	void rddpd(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rddpd(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].dp
RÜCKGABEWERT:	keiner
WIRKUNG:	Nach Ausführung des Befehls steht die Sollposition im Feld *dp* zur Verfügung. Der Wert wird in der achsspezifischen Anzeigeeeinheit zurückgeliefert.
ANMERKUNG:	siehe auch Kommandos rddp, rdrp, rdrpd

4.4.67 rddv, read desired velocity

BESCHREIBUNG:	Diese Funktion liefert die achsspezifische Soll-Geschwindigkeit des MCU-G3-Profilgenerators zurück. Im Idealfall entspricht der eingelesene Wert der tatsächlichen Achsgeschwindigkeit (Ist-Geschwindigkeit).
BORLAND DELPHI:	procedure rddv(var tsrp:TSRP);
C:	void rddv(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rddv(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].dv
RÜCKGABEWERT:	Nach Ausführen der Funktion, steht die Sollgeschwindigkeit im Register *dv* in der achsspezifischen Geschwindigkeitseinheit zur Verfügung.
ANMERKUNG:	Die Sollgeschwindigkeit kann nur durch entsprechende Verfahrbefehle beeinflußt werden.

4.4.68 rddvoffset, read desired velocity offset

BESCHREIBUNG:	Mit dieser Funktion kann der aktuell programmierte Wert des Achsen-Qualifizierers dvoffset gelesen werden.
BORLAND DELPHI:	function rddvoffset (an: integer; var value: double): integer;
C:	int rddvoffset(int an, double *value);
VISUAL BASIC:	Function rddvoffset (ByVal an As Long, value As Double) As Long
PARAMETER:	Mit *an* wird der auszulesende Achskanal angegeben (0, 1, ...). In *value* wird der aktuell gesetzte Geschwindigkeitswert in der achsspezifischen Positionseinheit zurückgegeben.
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Siehe hierzu auch Kapitel 4.4.152

4.4.69 rdEffRadius – Read Effective Radius

BESCHREIBUNG:	Mit diesem Befehl kann der effektive Radius für eine rotatorische Achse gelesen werden.
BORLAND DELPHI:	rdEffRadius (an: integer; var value: double);
C:	void rdEffRadius (long an, double *value);
VISUAL BASIC:	Sub rdEffRadius (an As Long, ByVal value As Double)
PARAMETER:	In *an* wird die Achsnummer angegeben, in *value* wird der wirksame Radius in der Einheit zurückgeliefert, die per PU gesetzt ist.
RÜCKGABEWERT:	keiner
ANMERKUNG:	siehe Kapitel 6.3.3

4.4.70 rdepc, read EEPROM programming cycle

BESCHREIBUNG:	Mit dieser Funktion kann die momentane Anzahl der MCU-G3 EEPROM Programmierzyklen gelesen werden. Die Zyklusnummer wird bei jedem Speichervorgrang im TOOLSET Programm *mcfg.exe* im EEPROM um eins erhöht. Das EEPROM läßt sich mindestens 10000 mal beschreiben.
BORLAND DELPHI:	procedure rdepc(var tsrp:TSRP);
C:	void rdepc(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdepc(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].epc
RÜCKGABEWERT:	Die momentane Programmier-Zyklusnummer befindet sich nach Ausführung dieses Befehls in der Struktur- bzw. Record-Komponente *epc*.

4.4.71 rdErrorReg, read Error Register

BESCHREIBUNG:	Mit dieser Funktion kann das ErrorRegister der RWMOS-Betriebssystemsoftware ausgelesen werden.
BORLAND DELPHI:	procedure rdErrorReg(var ErrorReg: integer);
C:	void rdErrorReg (long *ErrorReg);
VISUAL BASIC:	Sub rdErrorReg (ErrorReg As Integer)
RÜCKGABEWERT:	Der bitkodierte Wert des ErrorRegisters wird in ErrorReg zurückgeliefert. Die Funktion hat keinen Rückgabewert.
ANMERKUNG:	Aufbau des Error-Registers siehe nächstes Kapitel. Abnullen des Error Registers ist möglich durch Schreiben von 0 mit wrErrorReg().

4.4.71.1 Das Register ErrorReg

In dem Register *ErrorReg* werden diverse Fehlerzustände der RWMOS.Betriebssystemsoftware angezeigt. Das Register ist bitcodiert.

Tabelle 18: Bitkodierter Aufbau des ErrorReg-Wortes

Bit-Nr.	Name	Funktion	Hex
0	errAxDef	Achse in AS war mehrfach selektiert bei einem Verfahrkommando	1
1	errTargetVel	Zielgeschwindigkeit <> 0 am Spoolerende, obwohl ForbidTargetVel gesetzt ist: System wurde zurückgesetzt	2
2	errUnit	eine ungültige Einheit wurde verwendet	4
3	errCenterPoint	ungültiger Mittelpunkt bei Kreis programmiert oder es wurde ein Kreis mit Radius = 0 programmiert	8
4	errSpooler Overrun	Spoolerüberlauf bei einer Achse erkannt	10
5	ProfileToSmall	Im Spoolerbetrieb werden hintereinander mindestens zwei Verfahrprofile ausgeführt, deren Ausführungszeit kürzer als die Abtastzeit ist. Dies kann Fehler im Programmablauf verursachen und ist nicht zulässig.	20
6	SplineSizeErr	zu viele Spline Sätze werden geladen	40
7	RotationFail	Fehler bei Achsrotation	80
8	PciBusError	Fehler im Interrupt-Cause-Register der PCI-Brücke erkannt	100
9	CheckMonitor Screen	Eine Fehlerausgabe im Monitor Screen wurde generiert	200
10	SsfWait Refused	Mindestens ein SSF Wartebefehl wurde ignoriert, weil die Zielgeschwindigkeit des vorhergehenden Verfahrbefehls ungleich 0 war Dies deutet auf einen Programmierfehler in der Anwendersoftware hin!	400
11	SpoolerLoad Error	Beim Beschreiben des Spoolers ist ein Fehler aufgetreten, weil gleichzeitig ein Verfahrprofil vom System generiert wurde. Dies kann passieren wenn z.B. während dem Aufruf eines Interpolationskommandos ein Endschalter anspricht.	800
12	VelocityZero	Dieses Bit zeigt an, das ein Interpolationskommando mit Verfahrgeschwindigkeit = 0 erkannt wurde. Abhängig vom Bit InhibitProfileRefuse (Register MODEREG Kapitel 6.3.1.5) wird das Profil automatisch verworfen. Dies deutet auf einen Programmierfehler in der Anwendersoftware oder auf ein Konfigurationsproblem des Anwenders hin!	1000
13	AccelZero	Dieses Bit zeigt an, das ein Interpolationskommando mit Beschleunigung = 0 erkannt wurde. Abhängig vom Bit InhibitProfileRefuse (Register MODEREG Kapitel 6.3.1.5) wird das Profil automatisch verworfen. Dies deutet auf einen Programmierfehler in der Anwendersoftware oder auf ein Konfigurationsproblem des Anwenders hin!	2000
14	LimitDefError	Ein ungültiger Begrenzungswert wurde erkannt in mcpmax, mcpmin, mcpcp oder mcpcn (ungültiger Zahlenwert)	4000
15	ZeroProfile	Ein Interpolationskommando wurde verworfen, weil der angegebene Verfahrweg nahezu oder gleich 0 ist.	8000
16	RadiusError	Ein Kreis- oder Helix-Kommando wurde verworfen, weil der zu realisierende Kreisradius nahezu oder gleich 0 ist.	0001 0000

Fortsetzung Tabelle 19: Bitkodierter Aufbau des ErrorReg-Wortes

Bit-Nr.	Name	Funktion	Hex
17	SpoolerDeep ToLess	Die eingetragenen Verfahrprofile im Spooler reichen für die Look-Ahead Berechnung nicht aus. Dadurch werden unnötige Geschwindigkeitsbegrenzungen verursacht. In ungünstigen Fällen sind unerlaubte Beschleunigungen möglich.	0002 0000
18	IO ResetHandled	Im IO Bereich der Steuerung fand ein aussergewöhnelicher Reset statt. Diese kann auf ein Hardwareproblem hindeuten	0004 0000
19	JSatSAFdone	Die Spooler-Synchronitätsüberwachung hat einen Fehlerzustand erkannt und mindestens eine Achse ausserplanmäßig gestoppt und aus der Interpolationsgruppe herausgenommen.	0008 0000
20	reserved	Dieses Bit ist für die Behandlung einer Option reserviert.	0010 0000
21..31		reserviert für zukünftige Verwendung, diese Flags haben einen undefinierten Wert	

4.4.72 rdf, read filter

BESCHREIBUNG:	Mit Hilfe dieses Befehls können die aktuellen achsspezifischen PIDF-Filter-Koeffizienten der MCU-G3 eingelesen werden. Die Defaultwerte dieser Koeffizienten werden mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure rdf(var tsrp:TSRP);
C:	void rdf(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdf(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].kp, TSRP[n].ki, TSRP[n].kd, TSRP[n].kpl, TSRP[n].kfca, TSRP[n].kfcv n = 0 .. Anzahl vorhandener Achse n-1
RÜCKGABEWERT:	Nach Ausführung des Befehls stehen die Rückgabewerte in den oben aufgeführten TSRP-Struktur- bzw. Record-Komponenten.
ANMERKUNG:	Weitere Angaben zum PIDF-Filter sind im Kapitel 2.1.2, [BHB / Kapitel 4.1.1] und [IHB / Kapitel 6.2] enthalten. PCAP-Befehl *uf()*

4.4.73 rdGCR, read gear configuration register

BESCHREIBUNG:	Mit dieser Funktion kann das achsspezifische Gear Configuration Register gelesen werden. [Kapitel 6.3.3]
BORLAND DELPHI:	procedure rdGCR (an: integer; var value: integer);
C:	void rdGCR (long an, long *value);
VISUAL BASIC:	Sub rdGCR (ByVal an As Long, value As Long)
PARAMETER:	Mit *an* wird der auszulesende Achskanal angegeben (0, 1, ...). In *value* wird der Inhalt des GCR Registers zurückgegeben.
RÜCKGABEWERT:	keiner
ANMERKUNG:	Siehe auch Doku zum Resourcen-Interface - GEAR

4.4.74 rdgf, read gear factor

BESCHREIBUNG:	Diese Funktion liefert den achsspezifischen Getriebe-Faktor {gf} zurück. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure rdgf(var tsrp:TSRP);
C:	void rdgf(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdgf(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].gf
RÜCKGABEWERT:	Nach Ausführen der Funktion, steht der Faktor im Register *gf* in der achsspezifischen Einheit zur Verfügung.
ANMERKUNG:	Der Getriebefaktor kann mit dem PCAP-Befehl *wrgf()* jederzeit gesetzt werden.

4.4.75 rdgfaux, read gear factor auxiliary channel

BESCHREIBUNG:	Diese Funktion liefert das achsspezifische Verhältnis zwischen Schrittmotor-Auflösung und Enkoder-Kanal bei Stepper-Systemen mit Enkoderverification zurück. Defaultwert ist 1.0, der Wert kann nur zur Laufzeit verändert werden.
BORLAND DELPHI:	function rdgfaux (an: integer; var value: double) : integer;
C:	int rdgfaux(int an, double *value)
VISUAL BASIC:	Function rdgfaux (ByVal an As Long, value As Double) As Long
RÜCKGABEWERT:	Die Funktion liefert nach erfolgreicher Ausführung 0 zurück. In diesem Fall steht in value der achsspezifische Wert von gfaux zur Verfügung. Bei einem Rückgabewert ≠ 0 konnte der Wert nicht gelesen werden weil z.B. RWMOS.ELF das Kommando nicht unterstützt. 0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Der Faktor kann mit dem PCAP-Befehl *wrgfaux()* jederzeit gesetzt werden. Siehe auch Kapitel 6.3.3

4.4.76 rdhac, read home acceleration

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische Referenzfahrtbeschleunigung *hac* eingelesen werden. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure rdhac(var tsrp:TSRP);
C:	void rdhac(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdhac(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].hac
RÜCKGABEWERT:	Nach Ausführung des Befehls steht die Referenzfahrtbeschleunigung im Feld *hac* zur Verfügung. Der Wert wird in der achsspezifischen Beschleunigungseinheit zurückgeliefert.
ANMERKUNG:	Die Referenzfahrtbeschleunigung kann unter anderem mit dem PCAP-Befehl *wrhac()* jederzeit gesetzt werden.

4.4.77 rdhvl, read home velocity

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische Referenzfahrtgeschwindigkeit *hvl* eingelesen werden. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure rdhvl(var tsrp:TSRP);
C:	void rdhvl(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdhvl(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].hvl
RÜCKGABEWERT:	Nach Ausführung des Befehls steht die Referenzfahrtgeschwindigkeit im Feld *hvl* zur Verfügung. Der Wert wird in der achsspezifischen Geschwindigkeitseinheit zurückgeliefert.
ANMERKUNG:	Die Referenzfahrtgeschwindigkeit kann unter anderem mit dem PCAP-Befehl *wrhvl()* jederzeit gesetzt werden.

4.4.78 rdifs, read interface status

Funktionsbeschreibung gilt nur für MCU-3000 / MCU-3100 / MCU-3400C

BESCHREIBUNG:	Mit diesem Register können Statusinformationen der MCU-G3 eingelesen werden.
BORLAND DELPHI:	function rdifs(var tsrp:TSRP): integer;
C:	int rdifs(struct TSRP far *tsrp);
VISUAL BASIC:	Function rdifs(DTSRP As TSRP) As Long
TSRP-KOMPONENTEN:	TSRP[n].ifs
RÜCKGABEWERT:	Der bitkodierte Funktionswert befindet sich in der Struktur- bzw. Recordkomponente *ifs* und hat den in nachfolgend abgedruckter Tabelle 20 beschriebenen Aufbau. **Funktionsrückgabewert:** 0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen

4.4.78.1 Achsenqualifizierer ifs

Nachfolgender Abschnitt gilt nur für die MCU-3000 / MCU-3100 / MCU-3400C.
Mit diesem Register können verschiedene Status- und Fehler-Informationen der MCU-G3 abgefragt werden. Sofern die jeweilige Status- oder Error-Information gültig ist, wird dies mit dem Wert 1 an der jeweiligen Bitposition angezeigt. Die einzelnen Bits repräsentieren wichtige interne Zustandsinformationen der MCU-3000. Ursachen für diese Fehler können Spannungsversorgungs-, EMV- oder Hardwareprobleme sein und dürfen im Normalfall nicht auftreten. Bei Auftreten eines derartigen Fehlers wird das Steuerungsinterne I/O Interface zurückgesetzt. Ein ordnungsgemässes Arbeiten mit der Steuerung ist erst nach einem Reboot der Steuerung gewährleistet.
Diese Zustandsinformationen müssen von einem Applikationsprogramm zyklisch überwacht werden.

Tabelle 20: Bitkodierter Aufbau des ifs-Wortes

Bit-Nr.	Funktion
0	*edv*: Die im EEPROM abgelegten Systeminformationen und Daten sind gültig.
1	*cncrdy*: Das CNC-Betriebsbereit-Relais ist aktiv (geschlossen).
16	*pfe*: Das Power-Fail-Error-Flag wird immer dann auf „1“ gesetzt, wenn die Betriebsspannung auf der MCU-G3 eine Schwellenspannung von 2.85V unterschreitet. Nach dem Einschalten der Baugruppe ist das Flag ebenfalls auf „1“ gesetzt.
17	*wdog*: Das Watchdog-Flag wird auf „1“ gesetzt, sofern die Watchdog-Sicherheitslogik (Primärkreis) auf der MCU-G3 angesprochen hat.
18	*iae*: Das Invalid-Access-Error-Flag wird auf „1“ gesetzt, sofern innerhalb der Betriessystemsoftware *rw_MOS* ein ungültiger Zugriff stattgefunden hat.
19	*scwdog*: Das Watchdog-Flag wird auf „1“ gesetzt, sofern die Watchdog-Sicherheitslogik (Sekundärkreis) auf der MCU-G3 angesprochen hat.
20	*scpfe*: Das Power-Fail-Error-Flag wird immer dann auf „1“ gesetzt, wenn die Betriebsspannung auf der MCU-G3 eine Schwellenspannung von 4.75V unterschreitet. Nach dem Einschalten der Baugruppe ist das Flag ebenfalls auf „1“ gesetzt.
21	Bus-Error-Flag: zeigt einen Fehler in der Kommunikation an z.b. bei ENDAT, SSI oder EtherCAT
22	EpmBaseResetFlag: ein unerwarteter Hardware Reset im I/O-Bereich der Basisplatine wurde erkannt. Dies deutet auf EMV-Probleme oder einen Hardwarefehler hin.
23	EpmOpmfResetFlag: ein unerwarteter Hardware Reset im I/O-Bereich des Optionprints wurde erkannt. Dies deutet auf EMV-Probleme oder einen Hardwarefehler hin.
23..31	Nicht belegt, diese Flags haben einen undefinierten Wert

Anmerkungen: Die Fehlerflags 16..20 werden in einer Initialisierungsroutine der *rw_MOS*-Firmware aus einem internen Logikregister in das *ifs*-Register kopiert. Das Logikregister wird anschließend gelöscht, d.h. die Flags stehen nach einem zweiten Bootvorgang (*BootFile*) nicht mehr zur Verfügung. Die Flags können auch durch den *rifs()*-Befehl [Kapitel 4.4.78] zurückgesetzt werden.

4.4.79 rdifsb, read interface status bit

Funktionsbeschreibung gilt nur für MCU-3000 / MCU-3100 / MCU-3400C	
BESCHREIBUNG:	Mit dieser Funktion kann eine interface Statusinformation abgefragt werden. Die Achsnummer muss im Parameter *an* (0, 1, ... *MAXAXIS-1*) spezifiziert werden.
BORLAND DELPHI:	function rdifsb(an:integer; bitnr:integer):integer;
C:	int rdifssb(int an, int bitnr);
VISUAL BASIC:	Function rdifsb(ByVal an As Long, ByVal bitnr As Long) As Long
RÜCKGABEWERT:	Die Funktion liefert den Wert 1 bzw. TRUE zurück, sofern der entsprechende Eingang von *bitnr* aktiv ist. Die Zuordnung von *bitnr* zu den jeweiligen Statusinformationen wird in Tabelle 20 beschrieben, jedoch erfolgt bei *bitnr* die Zählweise bei dem Wert 1, d.h. um beispielsweise *edv* abzufragen, muss *bitnr* den Wert 1 haben!
ANMERKUNG:	Siehe auch PCAP-Befehl *rdifs()*

4.4.80 rdigi, reset digital inputs

Funktionsbeschreibung gilt nur für MCU-3000 / MCU-3100 / MCU-3400C	
BESCHREIBUNG:	Mit diesem Befehl können die in *digi* achsspezifisch gespeicherten Statusinformationen gelöscht werden.
BORLAND DELPHI:	procedure rdigi(var tsrp:TSRP);
C:	void rdigi(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdigi(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].digi n = 0 .. Anzahl der Achsen -1
ANMERKUNG:	*rddigi()* [Kapitel 4.4.59]

4.4.81 rdipw, read in position window

BESCHREIBUNG:	Diese Funktion liefert das achsspezifische In-Positions-Fenster zurück.
BORLAND DELPHI:	procedure rdipw(var tsrp:TSRP);
C:	void rdipw(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdipw(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].ipw
ANMERKUNG:	Nach Ausführen der Funktion steht das In-Positions-Fenster im Register *ipw* in der achsspezifischen Positionseinheit zur Verfügung. PCAP-Befehl *wripw()*

4.4.82 rdirqpc, read interrupt request PC

BESCHREIBUNG:	Mit diesem Befehl kann der momentane Zustand der auf der MCU-G3 generierten Interruptquelle abgefragt werden. Sofern der Interrupt aktiv ist, liefert die Funktion den Wert 1 zurück, ansonsten den Wert 0.
BORLAND DELPHI:	function rdirqpc: integer;
C:	int rdirqpc(void);
VISUAL BASIC:	Function rdirqpc() As Long
ANMERKUNG:	Der Interrupt kann u.a. durch die Systemvariable *IRQPC* mit Hilfe eines SAP-Programms gesetzt bzw. rückgesetzt werden [Kapitel 6.3.1.1 - PC-Interrupt-Generierung].

4.4.83 rdjac, read jog accleration

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische *jog*- (auch Eilgang-) Beschleunigung *jac* eingelesen werden. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure rdjac(var tsrp:TSRP);
C:	void rdjac(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdjac(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].jac
RÜCKGABEWERT:	Nach Ausführung des Befehls steht die *jog*-Beschleunigung im Feld *jac* zur Verfügung. Der Wert wird in der achsspezifischen Beschleunigungseinheit zurückgeliefert.
ANMERKUNG:	Die *jog*-Beschleunigung kann mit dem PCAP-Befehl *wrjac()* jederzeit gesetzt werden.

4.4.84 rdJerkRel, read jerkrel

BESCHREIBUNG:	Mit diesem Befehl wird der achsspezifische Parameter *jerkrel* in *value* eingelesen.
BORLAND DELPHI:	procedure rdJerkRel (an: integer; var value: double);
C:	void rdJerkRel (long an, double *value);
VISUAL BASIC:	Sub rdJerkRel (an As Long, ByVal value As Double)
PARAMETER:	an = Achsnummer (0..n) Double = Platzhalter für Funktionswert
RÜCKGABEWERT:	keiner
ANMERKUNG:	*jerkrel* hat immer einen Wert von 0..1. Siehe dazu auch Kapitel 6.3.3.

4.4.84.1 Achsenqualifizierer *jerkrel*

Dies ist ein Faktor zur Parametrierung des Beschleunigungsverlaufs bei S-förmigen Geschwindigkeitsprofilen (Ruckbegrenzung). Dieser Faktor ist nur wirksam wenn ein S-Profil angewählt ist (siehe Register MODEREG Kapitel 6.3.1.5) und hat folgende Bedeutung:
Die Beschleunigung, welche bei S-Profilen angegeben wird, ist stets die mittlere Beschleunigung über den gesamten Beschleunigungs-/Bremsvorgang. Die Maximalbeschleunigung in der Beschleunigungs-/Bremsrampe berechnet sich nach:

$$a_{max} = a * (1 + jerkrel)$$

Auf den Verlauf der Beschleunigung hat der Wert von *jerkrel* folgenden Einfluss:

0 = rechteckförmiger Beschleunigungsverlauf
1 = dreieckförmiger Beschleunigungsverlauf
dazwischen = trapezförmiger Beschleunigungsverlauf

Beispiel: *jerkrel* wird der Wert 0.2 zugewiesen.

- Die Beschleunigung hat nun bei allen Profilen einen trapezförmigen Verlauf.
- Die maximale Beschleungung in der Trapezmitte ist das 1.2-fache der eingestellten Beschleunigung.

Die mittlere Beschleunigung über den gesamten Beschleunigungs- oder Bremsvorgang, ist die programmierte Beschleunigung (jac bei JOG-Befehlen oder trac bei MOVE-Befehlen).
Für *jerkrel* sind Werte zwischen 0 und 1 möglich. Defaultwert ist 1. Werte ausserhalb des Bereiches von 0..1 werden auf 0 bzw. 1 begrenzt.

4.4.85 rdjtvl, read jog target velocity

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische *jog*- (auch Eilgang-) Zielgeschwindigkeit *jtvl* eingelesen werden. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure rdjtvl(var tsrp:TSRP);
C:	void rdjtvl(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdjtvl(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].jtvl
RÜCKGABEWERT:	Nach Ausführung des Befehls steht die *jog*-Zielgeschwindigkeit im Feld *jtvl* zur Verfügung. Der Wert wird in der achsspezifischen Geschwindigkeitseinheit zurückgeliefert.
ANMERKUNG:	Die Zielgeschwindigkeit kann unter anderem mit dem PCAP-Befehl *wrjtvl()* jederzeit gesetzt werden.

4.4.86 rdjvl, read jog velocity

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische *jog*- (auch Eilgang-) Geschwindigkeit *jvl* eingelesen werden. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure rdjvl(var tsrp:TSRP);
C:	void rdjvl(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdjvl(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].jvl
RÜCKGABEWERT:	Nach Ausführung des Befehls steht die *jog*-Geschwindigkeit im Feld *jvl* zur Verfügung. Der Wert wird in der achsspezifischen Geschwindigkeitseinheit zurückgeliefert.
ANMERKUNG:	Die Zielgeschwindigkeit kann unter anderem mit dem PCAP-Befehl *wrjvl()* jederzeit gesetzt werden.

4.4.87 rdledgn, read led green

BESCHREIBUNG:	
MCU-3000 / APCI-8001:	Der aktuelle Zustand der LED D29 (grün) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-3100 / APCI-8008:	Der aktuelle Zustand der LED D53 (grün) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-6000 / APCI-8401:	Der aktuelle Zustand der LED D10 (grün) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-3400C/CPCI-8004:	Der aktuelle Zustand der LED D36 (grün) kann mit Hilfe dieser Funktion eingelesen werden.
BORLAND DELPHI:	function rdledgn: integer;
C:	int rdledgn(void);
VISUAL BASIC:	Function rdledgn() As Long
RÜCKGABEWERT:	Der Rückgabewert der Funktion ist 1, sofern die Leuchtdiode eingeschaltet ist, ansonsten 0.
ANMERKUNG:	PCAP-Befehl *wrledgn()*, Systemvariable *LEDGN*

4.4.88 rdledrd, read led red

BESCHREIBUNG:	
MCU-3000 / APCI-8001:	Der aktuelle Zustand der LED D31 (rot) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-3100 / APCI-8008:	Der aktuelle Zustand der LED D56 (rot) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-6000 / APCI-8401:	Der aktuelle Zustand der LED D12 (rot) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-3400C/CPCI-8004:	Der aktuelle Zustand der LED D38 (rot) kann mit Hilfe dieser Funktion eingelesen werden.
BORLAND DELPHI:	function rdledrd: integer;
C:	int rdledrd(void);
VISUAL BASIC:	Function rdledrd() As Long
ANMERKUNG:	PCAP-Befehl *wrledrd()*, Systemvariable *LEDRD*

4.4.89 rdledyl, read led yellow

BESCHREIBUNG:	
MCU-3000 / APCI-8001:	Der aktuelle Zustand der LED D30 (gelb) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-3100 / APCI-8008:	Der aktuelle Zustand der LED D55 (gelb) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-6000 / APCI-8401:	Der aktuelle Zustand der LED D11 (gelb) kann mit Hilfe dieser Funktion eingelesen werden.
MCU-3400C/CPCI-8004:	Der aktuelle Zustand der LED D37 (gelb) kann mit Hilfe dieser Funktion eingelesen werden.
BORLAND DELPHI:	function rdledyl: integer;
C:	int rdledyl(void);
VISUAL BASIC:	Function rdledyl() As Long
ANMERKUNG:	PCAP-Befehl *wrledyl()*, Systemvariable *LEDYL*

4.4.90 rdlp, read latched position

BESCHREIBUNG:	Diese Funktion liefert die achsspezifische Latch-Position zurück. Der Latch-Vorgang kann durch verschiedene Mechanismen ausgelöst werden: 1. Beim Aktivieren eines mit LP-Funktion projektierten Eingangs. Hierbei beträgt die maximale Verzögerungszeit zwei Abtastintervalle (2.56ms). Ein neuer Latch-Vorgang wird erst nach disaktivieren des Latcheingangs ermöglicht. Diese Methode sollte nur verwendet werden wenn 3 nicht möglich ist. 2. Sofern zuvor ein *lps()*-PCAP-Kommando [Kapitel 4.4.25] ausgeführt wurde und die dort im Parameter *mst* spezifizierte Verzögerung abgelaufen ist. Diese Methode sollte nur verwendet werden wenn 3 nicht möglich ist. 3. In Echtzeit (max. 1µs Verzögerung) durch vorbelegte Digitaleingänge der MCU-G3. Ein neuer Latch-Vorgang wird erst nach disaktivieren des Latcheingangs ermöglicht. Dies ist die bevorzugte Methode für die Erfassung gelatchter Positionswerte. Bei allen Methoden wird die Ist-Position {*rp}* der Motorachse zwischengespeichert. Bei Schrittmotorsystemen oder Analogrückmeldung mit Enkoderverifikation kann auch der Hilfskanal AUX gelatcht werden, wenn die Option „*Use Encoder for position feedback“* aktiviert ist (mcfg Systemdaten).
BORLAND DELPHI:	procedure rdlp(var tsrp:TSRP);
C:	void rdlp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdlp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].lp
RÜCKGABEWERT:	Nach Ausführen der Funktion, steht die Latchposition im Register *lp* in der achsspezifischen Positionseinheit zur Verfügung. Die Priorität der drei Methoden ist gleichbedeutend mit der Reihenfolge der Auflistung, d.h. Echtzeit-Latchen hat die höchste Priorität.
ANMERKUNG:	PCAP-Befehl *wrlp()*

4.4.91 rdlpndx, read latched position index

Funktionsbeschreibung gilt nur für MCU-3000 / MCU-3100 / MCU-3400C

BESCHREIBUNG:	Diese Funktion liefert die achsspezifische Latch-Position des Indexsignals (Nullspur) zurück. Beim Akivieren der Nullspur des Inkremetalgebers wird die Ist-Position {*rp}* der Motorachse in Echtzeit zwischengespeichert.
BORLAND DELPHI:	procedure rdlpndx(var tsrp:TSRP);
C:	void rdlpndx(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdlpndx(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].lp
RÜCKGABEWERT:	Nach Ausführen der Funktion, steht die Latchposition im Register *lp* in der achsspezifischen Positionseinheit zur Verfügung.
ANMERKUNG:	Das Latchen der Nullspur vom Inkrementalgeber ist hilfreich bei der Enkoderverifikation und bei der Referenzfahrt-Programmierung. PCAP-Befehl *wrlpndx()*

4.4.92 rdlsm, read left spool memory

BESCHREIBUNG:	Dieser Befehl liefert den freien Spoolbereich in Bytes zurück. Mit Hilfe eines PCAP- oder SAP- Anwenderprogramms kann der frei verfügbare Spoolbereich jederzeit abgefragt und gegebenenfalls nachgeladen werden. Somit ist es möglich sehr große Verfahrprofile ohne Unterbrechung der Profilgenerierung nachzuladen. Das Laden des Spoolbereichs erfolgt mit *spool*-Befehlen und kann mit beiden Programmiermethoden (PCAP und SAP) durchgeführt werden. Alle *spool*-Befehle bewirken eine Abnahme des frei verfügbaren Spoolbereichs und alle aus dem Spoolbereich ausgeführten Befehle ein Anwachsen.
BORLAND DELPHI:	procedure rdlsm(var tsrp:TSRP);
C:	void rdlsm(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdlsm(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].lsm
ANMERKUNG:	Die Spoolergröße ist achsspezifisch, d.h. dass ggf. der freie Spoolbereich der einzelnen Achskanäle stark unterschiedlich sein kann. Je nach Konfiguration und Anzahl der Achsen stehen ca. 145kByte Spoolbereich pro Achskanal zur Verfügung. Der benötigte Spoolerspeicher pro Befehl kann sich bei zukünftigen Betriebssystemversionen ändern und sollte nicht zur Bestimmung der im Spooler vorhandenen Verfahrprofile verwendet werden.

4.4.93 rdMaxAcc – Read Maximum Acceleration Check

BESCHREIBUNG:	Mit diesem Befehl kann der maximale achsspezifische Beschleunigungswert (*MaxAcc*) gelesen werden. Dieser Wert kann von der RWMOS-Betriebssystemsoftware verwendet werden, um die Bahnbeschleunigung derart zu begrenzen, dass keine der an einer Linear-Interpolation beteiligten Achsen, ihre maximal erlaubte Beschleunigung überschreitet. Im Bedarfsfall wird die Bahnbeschleunigung entsprechend verringert.
BORLAND DELPHI:	rdMaxAcc (an: integer; var value: double);
C:	void rdMaxAcc (long an, double *value);
VISUAL BASIC:	Sub rdMaxAcc (an As Long, ByVal value As Double)
PARAMETER:	In *an* wird die Achsnummer angegeben, in *value* wird die maximal erlaubte Beschleunigung zurückgeliefert. Dieser Wert wird stets in der Interpolationseinheit interpretiert.
RÜCKGABEWERT:	keiner
ANMERKUNG:	siehe Kapitel 4.4.171 und 6.3.3

4.4.94 rdMaxVel – Read Maximum Velocity Check

BESCHREIBUNG:	Mit diesem Befehl kann der maximale achsspezifische Geschwindigkeitswert (*MaxVel*) für Linearinterpolationsbefehle gelesen werden. Dieser Wert kann von der RWMOS-Betriebssystemsoftware verwendet werden, um die Bahngeschwindigkeit derart zu begrenzen, dass keine der an einer Linear-Interpolation beteiligten Achsen, ihre maximal erlaubte Geschwindigkeit überschreitet. Im Bedarfsfall wird dann die Bahngeschwindigkeit verringert.
BORLAND DELPHI:	rdMaxVel (an: integer; var value: double);
C:	void rdMaxVel (long an, double *value);
VISUAL BASIC:	Sub rdMaxVel (an As Long, ByVal value As Double)
PARAMETER:	In *an* wird die Achsnummer angegeben, in *value* wird die maximal erlaubte Geschwindigkeit zurückgeliefert. Dieser Wert wird stets in der Interpolationseinheit interpretiert.
RÜCKGABEWERT:	keiner
ANMERKUNG:	siehe Kapitel 4.4.172 und 6.3.3

4.4.95 rdMCiS – Read Move Commands in Spooler

BESCHREIBUNG:	Mit dieser Funktion kann die Anzahl der Bewegungskommandos im Spooler einer Achse gelesen werden.
BORLAND DELPHI:	procedure rdMCiS (an: integer; var value: integer);
C:	void rdMCiS (long an, long *value);
VISUAL BASIC:	Sub rdMCiS (an As Long, ByVal value As Long)
PARAMETER:	an = Achsnummer
RÜCKGABEWERT:	In value wird die Anzahl der Bewegungskommandos im Spooler der entsprechenden Achse zurückgeliefert.
KOMMENTAR:	Diese Funktionalität ist erst ab Versionen nach Mai 2002 verfügbar.
ANMERKUNG:	Mit Hilfe dieses Kommandos kann der aktuelle Bearbeitungsstand im Spooler ermittelt werden.

4.4.96 rdmcp, read motor command port

BESCHREIBUNG:	Mit diesem Befehl kann die aktuelle Stellgröße der Motor-Command-Ports eingelesen werden.
BORLAND DELPHI:	procedure rdmcp(var tsrp:TSRP);
C:	void rdmcp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdmcp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].mcp
RÜCKGABEWERT:	Der Rückgabewert steht nach Ausführung des Befehls im Feld *mcp* zur Verfügung.
MCU-3000 / APCI-8001: **MCU-3100 / APCI-8008:** **MCU-3400C/CPCI-8004:**	Bei Servo-Achsen wird ein Wert im Bereich -32767 .. 32767 zurückgeliefert. Dies entspricht einer Sollwertausgangsspannung von ca. -10V .. +10V. Bei Schrittmotorachsen handelt es sich bei diesem Wert um eine Verzögerungszeit, die für die ausgegebene Schrittfrequenz maßgebend ist. Die Verzögerungszeit kann wie folgt in die Einheit [s] umgerechnet werden: `tver = (mcp+1) * 2 / CLOCK;` *Beispiel: mit mcp = 12499 und CLOCK = 70MHz* *wird tver = 0.333ms und f = 3kHz* Nach jedem Ablauf der Verzögerungszeit tver wird das Puls-Signal umgeschaltet, d.h. nach 2*tver wird ein Schrittsignal mit f = 1 / (2*tver) [Hz] ausgegeben. Der in mcp zurückgelieferte Wert liegt im Bereich von -1048575 .. +1048575. Das Vorzeichen bestimmt die aktuelle Drehrichtung, d.h für die Berechnung von tver ist nur der Betrag von mcp heranzuziehen. Sofern in mcp der Wert 0 zurückgeliefert wird, bedeutet dies, dass kein Schrittsignal ausgegeben wird, d.h. der Motor steht
MCU-6000 / APCI-8401:	Bei Servo-Achsen wird ein Wert im Bereich -2047 .. 2047 zurückgeliefert. Dies entspricht einer Sollwertausgangsspannung von ca. -10V .. +10V. Bei Schrittmotorachsen handelt es sich bei diesem Wert um eine Verzögerungszeit, die für die ausgegebene Schrittfrequenz maßgebend ist. Die Verzögerungszeit kann wie folgt in die Einheit [s] umgerechnet werden: `tver = (mcp+1) * 16 / CLOCK;` *Beispiel: mit mcp = 1874 und CLOCK = 30MHz* *wird tver = 1ms und f = 1kHz* Nach jedem Ablauf der Verzögerungszeit tver wird das Puls-Signal umgeschaltet, d.h. nach 2*tver wird ein Schrittsignal mit f = 1 / (2*tver) [Hz] ausgegeben. Der in mcp zurückgelieferte Wert liegt im Bereich von -32767 .. +32767. Das Vorzeichen bestimmt die aktuelle Drehrichtung, d.h für die Berechnung von tver ist nur der Betrag von mcp heranzuziehen. Sofern in mcp der Wert 0 zurückgeliefert wird, bedeutet dies, dass kein Schrittsignal ausgegeben wird, d.h. der Motor steht

4.4.97 rdMDVel – Read Maximum Velocity Skip

BESCHREIBUNG:	Mit diesem Befehl kann der maximale achsspezifische Geschwindigkeitssprung (*MDVEL*) gelesen werden. Dieser Wert wird von der Look-Ahead Funktionalität der RWMOS-Betriebssystemsoftware verwendet, um die Bahngeschwindigkeit derart zu begrenzen, dass keine der an einer Interpolation beteiligten Achsen, ihren maximal erlaubten Geschwindigkeitssprung überschreitet.
BORLAND DELPHI:	rdMDVel (an: integer; var value: double);
C:	void rdMDVel (long an, double *value);
VISUAL BASIC:	Sub rdMDVel (an As Long, ByVal value As Double)
PARAMETER:	In *an* wird die Achsnummer angegeben, in *value* wird der maximal erlaubte Geschwindigkeitssprung zurückgeliefert. Dieser wird stets in der interpolationsspezifischen Geschwindigkeitseinheit interpretiert.
RÜCKGABEWERT:	keiner
ANMERKUNG:	siehe Kapitel 4.4.174 und 6.3.3

4.4.98 rdModeReg – Read MODEREG

BESCHREIBUNG:	Mit diesem Befehl wird das Register MODEREG der RWMOS-Betriebssystemsoftware gelesen.
BORLAND DELPHI:	procedure rdModeReg (var value: integer);
C:	void rdModeReg(long *value);
VISUAL BASIC:	Sub rdModeReg (value As Long)
PARAMETER:	in value wird ModeReg zurückgeliefert
ANMERKUNG:	Siehe hierzu auch Kapitel 6.3.1.5 und Funktion wrModeReg Kapitel 4.4.175.

4.4.99 rdmpe, read maximum position error

BESCHREIBUNG:	Diese Funktion liefert den achsspezifischen Schleppfehlergrenzwert zurück.
BORLAND DELPHI:	procedure rdmpe(var tsrp:TSRP);
C:	void rdmpe(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdmpe(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].mpe
ANMERKUNG:	Nach Ausführen der Funktion steht der maximal erlaubte Schleppfehler im Register *mpe* in der achsspezifischen Positionseinheit zur Verfügung. PCAP-Befehl *wrmpe()*

4.4.100 rdnfrax – read No-Feed-Rate-Axis

BESCHREIBUNG:	Mit diesem Befehl wird das Register NFRAX der RWMOS-Betriebssystemsoftware gelesen.
BORLAND DELPHI:	rdnfrax (var value: integer);
C:	void rdnfrax (long *value);
VISUAL BASIC:	Sub rdnfrax (ByVal value As Long)
PARAMETER:	in value wird NFRAX zurückgeliefert
ANMERKUNG:	Siehe hierzu auch Kapitel 6.3.1 und Funktion wrnfrax Kapitel 4.4.177.

4.4.101 rdPosErr, read Position Error

BESCHREIBUNG:	Diese Funktion liefert den achsspezifischen Schleppfehler des MCU-G3-Istwert-Kanals zurück.
BORLAND DELPHI:	procedure rdPosErr (var an: integer; var value: double);
C:	void rdPosErr (long an, double *value)
VISUAL BASIC:	Sub rdPosErr (an As Long, value As Double)
PARAMETER:	*an* = Achnummer (0..n) *value* = gelesener Schleppfehler
RÜCKGABEWERT:	Nach Ausführen der Funktion, steht der Schleppfehler der Achse *an* in der Variablen *value* in der achsspezifischen Positionseinheit zur Verfügung.
ANMERKUNG:	Der Schleppfehler kann nicht geschrieben werden. [Kapitel 6.3.3]

4.4.102 rdPcapIndex

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische Variable PcapIndex gelesen werden.
BORLAND DELPHI:	function rdPcapIndex (an: integer; var PcapIndex: integer): integer;
C:	int rdPcapIndex (int an, int * PcapIndex);
VISUAL BASIC:	Function rdPcapIndex (ByVal an As Long, PcapIndex As Long) As Long
Parameter:	Mit *an* wird der auszulesende Achskanal angegeben (0, 1, ...). In *PcapIndex* wird der zu lesende Indexwert zurückgegeben.
Rückgabewert:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Siehe hierzu auch smlai, smlri, spda, spdr, ssfi, ssfni.

4.4.103 rdrp, read real position

BESCHREIBUNG:	Diese Funktion liefert die achsspezifische aktuelle Position (= tatsächliche Position oder real position) zurück. Das Auslesen der Position kann zu jedem beliebigen Zeitpunkt, also auch während dem Verfahren der Achse, durchgeführt werden. Pro Abtastzyklus (1.28ms) steht ein neuer Istwert zur Verfügung.
BORLAND DELPHI:	procedure rdrp (var tsrp:TSRP);
C:	void rdrp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdrp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].rp
ANMERKUNG:	Nach Ausführen der Funktion steht die aktuelle Position im Register *rp* in der achsspezifischen Positionseinheit zur Verfügung.

4.4.104 rdrpd – read real position in display unit

BESCHREIBUNG:	Diese Funktion liefert die achsspezifische aktuelle Position (= tatsächliche Position oder real position) achsspezifischen Anzeigeeinheit (display unit) zurück. Das Auslesen der Position kann zu jedem beliebigen Zeitpunkt, also auch während dem Verfahren der Achse, durchgeführt werden. Pro Abtastzyklus (1.28ms) steht ein neuer Istwert zur Verfügung.
BORLAND DELPHI:	procedure rdrpd(var tsrp:TSRP);
C:	void rdrpd(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdrpd(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].rp
RÜCKGABEWERT:	keiner
WIRKUNG:	Nach Ausführen der Funktion steht die aktuelle Position im Feld *rp* in der achsspezifischen Anzeigeeinheit zur Verfügung.
ANMERKUNG:	siehe auch Kommandos rddp, rdrp, rddpd

4.4.105 rdrv, read real velocity

BESCHREIBUNG:	Diese Funktion liefert die achsspezifische Ist-Geschwindigkeit des MCU-G3-Istwert-Kanals zurück.
BORLAND DELPHI:	procedure rdrv (var an: integer; var value: double);
C:	void rddv (int *an, double *value);
VISUAL BASIC:	Sub rddv (an As Long, value As Double)
PARAMETER:	*an* = Achnummer (0..n) *value* = gelesener Geschwindigkeitswert
RÜCKGABEWERT:	Nach Ausführen der Funktion, steht die Istgeschwindigkeit in der Variablen *value* in der achsspezifischen Geschwindigkeitseinheit zur Verfügung.
ANMERKUNG:	Die Istgeschwindigkeit kann nicht geschrieben, jedoch auch bei geöffnetem Regelkreis gelesen werden.

4.4.106 rdSampleTime – Read Sample Time

BESCHREIBUNG:	Mit diesem Befehl kann die Abtast-Zykluszeit der Steuerung ermittelt werden.
BORLAND DELPHI:	function rdSampleTime (var value: integer) as integer;
C:	void rdSampleTime(long *value);
VISUAL BASIC:	Function rdSampleTime (ByVal value As Long) As Long
RÜCKGABEWERT:	1 bei Erfolg, 0 bei Misserfolg, wenn z.B. RWMOS.ELF diese Funktion noch nicht unterstützt
PARAMETER:	in value wird die Abtastzeit in µs zurückgeliefert
ANMERKUNG:	Die Abtast-Zyklus-Zeit wird als Ganzzahl in µs angezeigt. Defaultwert ist 1280.

4.4.107 rdsdec, read stop deceleration

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische Stopverzögerung *sdec* eingelesen werden. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure rdsdec(var tsrp:TSRP);
C:	void rdsdec(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdsdec(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].sdec
RÜCKGABEWERT:	Nach Ausführung des Befehls steht die Stopverzögerung im Feld *sdec* zur Verfügung. Der Wert wird in der achsspezifischen Beschleunigungseinheit zurückgeliefert.
ANMERKUNG:	Die Stopverzögerung kann unter anderem mit dem PCAP-Befehl *wrsdec()* jederzeit neu gesetzt werden.

4.4.108 rdsll, read software limit left

BESCHREIBUNG:	Diese Funktion liefert die achsspezifische linke Software-Endlagen-Position zurück.
BORLAND DELPHI:	procedure rdsll(var tsrp:TSRP);
C:	void rdsll(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdsll(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].sll
ANMERKUNG:	Nach Ausführen der Funktion steht die linke Software-Endlagen-Position im Register *sll* in der achsspezifischen Positionseinheit zur Verfügung. PCAP-Befehl *wrsll()*

4.4.109 rdslr, read software limit right

BESCHREIBUNG:	Diese Funktion liefert die achsspezifische rechte Software-Endlagen-Position zurück.
BORLAND DELPHI:	procedure rdslr(var tsrp:TSRP);
C:	void rdslr(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdslr(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].slr
ANMERKUNG:	Nach Ausführen der Funktion steht die rechte Software-Endlagen-Position im Register *slr* in der achsspezifischen Positionseinheit zur Verfügung. PCAP-Befehl *wrslr()*

4.4.110 rdslsp, read Slits / Stepperpulses

BESCHREIBUNG:	Mit dieser Funktion kann die achsspezifische Auflösung pro Motorumdrehung {slsp} bei Enkoder- oder Schrittmotorsystemen ermittelt werden. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	function rdslsp (an: integer; var value: double): integer;
C:	int rdslsp (long an, double *value);
VISUAL BASIC:	Function rdslsp (ByVal an As Long, value As Double) As Long
TSRP-KOMPONENTEN:	keine
RÜCKGABEWERT:	1 bei Erfolg, 0 bei Mißerfolg, z.B. wenn die Funktion von RWMOS.ELF noch nicht unterstützt wird. Nach erfolgreicher Ausführung der Funktion, steht der Faktor slsp in value in den Einheiten zur Verfügung, die in mcfg gewählt wurden.
ANMERKUNG:	slsp kann mit dem PCAP-Befehl *wrslsp()* gesetzt werden. Siehe hierzu auch Achsenqualifizierer slsp.

4.4.111 rdtp, read target position

BESCHREIBUNG:	Mit Hilfe dieser Funktion kann die Zielposition (target position) achsspezifisch abgefragt werden. Die Zielposition wird immer als absolute Weg- bzw. Winkelgröße zurückgeliefert.
BORLAND DELPHI:	procedure rdtp(var tsrp:TSRP);
C:	void rdtp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdtp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].tp
ANMERKUNG:	Nach Ausführen der Funktion steht die Zielposition des letzten Verfahrbefehls im Register *tp* in der achsspezifischen Positionseinheit zur Verfügung. Der Befehl dient nur zu Kontrollzwecken.

4.4.112 rdtpd – read target position in display unit

BESCHREIBUNG:	Mit Hilfe dieser Funktion kann die Zielposition (target position) in der achsspezifischen Anzeigeeinheit (display unit) abgefragt werden. Die Zielposition wird immer als absoluter Positionswert zurückgeliefert.
BORLAND DELPHI:	procedure rdtpd(var tsrp:TSRP);
C:	void rdtpd(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rdtpd(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].tp
RÜCKGABEWERT:	keiner
WIRKUNG:	Nach Ausführen der Funktion steht die Zielposition des letzten Verfahrbefehls im Register *tp* in der achsspezifischen Anzeigeeinheit zur Verfügung. Der Befehl dient nur zu Kontrollzwecken.
ANMERKUNG:	siehe auch Kommandos rdtp, rddp, rdrp, rdrpd, rddpd

4.4.113 rdtrac, read trajectory acceleration

BESCHREIBUNG:	Lesen der RWMOS Systemvariablen TRAC
BORLAND DELPHI:	function rdtrac (var value:double) : integer;
C:	int rdtrac (double *value);
VISUAL BASIC:	Function rdtrac (value As Double) as long
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	TRAC ist die Interpolations Bahnbeschleunigung, die bei Interpolationsbefehlen, welche aus der rw_SymPas Programmierumgebung aufgerufen werden, herangezogen wird (siehe auch *rw_SymPas* System-Parameter in Tabelle 34).

4.4.114 rdtrovr, read trajectory override

BESCHREIBUNG:	Dieser Befehl liest eine Zwischengröße des aktuell gesetzten Bahngeschwindigkeitskorrekturwertes, welcher bei allen Interpolationsbefehlen (*move*-Befehlen) und den entsprechend selektierten Achsen (PCAP-Befehl *utrovr()*) berücksichtigt wird.
BORLAND DELPHI:	procedure rdtrovr(var value:double);
C:	void rdtrovr(double *value);
VISUAL BASIC:	Sub rdtrovr(value As Double)
RÜCKGABEWERT:	Nach Ausführung des Befehls steht der Bahngeschwindigkeitskorrekturwert in der Variablen *value*.
ANMERKUNG:	PCAP-Befehle *utrvr()*, *wrtrovr()*, *wrjovr()*, *rdtrovr()* und *rdjovr()*

4.4.115 rdtrovrst, read trajectory override settling time

BESCHREIBUNG:	Mit diesem Befehl kann die programmierte Override-Settling-Time (siehe wrtrovrst Kapitel 4.4.186) ausgelesen werden.
BORLAND DELPHI:	function rdtrovr(var value:double) : integer;
C:	int rdtrovr(double *value);
VISUAL BASIC:	Function rdtrovr(value As Double) as long
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt , Kommunikation mit der Steuerungsbaugruppe ist unterbrochen Die gesetzte Override-Settling-Time wird in *value* zurückgeliefert.
ANMERKUNG:	PCAP-Befehl wrtrovrst

4.4.116 rdtrvl, read trajectory velocity

BESCHREIBUNG:	Lesen der RWMOS Systemvariablen TRVL
BORLAND DELPHI:	function rdtrvl (var value:double) : integer;
C:	int rdtrvl (double *value);
VISUAL BASIC:	Function rdtrvl (value As Double) as long
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	TRVL ist die Interpolations Bahngeschwindigkeit, die bei Interpolationsbefehlen, welche aus der rw_SymPas Programmierumgebung aufgerufen werden, herangezogen wird (siehe auch *rw_SymPas* System-Parameter in Tabelle 34).

4.4.117 rdtrtvl, read trajectory target velocity

BESCHREIBUNG:	Lesen der RWMOS Systemvariablen TRTVL
BORLAND DELPHI:	function rdtrtvl (var value:double) : integer;
C:	int rdtrtvl (double *value);
VISUAL BASIC:	Function rdtrtvl (value As Double) as long
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	TRTVL ist die Interpolations Bahn-Zielgeschwindigkeit, die bei Interpolationsbefehlen, welche aus der rw_SymPas Programmierumgebung aufgerufen werden, herangezogen wird (siehe auch *rw_SymPas* System-Parameter in Tabelle 34).

4.4.118 rdZeroOffset, read zero offset

BESCHREIBUNG:	Mit Hilfe dieses Befehls kann die aktuell gesetzte achsspezifische Nullpunktverschiebung (zero offset) gelesen werden. Der Absolutwert der aktuell gesetzen Nullpunktverschiebung wird in *value* in der achsspezifischen Positionseinheit zurückgeliefert. Mit dem Parameter *an* wird der Achsindex des auszulesenden Achskanals (0..n) angegeben.
BORLAND DELPHI:	function rdZeroOffset (an: integer; var value: double) : integer;
C:	int rdZeroOffset (integer an, double *value);
VISUAL BASIC:	Function rdZeroOffset (ByVal an As Long, value As Double) As Long
Rückgabewert:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Die Nullpunktverschiebung kann beispielsweise mit den PCAP-Befehlen szpa (siehe Kapitel 4.4.135) oder szpr (siehe Kapitel 4.4.136) gesetzt werden.

4.4.119 rifs, reset interface status register

Funktionsbeschreibung gilt nur für MCU-3000 / MCU-3100 / MCU-3400C

BESCHREIBUNG:	Mit diesem Befehl können verschiedene Fehlerflags im MCU-G3 Interface-Status-Register *ifs* rückgesetzt werden. Im Detail sind dies die Fehlerbits 16, 17, 18 - *pfe*, *wdog*, und *iae*. Das Rücksetzen sollte nur in Ausnahmesituationen, z.B. in einer Fehlerüberwachungsroutine, ausgeführt werden.
BORLAND DELPHI:	procedure rifs(var tsrp:TSRP);
C:	void rifs(struct TSRP far *tsrp);
VISUAL BASIC:	Sub rifs(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].ifs
ANMERKUNG:	[Kapitel 4.4.78 - *rdifs()*]

4.4.120 RPtoDP, Real-Position to Desired-Position

BESCHREIBUNG:	Mit Hilfe dieses Befehles, kann die Istposition einer Achse {rp} in die Sollposition {dp} übernommen werden. Dieser Befehl wird ohne Laufzeitverzögerung durchgeführt. Die Ausführung des Befehls ist aber nur wirksam, wenn sich die betreffenden Achsen nicht in einen Verfahrprofil befinden, da ansonsten die Sollposition sofort wieder durch den berechneten Wert der Profilgenerierung ersetzt wird. Es ist aber durchaus möglich, eine verfahrende Achse, die sich mit einer Zielgeschwindigkeit ≠ 0 bewegt zu korrigieren. Parameter sind die betreffenden Achsen.
BORLAND DELPHI:	procedure RPtoDP(var as: AS);
C:	void RPtoDP (struct AS far *as);
VISUAL BASIC:	Sub RPtoDP (DASEL As ASEL)
RÜCKGABEWERT:	keiner
ANMERKUNG:	Dieses Kommando kann verwendet werden, wenn eine oder mehrere Achse sich nicht mehr im Regelungseingriff befinden, weil sich ein Schleppfehler z.B. durch Blockierung der Achsen aufgebaut hat. Nach Aufhebung der Ursache kann dann die Regelung an der aktuellen Position, auch bei verfahrenden Achsen, nahtlos wieder aufgenommen werden. Dieses Kommando ist verfügbar ab RWMOS.ELF V2.5.3.100 und mcug3.dll V2.5.3.80.

4.4.121 rs, reset system

BESCHREIBUNG:	Dieser Befehl bewirkt das Rücksetzen des kompletten Achssystems. Die Digital-Ausgänge werden auf die, mit Hilfe des TOOLSET-Programms *mcfg.exe* projektierten Defaultwerte gesetzt. Auf den Sollwertkanälen werden bei Servo-Achsen 0V Ausgangsspannung und bei Schrittmotor-Achsen 0Hz Schrittfrequenz ausgegeben. Bei allen Achsen wird der Lageregelkreis geöffnet. Die Spooler-Daten werden komplett verworfen. Alle CNC-Tasks werden angehalten. Alle projektierten Softwareendlagen werden nicht mehr überwacht. Alle Overridefaktoren (PCAP-Befehl *wrjovr()* und *wrtrovr()*) werden auf den Wert 1.0 gesetzt.
BORLAND DELPHI:	procedure rs;
C:	void rs(void);
VISUAL BASIC:	Sub rs()
ANMERKUNG:	Alle Systemdaten wie Beschleunigungen, Geschwindigkeiten, Filterparameter usw. bleiben gespeichert und brauchen deshalb nicht neu geladen zu werden. Die Statusflags des *ifs*-Registers werden durch diesen Befehl nicht beeinflußt. Der Inhalt aller Common Integer und Double-Variablen bleibt erhalten.

4.4.122 scp – set controller params

BESCHREIBUNG:	Diese Funktion wird für kundenspezifische Erweiterungen verwendet und dient dazu ein Parameterfeld von 15 x 15 Gleitpunktzahlen (vordefinierte Daten-Struktur CTRLRPARAMS) achsspezifisch an die Steuerung zu übertragen.
BORLAND DELPHI:	procedure scp (an: integer; var ctrlrparams: CTRLRPARAMS);
C:	void scp (long an, struct CTRLRPARAMS *ctrlrparams);
VISUAL BASIC:	Sub scp (ByVal an As Long, DCTRLRPARAMS As CTRLRPARAMS)
RÜCKGABEWERT:	keiner
WIRKUNG:	Die Werte in CTRLRPARAMS werden an die Steuerung übertragen.
ANMERKUNG:	Verwendung und Bedeutung der zu übergebenden Daten werden applikationsspezifisch dokumentiert.

4.4.123 sdels, spooler delete synchronous

BESCHREIBUNG:	Alle im Spooler eingetragenen Kommandos werden verworfen. Der gesamte Spoolbereich steht wieder zur freien Verfügung. Das Verwerfen der Spoolerdaten erfolgt für die in *AS* spezifizierten Achsen.
BORLAND DELPHI:	procedure sdels(var as:AS);
C:	void sdels(struct AS far *as);
VISUAL BASIC:	Sub sdels(DASEL As ASEL)
ANMERKUNG:	Die aktuelle Operation, wie z.B. ein Verfahrbefehl, wird fertig ausgeführt.

4.4.124 shp, set home position

BESCHREIBUNG:	Mit Hilfe dieses Befehls kann der achsspezifische Nullpunkt (home position) gesetzt werden. Der Parameter *tp* wird in der achsspezifischen Positionseinheit angegeben. Der Befehl wird im allgemeinen nach einem Referenzsuchlauf zum Setzen des Maschinennullpunktes verwendet. Er kann in beiden Betriebsarten Regelkreis geöffnet und Regelkreis geschlossen ausgeführt werden. Um ruckartige Motorbewegungen zu verhindern, sollte er jedoch nicht während dem Verfahren des selektierten Achskanals verwendet werden.
BORLAND DELPHI:	procedure shp(var tsrp:TSRP);
C:	void shp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub shp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].tp n = 0 .. Anzahl der vorhandenen Achsen-1
ANMERKUNG:	Bis zur ersten Ausführung dieses Befehls werden die projektierten Softwareendlagen nicht überwacht. Dies bedeutet, dass vor der Ausführung des *shp()*-Kommandos eine Referenzfahrt unter Verwendung aller *move*- und *jog*-Befehle durchgeführt werden kann. Die Software-Endlagen werden nach Ausführung des *shp()*-Kommandos bis zum nächsten *ra()* bzw. *RA()* oder *rs()* bzw. *RS*-Kommando überwacht. Die Bereitschaft zur Überwachung von Softwareendlagen wird mit dem Bit ref im axst-Register angezeigt. Das Kommando shp setzt die Istposition rp auf den angegebenen Wert, wobei eine eventuelle Verschiebung durch einen "backlash"-Wert erhalten bleibt. Ein eventueller Positionsoffset aufgrund eines Wertes in "dpoffset" bleibt in der Istposition unberücksichtigt, wirkt sich aber in dem sich neu ergebenden Wert der Sollposition (dp) aus. shp darf nicht gerufen werden während der Eintragung von Verfahrbefehlen in den Spooler insbesondere nicht bei Kommandos die werkzeugradiuskorrigiert werden oder bei Spline Befehlen.

4.4.125 spd, Spool Position Data

BESCHREIBUNG:	Mit Hilfe dem *spd* Kommando können Positionswerte gespoolt werden, welche jeweils abtastsynchron als Sollpositionswerte übernommen werden. Durch Aufruf dieses Kommandos für mehrere Achsen mit gleicher Satzanzahl ist eine interpolierte Verfahrbewegung möglich.
BORLAND DELPHI:	procedure spda (an:integer; size:integer; var spdbuf:SPDBUF);
C:	void spda (int an, int size, struct SPDBUF *spdbuf);
VISUAL BASIC:	Sub spda (ByVal an As Long, ByVal size As Long, SPDBUF As SPDBUF)
PARAMETER:	*an* ist der Index der anzusprechenden Achse. In *size* wird die Anzahl der Stützpunkte angegeben. *size* darf Werte zwischen 1 und 1000 annehmen. *spdbuf* ist ein Array, in dem die Positions-Stützpunkte übergeben werden.
ANMERKUNGEN:	• In diesem Zusammenhang ist das Kommando *sstvl* zu beachten, wenn ohne Zwischenstopp ein Übergang von einer Trajektorie in eine Kontur bestehend aus *spd* Kommandos stattfinden soll. • Dieses Kommando ist im wesentlichen gleich wie *spda*, jedoch wird hier kein PcapIndex übergeben.

.

4.4.126 spda, Spool Position Data Absolute

BESCHREIBUNG:	Mit Hilfe dem *spda* Kommando können Positionswerte gespoolt werden, welche jeweils Abtastsynchron als Sollpositionswerte übernommen werden. Durch Aufruf dieses Kommandos für mehrere Achsen mit gleicher Satzanzahl ist eine interpolierte Verfahrbewegung möglich. Der Ausführungszustand kann mit Hilfe eines Index ermittelt werden.
BORLAND DELPHI:	procedure spda (an:integer; size:integer; var spdbuf:SPDBUF ; PcapIndex: integer);
C:	void spda (int an, int size, struct SPDBUF *spdbuf, long PcapIndex);
VISUAL BASIC:	Sub spda (ByVal an As Long, ByVal size As Long, SPDBUF As SPDBUF, ByVal PcapIndex As Long)
PARAMETER:	*an* ist der Index der anzusprechenden Achse. In *size* wird die Anzahl der Stützpunkte angegeben. *size* darf Werte zwischen 1 und 1000 annehmen. *spdbuf* ist ein Array, in dem die Positions-Stützpunkte übergeben werden. In *PcapIndex* wird ein Index zur Identifikation des aktuellen Standes der Ausführung des Kommandos übergeben. Bei jedem Stützpunktwert wird dieser Index bei der Anzeige inkrementiert. Der Index kann mit Hilfe der Ressource PcapIndex (# 32) gelesen werden.
ANMERKUNG:	In diesem Zusammenhang ist das Kommando *sstvl* zu beachten, wenn ohne Zwischenstopp ein Übergang von einer Trajektorie in eine Kontur bestehend aus *spda* Kommandos stattfinden soll.

4.4.127 spdr, Spool Position Data Relative

Dieses Kommando ist im wesentlichen gleich wie *spda*, jedoch sind die Positionswerte in *spdbuf* Relativkoordinaten.

4.4.128 ssms, start spooled motions synchronous

BESCHREIBUNG:	Mit Hilfe von *spool*-Befehlen können Kommandos an die einzelnen Achskanäle der MCU-G3 übertragen werden. Diese werden in einer Warteschlange eingetragen. Der PCAP-Befehl *ssms()* veranlaßt den Synchronstart für die Spoolerbefehlsabarbeitung aller in *AS* spezifizierten Achsen.
BORLAND DELPHI:	procedure ssms(var as:AS);
C:	void ssms(struct AS far *as);
VISUAL BASIC:	Sub ssms(DASEL As ASEL)
ANMERKUNG:	Kapitel 2.2.8.2 - Spool-Modus

4.4.129 sstps, spooler stop synchronous

BESCHREIBUNG:	Mit Hilfe dieses Befehls wird die Befehlsabarbeitung aus dem Spooler aller in *AS* angewählten Achskanäle unterbrochen.
BORLAND DELPHI:	procedure sstps(var as:AS);
C:	void sstps(struct AS far *as);
VISUAL BASIC:	Sub sstps(DASEL As ASEL)
ANMERKUNG:	Der aktuelle Befehl wird komplett abgearbeitet. Die Befehle, die sich im Spooler befinden bleiben erhalten und können mit SSMS fortgesetzt werden. Vorsicht ist jedoch geboten, wenn sich im Spooler Verfahrkommandos mit Zielgeschwindigkeiten ≠ 0 befinden. In Fehlersituationen, wenn die Spoolerinhalte verworfen werden sollen ist es besser direkte Kommandos zum Stoppen der betroffenen Achsen zu verwenden (ms oder js), da diese Kommandos die aktuelle Kontur unterbrechen und den Spoolerinhalt gleichzeitig verwerfen.

4.4.130 sstvl, Spooler Set Target Velocity

BESCHREIBUNG:	Mit Hilfe des Befehls sstvl kann die Zielgeschwindigkeit einer im Spooler vorliegenden Trajektorie aller in *AS* spezifizierten Achsen auf einen definierten Wert gesetzt werden. Die Richtung der Zielgeschwindigkeit entspricht der Richung des letzten Verfahrkommandos. Die angegebene Zielgeschwindigkeit muss erreichbar sein und darf nicht durch Systemeinstellungen wie z.b. MdVel, MaxVel oder MaxAcc begrenzt sein. Sonst wird der Zielgeschwindigkeitswert entsprechend reduziert.
BORLAND DELPHI:	procedure sstvl(var as:AS, tvl: double);
C:	void sstvl (struct AS far *as, double tvl);
VISUAL BASIC:	Sub sstvl(DASEL As ASEL, ByVal tvl As Double)
PARAMETER:	In *as* werden die zu bearbeitenden Achsen definiert. In *tvl* wird die gewünschte Bahngeschwindigkeit übergeben.
ANMERKUNG:	Dieses Kommando kann verwendet werden um eine zuvor programmierte Trajektorie mit SPD Kommandos (spd, spda oder spdr) ohne Geschwindigkeitseinbruch fortzusetzen. Ohne dieses Kommando würde die zuvor programierte Kontur bei aktivem LookAhead am Positionsübergang auf die Geschwindigkeit 0 heruntergefahren werden.

4.4.131 ssf, Spool-Special-Function

BESCHREIBUNG:	Dieses Kommando ermöglicht dem Anwender auch andere Kommandos, als Verfahrbefehle im Spooler einzutragen. Mit dem Parameter *command* wird das auszuführende Kommando eingetragen.
BORLAND DELPHI:	procedure ssf(an: integer; command: integer; value: double); far; stdcall;
C:	void ssf(int axis, int command, double value);
VISUAL BASIC:	Sub ssf(ByVal an As Long, ByVal command As Long, ByVal value As Double)
AUFRUF-PARAMETER:	Der Wert *value* wird als Parameter bei der in *axis* angegebenen Achse eingetragen. Folgende Kommandos sind derzeit verfügbar:

Command	Beschreibung
0 .. 999	CI-Variable mit *Value* beschreiben.
1000	Spoolerabarbeitung anhalten, diese Anweisung wird nur dann ausgeführt, wenn die Zielgeschwindigkeit des letzten eingetragenen Profils gleich 0 ist
1001	Digitale Ausgänge setzen, die zu setzenden Ausgänge werden in *Value* bitweise angegeben.
1002	Digitale Ausgänge rücksetzen, die rückzusetzenden Ausgänge werden in *Value* bitweise angegeben.
1003	Spoolerabarbeitung anhalten für die in *Value* angegebene Zeit, Zeiteinheit ist 64 µs. Die tatsächliche Wartezeit wird auf Vielfache der Abtastzeit abgerundet. Diese Anweisung wird nur dann ausgeführt, wenn die Zielgeschwindigkeit des letzten eingetragenen Profils gleich 0 ist. Der Wartevorgang kann z.B.mit der Anweisung SSMS vorzeitig beendet werden.

Command	Beschreibung
1004	Spoolerabarbeitung anhalten bis die Eingänge aktiv sind, die in *Value* angegeben sind. Die Eingänge werden bitcodiert angegeben. Diese Anweisung wird nur dann ausgeführt, wenn die Zielgeschwindigkeit des letzten eingetragenen Profils gleich 0 ist. Der Wartevorgang kann z.B. mit der Anweisung SSMS vorzeitig beendet werden. Es können stets nur Eingänge der jeweiligen Achsgruppe angegeben werden.
1005	Spoolerabarbeitung anhalten bis die Common-Variable CI99 den Wert 0 enthält. Der in *Value* angegebene Wert wird zuvor in CI99 eingetragen. Wenn dieses Kommando verwendet werden soll, ist CI99 somit vorbelegt und darf nicht für andere Zwecke verwendet werden. (Siehe hierzu auch Kap. 4.4.131.1) **Hinweis:** das Ablöschen von CI99 **muss** mit dem PCAP-Kommando ClearCI99 erfolgen, da die Achsen ansonsten asynchron gestartet werden können.
1006	Spoolerabarbeitung anhalten, bis bei allen, in *Value* bitcodiert angegebenen Achsen dieses Kommando mit dem gleichen Parameter aktiviert bzw. ausgeführt wurde. Mit Hilfe dieses Kommandos ist es möglich, die Spoolerabarbeitung von verschiedenen Achsen zu synchronisieren. (Siehe hierzu auch Kap. 4.4.131.1)
1015	Der Parameter value wird zu CI99 hinzuaddiert. Danach wird die Spoolerabarbeitung angehalten bis CI99 den Wert 0 enthält. Der Wartebefehl wird nur ausgeführt, wenn die Zielgeschwindigkeit des vorherigen Profils gleich 0 ist. (Siehe hierzu auch Kap. 4.4.131.1) **Hinweis:** das Ablöschen von CI99 **muss** mit dem PCAP-Kommando ClearCI99 erfolgen, da die Achsen ansonsten asynchron gestartet werden können.

1025	In der Variablen CI99 wird das der Achse zugeordnete Bit gesetzt. Danach wird die Spoolerabarbeitung angehalten bis dieses Bit in CI99 wieder rückgesetzt ist. Der Wartebefehl wird nur ausgeführt, wenn die Zielgeschwindigkeit des vorherigen Profils gleich 0 ist. (Siehe hierzu auch Kap. 4.4.131.1) **Hinweis:** das Ablöschen von CI99 **muss** mit dem PCAP-Kommando ClearCI99 erfolgen, da die Achsen ansonsten asynchron gestartet werden können.

1101	PC-Interrupt-Anforderung aktivieren
1200 ... 1207	Motor-Command-Port mcp einer Achse im System mit dem Index 0..7 beschreiben. 1200 = 1. Achse, 1201 = 2. Achse usw.
2001	Zielgeschwindigkeit im letzten gespoolten Verfahrprofil abnullen.
10000 ... 10999	Bits in CI-Variable setzen. Die zu setzenden Bits sind in *Value* angegeben.

Command	Beschreibung
11000 ... 11999	Bits in CI-Variable rücksetzen. Die zurückzusetzenden Bits sind in *Value* angegeben.
20000 ... 20999	CD-Variable mit *Value* beschreiben.

4.4.131.1 Hinweise zu SSF Wartebefehlen

Einige der oben beschreibenen Wartebefehle verwenden die Common-Integer Variable CI99. Hierbei ist zu beachten, dass das Beschreiben dieser Variablen aus der PCAP-Programmierung durch einen direkten PCI-Speicherzugriff erfolgt und somit asynchron zur RWMOS Betriebssystemsoftware. Um eine einwandfreie Synchronität der Achsen nach einer Profilfortsetzung per SSF-Wartebefehl zu erreichen, muss deshalb das DLL Kommando ClearCI99 verwendet werden.
In einigen der oben angegebenen Kommandos werden bitcodierte Angaben z.B. von Eingängen, Ausgängen oder Achsen erwartet. Hierbei sind die entsprechende Bitnummern der jeweiligen Nummer des zu programmierenden Wertes zugeordnet.
So sollen zum Beispiel die Eingänge 1 und 3 bitcodiert angegeben werden: in diesem Fall muss der Hexadezimalwert 5 progammiert werden. Auf diese Weise ist es möglich, mehrere Achsen, Eingänge oder Ausgänge in einem Datenwort anzugeben.

BEISPIELE:

```
ssf(A1, 125, 999);   // CI125 mit dem Wert 999 beschreiben
ssf(A1, 1001, 1);    // Ausgang O1 bei Achse 1 setzen
ssf(A1, 1002, 4);    // Ausgang O3 bei Achse 1 rücksetzen
```

4.4.132 startcnct, start numeric controller task

BESCHREIBUNG:	Ein zuvor geladenes SAP-Programm kann mit diesem Befehl gestartet werden. Die in *TaskNr* (Werte 0..3) angewählte CNC-Task arbeitet das SAP-Programm vom Programmstart an ab. Das Laden kann unter anderem mit dem PCAP-Befehl *txbf2()* erfolgen.
BORLAND DELPHI:	procedure startcnct(TaskNr:integer);
C:	void startcnct(int TaskNr);
VISUAL BASIC:	Sub startcnct(ByVal TaskNr As Long)
ANMERKUNG:	Ein laufendes SAP-Programm wird vor Ausführung dieses Befehls automatisch gestoppt. PCAP-Befehl *txbf2()*

4.4.133 stepcnct, step numeric controller task

BESCHREIBUNG:	Dieser Befehl dient zur zeilenweisen Ausführung eines SAP-Programms.
BORLAND DELPHI:	procedure stepcnct (TaskNr:integer);
C:	void stepcnct(int TaskNr);
VISUAL BASIC:	Sub stepcnct(ByVal TaskNr As Long)
ANMERKUNG:	Der PCAP-Befehl *stepcnct()* führt eine Programmzeile in der angegebenen CNC-Task aus. Wenn die Zeile abgearbeitet ist, wird dies durch den Wert 2 im Element running der Datenstruktur CNCTS angezeigt (Kapitel 4.3.2.10).

4.4.134 stopcnct, stop numeric controller task

BESCHREIBUNG:	Dieser Befehl bewirkt den Programmstop des momentan ablaufenden SAP-Programms in der mit *TaskNr* (Werte 0..3) angewählten CNC-Task und versetzt diese CNC-Task in einen inaktiven Zustand. Das SAP-Programm kann unter anderem mit dem SAP-Befehl *CONTCNCT()* oder dem PCAP-Befehl *contcnct()* wieder fortgesetzt werden.
BORLAND DELPHI:	procedure stopcnct(TaskNr:integer);
C:	void stopcnct(int TaskNr);
VISUAL BASIC:	Sub stopcnct(ByVal TaskNr As Long)
ANMERKUNG:	Eventuell freigegebene EVENT-Handler im SAP-Programm werden nach Ausführen des *stopcnct()*-Befehls nicht mehr abgearbeitet. Der Antrieb sollte vor Ausführung dieses Befehls in einen sicheren Betriebszustand gebracht werden.

4.4.135 szpa, set zero position absolut

BESCHREIBUNG:	Mit Hilfe dieses Befehls kann ein achsspezifischer virtueller Nullpunkt (zero position) gesetzt werden. Der Parameter *Position* wird in der achsspezifischen Positionseinheit angegeben. Mit dem Parameter an wird die Achsnummer angegeben. Der Befehl kann in beiden Betriebsarten Regelkreis geöffnet und Regelkreis geschlossen ausgeführt werden. Um ruckartige Motorbewegungen zu verhindern, sollte er jedoch nicht während dem Verfahren des selektierten Achskanals verwendet werden.
BORLAND DELPHI:	procedure szpa(an: integer; Position: double);
C:	void szpa(int an, double Position);
VISUAL BASIC:	Sub szpa(ByVal an As Long, ByVal Position As Double)
ANMERKUNGEN:	• Durch Aufruf von szpa mit dem Positonswert 0 kann eine etwaig gesetzte Nullpunktverschiebung gelöscht werden. Der aktuell gesetzte Positionswert der Nullpunktverschiebung kann mit dem Kommando rdZeroOffset (Kapitel 4.4.118) gelesen werden. Siehe hierzu auch Bit ClearZeroPosition im Register ModeReg. • szpa (szpr) darf nicht gerufen werden während der Eintragung von Verfahrbefehlen in den Spooler insbesondere nicht bei Kommanods die werkzeugradiuskorrigiert werden oder bei Spline Befehlen.

4.4.136 szpr, set zero position relativ

BESCHREIBUNG:	Mit Hilfe dieses Befehls kann ein achsspezifischer virtueller Nullpunkt (zero position) relativ gesetzt werden. Der Parameter *Position* wird in der achsspezifischen Positionseinheit angegeben. Mit dem Parameter an wird die Achsnummer angegeben. Der Befehl kann in beiden Betriebsarten Regelkreis geöffnet und Regelkreis geschlossen ausgeführt werden. Um ruckartige Motorbewegungen zu verhindern, sollte er jedoch nicht während dem Verfahren des selektierten Achskanals verwendet werden.
BORLAND DELPHI:	procedure szpr(an: integer; Position: double);
C:	void szpr(int an, double Position);
VISUAL BASIC:	Sub szpr(ByVal an As Long, ByVal Position As Double)
ANMERKUNGEN:	• Durch Aufruf von szpa mit dem Positonswert 0 kann eine etwaig gesetzte Nullpunktverschiebung gelöscht werden. Der aktuell gesetzte Positionswert der Nullpunktverschiebung kann mit dem Kommando rdZeroOffset (Kapitel 4.4.118) gelesen werden. Siehe hierzu auch Bit ClearZeroPosition im Register ModeReg. • szpr (szpa) darf nicht gerufen werden während der Eintragung von Verfahrbefehlen in den Spooler insbesondere nicht bei Kommanods die werkzeugradiuskorrigiert werden oder bei Spline Befehlen.

4.4.137 txbf2, transmit binary file

BESCHREIBUNG:

Mit dieser Funktion wird die im String- bzw. Zeichen-Parameter spezifizierte Datei auf die MCU-G3 übertragen. Die angegebene Datei wird zunächst im aktuellen Arbeitsverzeichniss gesucht. Danach werden die Verzeichnisse, die in der Umgebungsvariable PATH angegeben sind durchsucht. In der Funktionslibrary existiert aus Kompatibilitätsgründen zusätzlich zu diesem Kommando die Funktion txbf(), welche aber keine Dateinamen mit Laufwerks- und Pfadinformationen unterstützt. Beim Aufruf von txbf2 werden im wesentlichen zwei spezielle Datei-Typen erlaubt. Dies sind zum einen die Systemdatei *system.dat* (bzw. Dateien mit kompatiblem Aufbau) und zum anderen die aus der IDE oder mit Hilfe des Kommandozeilen-Compilers *ncc.exe* generierten Autocode-Dateien (CNC-Files) mit den Dateierweiterungsnamen .CNC.
Das Übertragen der Systemdatei *system.dat* bewirkt folgendes:
Alle Achskanäle werden mit den achsspezifischen Systemdaten initialisiert. Die Filterkoeffizienten des PIDF-Filters werden, wie beim PCAP-Befehl *uf()*, neu berechnet. Diese Systemdaten können unter anderem im TOOLSET Programm *mcfg.exe* editiert werden. Evtl. zuvor veränderte Systemgrößen, z.B. achsspezifische Geschwindigkeiten, Beschleunigungen usw. werden durch diesen Befehl wieder überschrieben.
Achtung! Das Übertragen von CNC-Files bewirkt folgendes: Der momentane Programm-Arbeitsspeicher einer CNC-Task wird mit dem Inhalt der spezifizierten Autocode-Datei überschrieben. Deshalb wird die entsprechende Task vor dem Ladevorgang automatisch angehalten. Das CNC-File enthält unter anderem die Information, in welche Task es geladen werden muss (Task 0..3). Nachdem das CNC-File erfolgreich übertragen wurde, kann dieses mit dem PCAP-Befehl *startcnct()* oder PCAP-Befehl *STARTCNCT()* gestartet werden.

BORLAND DELPHI:

```
function txbf2(var filename:string):integer;
```

C:

```
int txbf2(char far *filename);
```

VISUAL BASIC:

```
Function txbf2(ByVal filename As String) As Long
```

RÜCKGABEWERT:

Die Funktion kann folgende Werte zurückliefern:

Rückgabe wert	Fehler-Beschreibung
0	kein Fehler
20	Datei konnte nicht geöffnet werden. Möglich Ursachen hierfür sind: - Der Dateiname ist ungültig - Die Datei existiert nicht - Der Pfad und Suchlaufwerk ist ungültig
21	Die Datei ist zu groß für den CNC-Task-Arbeitsspeicher.
22	Ungültiger Datei-Typ! (kein SAP-File oder keine System-Datei)
23	Interner Fehler bei Speicherreservierung.
24	Ungültige Task-Nummer ist angegeben. Bei einer Systemdatei zeigt dieser Fehler an, daß eine ungültige oder beschädigte Systemdatei verwendet wird.
25	Daten-Übertragungsfehler bei Remote-Systemen (WebServices)

ANMERKUNG:

Normalerweise ist das Laden der Systemdatei *system.dat* nur einmalig pro Systemstart notwendig. Hierzu sind auch die Angaben beim PCAP-Befehl *mcuinit()* zu beachten. Im Parameter *filename* können bei Bedarf Laufwerks- und Pfad-Namen spezifiziert werden.

4.4.138 txbfErrorReport, initialision error report

BESCHREIBUNG:	Mit dieser Funktion können die Fehlerrückgabwerte der oben beschriebenen Funktion *txbf2()* im Klartext angezeigt werden. Hierbei wird eine Message-Box am Bildschirm eröffnet, welche wiederum durch den Anwender quittiert werden muß.
BORLAND DELPHI:	procedure txbfErrorReport(filename:PChar; error:integer);
C:	void txbfErrorReport (char *filename, int error);
VISUAL BASIC:	Sub txbfErrorReport (ByVal filename As String, ByVal error As Long)
ANMERKUNG:	PCAP-Befehle InitMcuSystem(), InitMcuSystem2() und InitMcuSystem3()
BEISPIEL:	*txbferror = InitMcuSystem3(...); // Dateiübertragung ausführen* *txbfErrorReport(..., initerror); // Im Fehlerfall Fehlerrückgabewert* *// anzeigen*

4.4.139 uf, update filter

BESCHREIBUNG:	Mit Hilfe dieses Befehls kann das PIDF-Filter der MCU-G3 achsspezifisch gesetzt werden. Bevor der Befehl ausgeführt wird, muss sichergestellt sein, dass alle oben aufgeführten Strukturkomponenten initialisiert sind. Die Ausführung dieses Befehls kann jederzeit, auch während der Profilgenerierung, ausgeführt werden. Diese Eigenschaft ermöglicht eine echtzeitgerechte Anpassung an unterschiedliche Lastverhältnisse.
BORLAND DELPHI:	procedure uf(var tsrp:TSRP);
C:	void uf(struct TSRP far *tsrp);
VISUAL BASIC:	Sub uf(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].kp, TSRP[n].ki, TSRP[n].kd, TSRP[n].kpl, TSRP[n].kfca, TSRP[n].kfcv n = 0 .. Anzahl vorhandener Achsen-1
ANMERKUNG:	Weitere Angaben zum PIDF-Filter sind im Kapitel 2.1.2, [BHB / Kapitel 4.1.1] und [IHB / Kapitel 6.2] enthalten. PCAP-Befehl *rdf()*

4.4.140 utrovr, update trajectroy override

BESCHREIBUNG:	Für alle in *AS* angewählten Achskanäle wird der aktuell gesetzte Geschwindigkeitsoverride berücksichtigt.
BORLAND DELPHI:	procedure utrovr(var as:AS);
C:	void utrovr(struct AS far *as);
VISUAL BASIC:	Sub utrovr(DASEL As ASEL)
ANMERKUNG:	Mit diesem Kommando wird der zuletzt geschriebene Trajektorie-Override-Wert bei den selektierten Achsen übernommen. Wenn das Bit OvrMode im Modereg-Register nicht gesetzt ist, wird dieser Wert in die achsspezifische Variable jovr übernommen. Weitere Informationen sind beim PCAP-Befehl wrtrovr() nachzulesen. Abhängig vom per Funktion wrtrovrst() gesetzten Wert wird der Overridewert nicht schlagartig übernommen, sondern mit der angegebenen Rampenzeit angepasst.

4.4.141 wrasmo, write ASM-2003 outputs

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401	
BESCHREIBUNG:	Mit diesem Register können die Digital-Ausgänge des ASM-2003 gesetzt werden. Zu beachten ist, daß die Digitalausgänge auf dem ASM-2003 achsspezifisch gruppiert sind. Sofern ein Ausgang gesetzt werden soll, wird dies durch Setzen des jeweiligen Bits erreicht. Der bitkodierte Aufbau des *asmo*-Statuswortes kann folgender Tabelle 21 entnommen werden:
TURBO PASCAL:	procedure wrasmo(var tsrp:TSRP);
C:	void wrasmo(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrasmo (DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].asmo

Tabelle 21: Bitkodierter Aufbau des asmo-Wortes

Bit-Nr.	Achskanal 1, 3, 5 ... Funktion Stecker-PIN Reihe (Name)	Achskanal 2, 4, 6 ... Funktion Stecker-PIN Reihe (Name)
0	Ausgang X1-16 Z (O11)	Ausgang X2-02 Z (O21)
1	Ausgang X1-18 Z (O12)	Ausgang X2-04 Z (O22)
2	Ausgang X1-20 Z (O13)	Ausgang X2-06 Z (O23)
3	Ausgang X1-22 Z (O14)	Ausgang X2-08 Z (O24)
4	Ausgang X1-24 Z (O15)	Ausgang X2-10 Z (O25)
5	Ausgang X1-26 Z (O16)	Ausgang X2-12 Z (O26)
6	nicht belegt	nicht belegt
7	nicht belegt	nicht belegt
8	Ausgang X2-14 Z (O17)	Ausgang X2-20 Z (O27)
9	Ausgang X2-16 Z (O18)	Ausgang X2-22 Z (O28)
10	Ausgang X2-18 Z (O19)	Ausgang X2-24 Z (O29)
11..31	Nicht belegt.	

4.4.142 wrasmob, write ASM-2003 output bit

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401	
BESCHREIBUNG:	Mit dieser Funktion kann ein ASM-2003 Digital-Ausgang gesetzt bzw. rückgesetzt werden. Die Achsnummer muß im Parameter *an* (0, 1, ... *REALAXIS*) spezifiziert werden. Das Rücksetzen des Ausgangs erfolgt mit dem Wert 0 bzw. FALSE.
TURBO PASCAL:	procedure wrasmob(an:integer; bitnr:integer; value: integer);
C:	wrasmob(int an, int bitnr, int value);
VISUAL BASIC:	Sub wrasmob (ByVal an As Long, ByVal bitnr As Long, ByVal value As Long)
ANMERKUNG:	PCAP-Befehl *wrasmo()*

Tabelle 22: Zuordnung von *bitnr* zu den jeweiligen ASM-2003 Digitalausgängen

'bitnr'	**Achskanal 1, 3, 5 ... Funktion Stecker-PIN Reihe (Name)**	**Achskanal 2, 4, 6 ... Funktion Stecker-PIN Reihe (Name)**
1	Ausgang X1-16 Z (O11)	Ausgang X2-02 Z (O21)
2	Ausgang X1-18 Z (O12)	Ausgang X2-04 Z (O22)
3	Ausgang X1-20 Z (O13)	Ausgang X2-06 Z (O23)
4	Ausgang X1-22 Z (O14)	Ausgang X2-08 Z (O24)
5	Ausgang X1-24 Z (O15)	Ausgang X2-10 Z (O25)
6	Ausgang X1-26 Z (O16)	Ausgang X2-12 Z (O26)
7	Ausgang X2-14 Z (O17)	Ausgang X2-20 Z (O27)
8	Ausgang X2-16 Z (O18)	Ausgang X2-22 Z (O28)
9	Ausgang X2-18 Z (O19)	Ausgang X2-24 Z (O29)
10..32	Nicht belegt.	

4.4.143 wraux, write auxiliary register

BESCHREIBUNG:	Dieser Befehl setzt das achsspezifische Auxiliary Register auf den in *aux* gesetzten Wert.
BORLAND DELPHI:	procedure wraux (var tsrp:TSRP);
C:	void wraux (struct TSRP far *tsrp);
VISUAL BASIC:	Sub wraux (DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].aux
ANMERKUNG:	siehe auch Kapitel 4.4.52 und 6.3.3

4.4.144 wrcbcnct, write common buffer CNC-Task

BESCHREIBUNG:	Jede CNC-Task hat einen lokalen Speicherbereich, den sogenannten Common-Buffer, der sowohl von der jeweiligen CNC-Task als auch durch ein PCAP-Programm gelesen und beschrieben werden kann. Mit dieser Funktion kann der komplette CNC-Task-spezifische Buffer (oder nur ein Teil davon) beschrieben werden. Mit dem Funktionsparameter *cbcnct* erfolgt die Auswahl des CNC-Task-Buffers, die Anzahl zu schreibender Bytes und die Startadresse des Blocks, der an die MCU-G3 übertragen werden soll.
BORLAND DELPHI:	function wrcbcnct(var cbcnct:CBCNCT):integer;
C:	int wrcbcnct(struct CBCNCT far *cbcnct);
VISUAL BASIC:	Sub wrcbcnct(DCBCNCT As CBCNCT)
RÜCKGABEWERT:	Die Funktion *wrcbcnct()* hat folgenden bitkodierten Rückgabewert Bit-Nr 0 0 kein Fehler 0 1 wenn ungültige Task-Nummer. 1 0 kein Fehler 1 1 wenn maximal erlaubte Buffergröße überschritten Dies bedeutet, dass die Funktion im Normalfall den Wert 0 zurückliefert. 2 0 kein Fehler 2 Adressfehler / Speicherfehler
ANMERKUNG	Die CNC-Task-spezifische Buffergröße beträgt 1000 Bytes. Der Struktur- (Record) Aufbau von CBCNCT ist im Kapitel 4.3.2.9 abgedruckt. PCAP-Befehl *rdcbcnct()*, SAP-Befehle *RDCBx()* und *WRCBx()*

4.4.145 wrcd, write common double

BESCHREIBUNG:	Mit dieser Funktion können Schreibzugriffe auf die sogenannten Common-Variablen, dies sind vordefinierte System-Variablen der CNC-Task, erfolgen. Es handelt sich dabei um die *rw_SymPas*-Variablen CD0 .. CD999. Der erste Parameter gibt dabei die Nummer *ndx* der zu beschreibenden Double-Variablen an. Der Wertebereich von *ndx* ist dabei 0 bis 999. Der zweite Parameter ist ein Zeiger auf die Struktur CDBUF mit 1000 double-Variablen. Vor Ausführung des Befehls muss die zu schreibende Variable mit dem entsprechend gewünschten Wert initialisiert werden.
BORLAND DELPHI:	procedure wrcd(ndx: integer; var cdbuf:CDBUF);
C:	void wrcd(int ndx, struct CDBUF far *cdbuf);
VISUAL BASIC:	Sub wrcd(ByVal ndx As Long, CDBUF As CDBUF)
ANMERKUNG:	Der Inhalt aller Common Variablen bleibt auch nach einem Systemrücksetzvorgang, welcher z.B. durch das *rs()*-Kommando ausgeführt wird, gespeichert. Wenn dies nicht erwünscht ist, sollten die betreffenden Variablen beim Programmstart auf den gewünschten Wert gesetzt werden.

4.4.146 wrci, write common integer

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *wrcd()* bis auf den Unterschied dass es sich hier um die *rw_SymPas*-System-Variablen CI0 .. CI999 vom Typ LONGINT handelt.
BORLAND DELPHI:	procedure wrci(ndx: integer; var cibuf:CIBUF);
C:	void wrci(int ndx, struct CIBUF far *cibuf);
VISUAL BASIC:	Sub wrci(ByVal ndx As Long, CIBUF As CIBUF)
ANMERKUNG:	PCAP-Befehl *wrcd()*

4.4.147 wrControllerFlags– Write Controller Flags

BESCHREIBUNG:	Mit diesem Befehl wird das achsspezifische, bitcodierte Register ControllerFlags der RWMOS-Betriebssystemsoftware beschrieben.
BORLAND DELPHI:	procedure wrControllerFlags (an: integer; var value: integer);
C:	void wrControllerFlags (long an, long *value);
VISUAL BASIC:	Sub wrControllerFlags (ByVal an As Long, value As Long)
PARAMETER:	Mit *an* wird der anzusprechende Achskanal angegeben (0, 1, ...). In *value* wird der zu schreibende bitcodierte Wert des Registers ControllerFlags übergeben.
ANMERKUNG:	Mit Hilfe von Flags (bits) im achsspezifischen ControllerFlags-Register können unterschiedliche Optionen im Regelalgorithmus von RWMOS.ELF aktiviert bzw. gesteuert werden. (siehe hierzu auch Kapitel 4.4.59 und 6.3.1.4).

4.4.148 wrdigo, write digital outputs

Funktionsbeschreibung gilt nur für MCU-3000 / MCU-3100

BESCHREIBUNG:	Mit diesem Register können die Digital-Ausgänge der MCU-3000 / MCU-3100 gesetzt werden. Zu beachten ist, dass die Digitalausgänge auf der MCU-G3 nicht achsspezifisch gruppiert sind. Sofern ein Ausgang gesetzt werden soll, wird dies durch Setzen des jeweiligen Bits erreicht. Der bitkodierte Aufbau des *digo*-Statuswortes kann folgender Tabelle entnommen werden:

Bitkodierter Aufbau des digo-Wortes

Bit-Nr.	Funktion	Stecker X1 / PIN
0	Ausgang 1	26
1	Ausgang 2	27
2	Ausgang 3	28
3	Ausgang 4	29
4	Ausgang 5	30
5	Ausgang 6	31
6	Ausgang 7	32
7	Ausgang 8	33
8..31	Nicht belegt.	--

BORLAND DELPHI:	procedure wrdigo(var tsrp:TSRP);
C:	void wrdigo(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrdigo(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].digo

Funktionsbeschreibung gilt nur für MCU-3400C / CPCI-8004

BESCHREIBUNG:	Mit diesem Register können die Digital-Ausgänge der MCU-3400C / CPCI-8004 gesetzt werden. Zu beachten ist, dass die Digitalausgänge auf der MCU-G3 nicht achsspezifisch gruppiert sind. Sofern ein Ausgang gesetzt werden soll, wird dies durch Setzen des jeweiligen Bits erreicht. Der bitkodierte Aufbau des *digo*-Statuswortes kann folgender Tabelle entnommen werden:

Bitkodierter Aufbau des digo-Wortes

Bit-Nr.	Funktion	Stecker X1 / PIN
0	Ausgang 1	17
1	Ausgang 2	18
2	Ausgang 3	19
3	Ausgang 4	37
4	Ausgang 5	38
5	Ausgang 6	39
6..31	Nicht belegt.	--

BORLAND DELPHI:	procedure wrdigo(var tsrp:TSRP);
C:	void wrdigo(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrdigo(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].digo

4.4.149 wrdigob, write digital output bit

Funktionsbeschreibung gilt nur für MCU-3000 / MCU-3100

BESCHREIBUNG: Mit dieser Funktion kann ein MCU-3000 / APCI-8001 Digital-Ausgang gesetzt bzw. rückgesetzt werden. Die Achsnummer muss im Parameter *an* (0, 1, ... *MAXAXIS-1*) spezifiziert werden. Das Rücksetzen des Ausgangs erfolgt mit dem Wert 0 bzw. FALSE.

Zuordnung von *bitnr* zu den jeweiligen MCU-G3 Digitalausgängen

'bitnr'	Funktion	Stecker X1 / PIN
1	Ausgang 1	26
2	Ausgang 2	27
3	Ausgang 3	28
4	Ausgang 4	29
5	Ausgang 5	30
6	Ausgang 6	31
7	Ausgang 7	32
8	Ausgang 8	33
9..32	Nicht belegt.	--

BORLAND DELPHI: procedure wrdigob(an:integer; bitnr:integer; value: integer);

C: wrdigob(int an, int bitnr, int value);

VISUAL BASIC: Sub wrdigob(ByVal an As Long, ByVal bitnr As Long, ByVal value As Long)

ANMERKUNG PCAP-Befehl *wrdigo()*

Funktionsbeschreibung gilt nur für MCU-3400C / CPCI-8004

BESCHREIBUNG: Mit dieser Funktion kann ein MCU-3400C / CPCI-8004 Digital-Ausgang gesetzt bzw. rückgesetzt werden. Die Achsnummer muss im Parameter *an* (0, 1, ... *MAXAXIS-1*) spezifiziert werden. Das Rücksetzen des Ausgangs erfolgt mit dem Wert 0 bzw. FALSE.

Zuordnung von *bitnr* zu den jeweiligen MCU-G3 Digitalausgängen

'bitnr'	Funktion	Stecker X1 / PIN
1	Ausgang 1	17
2	Ausgang 2	18
3	Ausgang 3	19
4	Ausgang 4	37
5	Ausgang 5	38
6	Ausgang 6	39
7..32	Nicht belegt.	--

BORLAND DELPHI: procedure wrdigob(an:integer; bitnr:integer; value: integer);

C: wrdigob(int an, int bitnr, int value);

VISUAL BASIC: Sub wrdigob(ByVal an As Long, ByVal bitnr As Long, ByVal value As Long)

ANMERKUNG PCAP-Befehl *wrdigo()*

4.4.150 wrdp, write desired position

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische Sollposition (*dp*) geschrieben werden. Dieser Befehl wird normalerweise nie benötigt und sollte nur in ganz besonderen Fällen wie z.B. beim Test oder bei der Inbetriebnahme verwendet werden. Das Verändern der Sollposition wirkt sich lediglich in der Betriebsart Lageregelung aus. Bei großen Differenzen zwischen dieser Sollposition (*dp*) und der aktuellen Position (*rp*) muss damit gerechnet werden, dass der Motor mit der maximalen Systembeschleunigung auf diese Position nachgeführt wird.
BORLAND DELPHI:	procedure wrdp(var tsrp:TSRP);
C:	void wrdp(struct TSRP far *tsrp) ;
VISUAL BASIC:	Sub wrdp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].dp
ANMERKUNG:	Das Schreiben der Sollposition (*dp*) bei der Ausführung von Bewegungskommandos kann unter Umständen zu einem unkontrollierten Prozessverhalten führen und sollte deshalb vermieden werden. PCAP-Befehl *rddp()*

4.4.151 wrdpoffset, write desired position offset

BESCHREIBUNG:	Mit diesem Befehl kann der achsspezifische Sollpositions-Offset (*dpoffset*) beschrieben werden.
BORLAND DELPHI:	function wrdpoffset (an: integer; var value: double): integer;
C:	int wrdpoffset(int an, double *value);
VISUAL BASIC:	Function wrdpoffset (ByVal an As Long, value As Double) As Long
PARAMETER:	Mit *an* wird der anzusprechende Achskanal angegeben (0, 1, ...). In *value* wird der zu schreibende Positionsoffset in der achsspezifischen Positionseinheit übergeben.
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Im Allgemeinen dürfen hier nur geringfügige Änderungen programmiert werden, da die entsprechenden Sollwertänderungen jeweils einen Sprung der Achse bewirken. Siehe hierzu auch Achsen-Qualifizierer dpoffset in Tabelle 39. Mit einem Wert dvoffset (siehe Kapitel 4.4.152) kann die Änderungsgeschwindigkeit von dpoffset jedoch parametriert werden. Dieses Register kann z.B. für eine dem Positionsregler überlagerte Regelung oder für eine Spindellinearisierung / Spindelkorrektur verwendet werden. **Vorsicht:** Dieser Mechanismus wird intern verwendet von der sogenannten RTCP (Rotation Tool Center Point) Korrektur. Falls diese verwendet wird, darf der Sollpositions-Offset dpOffset bei den entsprechenden Achsen nicht beschrieben werden.

4.4.152 wrdvoffset, write desired velocity offset

BESCHREIBUNG:	Mit diesem Befehl kann die Änderungsgeschwindigkeit (dvoffset) des achsspezifischen Sollpositions-Offset (*dpoffset*) beschrieben werden.
BORLAND DELPHI:	function wrdvoffset (an: integer; var value: double): integer;
C:	int wrdvoffset(int an, double *value);
VISUAL BASIC:	Function wrdvoffset (ByVal an As Long, value As Double) As Long
PARAMETER:	Mit *an* wird der anzusprechende Achskanal angegeben (0, 1, ...). In *value* wird die zu schreibende Änderungsgeschwindigkeit in der achsspezifischen Positionseinheit übergeben.
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Mit dem Wert 0 werden Änderungen von dpoffset sofort übernommen. Defaultwert ist 0.

4.4.153 wrEffRadius – Write Effective Radius

BESCHREIBUNG:	Mit diesem Befehl kann der effektive Radius für eine rotatorische Achse geschrieben werden.
BORLAND DELPHI:	wrEffRadius (an: integer; var value: double);
C:	void wrEffRadius (long an, double *value);
VISUAL BASIC:	Sub wrEffRadius (an As Long, ByVal value As Double)
PARAMETER:	In an wird die Achsnummer angegeben, in value wird der wirksame Radius in der Einheit übergeben, die per PU verwendet wird.
ANMERKUNG:	siehe Kapitel 6.3.3

4.4.154 wrErrorReg, write Error Register

BESCHREIBUNG:	Mit dieser Funktion kann das ErrorRegister der RWMOS-Betriebssystemsoftware beschrieben werden.
BORLAND DELPHI:	procedure wrErrorReg(var ErrorReg: integer);
C:	void wrErrorReg (long *ErrorReg);
VISUAL BASIC:	Sub wrErrorReg (ErrorReg As Integer)
RÜCKGABEWERT:	Die Funktion hat keinen Rückgabewert.
ANMERKUNG:	Es macht nur Sinn, hier den Wert 0 zu schreiben um das ErrorRegister komplett zu löschen.

4.4.155 wrGCR, write gear configuration register

BESCHREIBUNG:	Mit dieser Funktion kann das achsspezifische Gear Configuration Register beschrieben werden. [Kapitel 6.3.3]
BORLAND DELPHI:	procedure wrGCR (an: integer; var value: integer);
C:	void wrGCR (long an, long *value);
VISUAL BASIC:	Sub wrGCR (ByVal an As Long, value As Long)
PARAMETER:	Mit *an* wird der auszulesende Achskanal angegeben (0, 1, ...). In *value* wird der Inhalt des GCR Registers übergegeben.
RÜCKGABEWERT:	keiner
ANMERKUNG:	Siehe auch Doku zum Resourcen-Interface - GEAR

4.4.156 wrgf, write gear factor

BESCHREIBUNG:	Mit diesem Befehl kann der achsspezifische Getriebe-Faktor in der entsprechenden Einheit neu gesetzt werden. Dies ist z.B. notwendig bei Schalt-Getrieben oder durch laufzeitbedingte Änderungen von Systemgrößen wie z.B. Werkstück- oder Werkzeug-Abmessungen oder anderen Korrekturfaktoren.
BORLAND DELPHI:	procedure wrgf(var tsrp:TSRP);
C:	void wrgf(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrgf(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].gf
ANMERKUNG:	Zu beachten ist, dass gerade bei großen Änderungen des Getriebe-Faktors die aktuellen achsspezifischen Beschleunigungs- und Geschwindigkeitsparameter an diesen neuen Faktor angepaßt werden müssen, da dieser zur Umrechnung dieser Systemparameter herangezogen wird. Der aktuell gesetzte Wert von *gf* kann mit dem PCAP-Befehl *rdgf()* gelesen werden.

4.4.157 wrgfaux, write gear factor auxiliary channel

BESCHREIBUNG:	Mit dieser Funktion kann das achsspezifische Verhältnis zwischen Schrittmotor-Auflösung und Enkoder-Kanal bei Stepper-Systemen mit Enkoderverification geschrieben werden. Defaultwert ist 1.0, der Wert kann nur zur Laufzeit verändert werden.
BORLAND DELPHI:	function wrgfaux (an: integer; var value: double) : integer;
C:	int wrgfaux(int an, double *value)
VISUAL BASIC:	Function wrgfaux (ByVal an As Long, value As Double) As Long
RÜCKGABEWERT:	Die Funktion liefert nach erfolgreicher Ausführung 0 zurück. In diesem Fall konnte der Wert in value erfolgreich auf die Achse an geschrieben werden. Bei einem Rückgabewert ≠ 0 konnte der Wert nicht geschrieben werden weil z.B. RWMOS.ELF das Kommando nicht unterstützt. 0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Der Faktor kann mit dem PCAP-Befehl *rdgfaux()* jederzeit gelesen werden. Siehe auch Kapitel 6.3.3

4.4.158 wrhac, write home acceleration

BESCHREIBUNG:	Mit diesem Befehl wird die achsspezifische Maximalbeschleunigung *hac* für alle Referenzfahrtbefehle (*home*-Befehle) gesetzt. Sofern dieser Befehl nicht zur Ausführung kommt, wird mit dem im TOOLSET-Programm *mcfg.exe* festgelegten Systemparameter gearbeitet. Der Systemparameter kann zu jedem beliebigen Zeitpunkt überschrieben werden.
BORLAND DELPHI:	procedure wrhac(var tsrp:TSRP);
C:	void wrhac(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrhac(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].hac
ANMERKUNG:	Der aktuell gesetzte Wert von *hac* kann mit dem PCAP-Befehl *rdhac()* gelesen werden.

4.4.159 wrhvl, write home velocity

BESCHREIBUNG:	Mit diesem Befehl wird die achsspezifische Maximalgeschwindigkeit mit Hilfe der Variablen *hvl* für alle Referenzfahrtbefehle (*home*-Befehle) gesetzt. Sofern dieser Befehl nicht zur Ausführung kommt, wird mit dem im TOOLSET-Programm *mcfg.exe* festgelegten Systemparameter gearbeitet. Der Systemparameter kann zu jedem beliebigen Zeitpunkt überschrieben werden.
BORLAND DELPHI:	procedure wrhvl(var tsrp:TSRP);
C:	void wrhvl(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrhvl(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].hvl
ANMERKUNG:	Der aktuell gesetzte Wert von *hac* kann mit dem PCAP-Befehl *rdhvl()* gelesen werden.

4.4.160 wripw, write in position window

BESCHREIBUNG:	Mit diesem Befehl kann das mit Hilfe des TSW-Programms *mcfg.exe* festgelegte In-Positions-Fenster {ipw} während der Laufzeit verändert werden. Das Fenster wird auf den in *ipw* gesetzten Wert neu festgelegt. Die Wertangabe erfolgt in der achsspezifischen Positionseinheit.
BORLAND DELPHI:	procedure wripw(var tsrp:TSRP);
C:	void wripw(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wripw(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].ipw
ANMERKUNG:	Das In-Positions-Fenster wird nur dann überwacht, sofern ein Wert größer 0.0 spezifiziert wurde. [mcfg / Kapitel 1.7.2.1.11] PCAP-Befehl rdipw*()*

4.4.161 wrjac, write jog acceleration

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *wrhac().* Jedoch wird hier die maximale Systembeschleunigung mit Hilfe der Variablen *jac* für alle *jog*-Befehle festgelegt.
BORLAND DELPHI:	procedure wrjac(var tsrp:TSRP);
C:	void wrjac(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrjac(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].jac
ANMERKUNG:	Der aktuell gesetzte Wert von *jac* kann mit dem PCAP-Befehl *rdjac()* gelesen werden.

4.4.162 wrJerkRel, write jerkrel

BESCHREIBUNG:	Mit diesem Befehl wird der achsspezifische Parameter *jerkrel* beschrieben.
BORLAND DELPHI:	procedure wrJerkRel (an: integer; var value: double);
C:	void wrJerkRel (long an, double *value);
VISUAL BASIC:	Sub wrJerkRel (an As Long, ByVal value As Double)
PARAMETER:	an = Achsnummer (0..n) value = zu schreibender Wert
RÜCKGABEWERT:	keiner
ANMERKUNG:	*jerkrel* kann nur einen Wert von 0..1 zugewiesen Werten. Grössere oder kleinere Werte werden begrenzt. Siehe dazu auch Kapitel 4.4.84 und 6.3.3.

4.4.163 wrjovr, write jog override

BESCHREIBUNG:	Dieser Befehl setzt den achsspezifischen Geschwindigkeitskorrekturwert. Dieser Korrekturwert wird bei allen *Jog*-Befehlen berücksichtigt. Der Parameter *jovr* muss einen Wert größer 0.0 haben. Alle Werte kleiner 1.0 resultieren in einer Reduzierung der Achsgeschwindigkeit. Sofern *value* einen Wert größer 1.0 hat, äußert sich dies mit einer Erhöhung der Geschwindigkeit.
BORLAND DELPHI:	procedure wrjovr(var trsp:TSRP);
C:	void wrjovr(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrjovr(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].jovr
ANMERKUNG:	Zu beachten ist, dass der spezifizierte Korrekturwert gleichermaßen auf die aktuelle Achsbeschleunigung wirkt. Ein zu rasches Anheben- bzw. Absenken des Korrekturwertes kann sich in einem Beschleunigungssprung (Ruck) der Achse äußern. Der Korrekturfaktor sollte deshalb über Verzögerungsschleifen linear bis zum gewünschten Endwert inkrementiert- bzw. dekrementiert werden. Bei der Ausführung der PCAP-Befehle *ra()*, *rs()* oder SAP-Befehle *RA(), RS*, wird der Override-Faktor auf den Defaultwert 1.0 initialisiert. Bei Aufruf von utrovr und selektierter Achse, wird der Wert von jovr ebenfalls gesetzt, wenn dies nicht explizit in der Registervariable ModeReg abgeschaltet ist. PCAP-Befehl *rdjovr()*

4.4.164 wrjtvl, write jog target velocity

BESCHREIBUNG:	Mit diesem Befehl wird die achsspezifische jog-Zielgeschwindigkeit mit Hilfe der Variablen *jtvl* für die *jog*-Befehle *ja()* und *jr()* gesetzt. Sofern dieser Befehl nicht zur Ausführung kommt, wird mit dem im TOOLSET-Programm *mcfg.exe* festgelegten Systemparameter gearbeitet. Der Systemparameter kann zu jedem beliebigen Zeitpunkt überschrieben werden.
BORLAND DELPHI:	procedure wrjtvl(var tsrp:TSRP);
C:	void wrjtvl(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrjtvl(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].jtvl
ANMERKUNG:	Der aktuell gesetzte Wert von *jtvl* kann mit dem PCAP-Befehl *rdjtvl()* gelesen werden.

4.4.165 wrjvl, write jog velocity

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *wrhvl().* Jedoch wird hier die maximale Verfahrgeschwindigkeit mit Hilfe der Variablen *jvl* für alle *jog*-Befehle festgelegt.
BORLAND DELPHI:	procedure wrjvl(var tsrp:TSRP);
C:	void wrjvl(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrjvl(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].jvl
ANMERKUNG:	Der aktuell gesetzte Wert von *jvl* kann mit dem PCAP-Befehl *rdjvl()* gelesen werden.

4.4.166 wrledgn, write led green

BESCHREIBUNG:	
MCU-3000 / APCI-8001:	Mit diesem Befehl kann die grüne SMD-Leuchtdiode D29 ein- bzw. ausgeschaltet werden. Das Einschalten erfolgt mit dem Wert 1, das Ausschalten mit dem Wert 0.
MCU-3100 / APCI-8008:	Mit diesem Befehl kann die grüne SMD-Leuchtdiode D53 ein- bzw. ausgeschaltet werden. Das Einschalten erfolgt mit dem Wert 1, das Ausschalten mit dem Wert 0.
MCU-6000 / APCI-8401:	Mit diesem Befehl kann die grüne SMD-Leuchtdiode D10 ein- bzw. ausgeschaltet werden. Das Einschalten erfolgt mit dem Wert 1, das Ausschalten mit dem Wert 0.
MCU-3400C/CPCI-8004:	Mit diesem Befehl kann die grüne SMD-Leuchtdiode D36 ein- bzw. ausgeschaltet werden. Das Einschalten erfolgt mit dem Wert 1, das Ausschalten mit dem Wert 0.
BORLAND DELPHI:	procedure wrledgn(value:integer);
C:	void wrledgn(int value);
VISUAL BASIC:	Sub wrledgn(ByVal value As Long)
ANMERKUNG:	Dieser Befehl dient vor allem als Test- und Diagnosehilfsmittel. Die SMD-Leuchtdiode befindet sich am hinteren oberen Ende der Lötseite.

4.4.167 wrledrd, write led red

BESCHREIBUNG: **MCU-3000 / APCI-8001:**	wie PCAP-Befehl *wrledgn()*, jedoch rote LED D31
MCU-3100 / APCI-8008:	wie PCAP-Befehl *wrledgn()*, jedoch rote LED D56
MCU-6000 / APCI-8401:	wie PCAP-Befehl *wrledgn()*, jedoch rote LED D12
MCU-3400C/CPCI-8004:	wie PCAP-Befehl *wrledgn()*, jedoch rote LED D38
BORLAND DELPHI:	procedure wrledrd(value:integer);
C:	void wrledrd(int value);
VISUAL BASIC:	Sub wrledrd(ByVal value As Long)

4.4.168 wrledyl, write led yellow

BESCHREIBUNG: **MCU-3000 / APCI-8001:**	wie PCAP-Befehl *wrledgn()*, jedoch gelbe LED D30
MCU-3100 / APCI-8008:	wie PCAP-Befehl *wrledgn()*, jedoch gelbe LED D55
MCU-6000 / APCI-8401:	wie PCAP-Befehl *wrlegnl()*, jedoch gelbe LED D11
MCU-3400C/CPCI-8004:	wie PCAP-Befehl *wrlegnl()*, jedoch gelbe LED D37
BORLAND DELPHI:	procedure wrledyl(value:integer);
C:	void wrledyl(int value);
VISUAL BASIC:	Sub wrledyl(ByVal value As Long)

4.4.169 wrlp, write latched position

BESCHREIBUNG:	Dieser Befehl setzt die achsspezifische Latchposition auf den in *lp* gesetzten Wert. Die Wertangabe erfolgt in der achsspezifischen Positionseinheit.
BORLAND DELPHI:	procedure wrlp(var tsrp:TSRP);
C:	void wrlp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrlp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].lp
ANMERKUNG:	PCAP-Befehl *rdlp()*

4.4.170 wrlpndx, write latched position index

Funktionsbeschreibung gilt nur für MCU-3000 / MCU-3100 / MCU-3400C

BESCHREIBUNG:	Dieser Befehl setzt die achsspezifische Latchposition der Nullspur (Index) auf den in *lp* gesetzten Wert. Die Wertangabe erfolgt in der achsspezifischen Positionseinheit.
BORLAND DELPHI:	procedure wrlpndx(var tsrp:TSRP);
C:	void wrlpndx(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrlpndx(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].lp
ANMERKUNG:	PCAP-Befehl *rdlpndx()*

4.4.171 wrMaxAcc – Write Maximum Acceleration Check

BESCHREIBUNG:	Mit diesem Befehl kann die maximale achsspezifische Beschleunigung (*MAXACC*) geschrieben werden. Dieser Wert wird von der RWMOS-Betriebssystemsoftware verwendet, um die Bahnbeschleunigung derart zu begrenzen, dass keine der an einer Linearinterpolation beteiligten Achsen, ihre maximal erlaubte Beschleunigung überschreitet.
BORLAND DELPHI:	wrMaxAcc (an: integer; var value: double);
C:	void wrMaxAcc (long an, double *value);
VISUAL BASIC:	Sub wrMaxAcc (an As Long, ByVal value As Double)
PARAMETER:	In *an* wird die Achsnummer angegeben, in *value* wird die maximal erlaubte Beschleunigung in der interpolationsspezifischen Beschleunigungseinheit übergeben (PU und TU).
ANMERKUNG:	Um diese Überwachung zu aktivieren, muss Bit 7 im MODEREG-Register gesetzt werden (siehe Kapitel 6.3.1.5). Mit dem Wert 0, wird die Überwachung bei der jeweiligen Achse unterdrückt. Die Funktion ist nur bei gespoolten Kommandos verfügbar.

4.4.172 wrMaxVel – Write Maximum Velocity Check

BESCHREIBUNG:	Mit diesem Befehl kann die maximale achsspezifische Geschwindigkeit (*MAXVEL*) geschrieben werden. Dieser Wert wird von der RWMOS-Betriebssystemsoftware verwendet, um die Bahngeschwindigkeit derart zu begrenzen, dass keine der an einer Linearinterpolation beteiligten Achsen, ihre maximal erlaubte Geschwindigkeit überschreitet.
BORLAND DELPHI:	wrMaxVel (an: integer; var value: double);
C:	void wrMaxVel (long an, double *value);
VISUAL BASIC:	Sub wrMaxVel (an As Long, ByVal value As Double)
PARAMETER:	In *an* wird die Achsnummer angegeben, in *value* wird die maximal erlaubte Geschwindigkeit übergeben. Dieser Wert wird stets in der Interpolationseinheit interpretiert.
ANMERKUNG:	Um diese Überwachung zu aktivieren, muss Bit 7 im MODEREG-Register gesetzt werden (siehe Kapitel 6.3.1.5). Mit dem Wert 0, wird die Überwachung bei der jeweiligen Achse unterdrückt. Die Funktion ist nur bei gespoolten Kommandos verfügbar.

4.4.173 wrmcp, write motor command port

Beschreibung:

Dieser Befehl dient zum Beschreiben des Motor-Command-Ports auf den im Feld *mcp* gesetzten Wert. Dies ist vor allem bei der Inbetriebnahme hilfreich, wenn z.B. der Sollwertkanal des Antriebssystems überprüft werden soll. Im Idle-Mode (keine Lageregelung) kann die Motorachse mit diesem Befehl ungeregelt verfahren werden. Auf diese Art kann z.B. die Drehrichtung des Antriebes, die korrekte Arbeitsweise der Impulserfassung und Endschalter u.a. überprüft werden, bevor die Inbetriebnahme in der Betriebsart Lageregelung fortgesetzt wird.
Das Beschreiben des Motor-Command-Ports macht i.A. nur Sinn bei geöffnetem Regelkreis, da der Lageregler diesen Port ansonsten abtastsynchron überschreibt.

MCU-3000 / APCI-8001:
MCU-3100 / APCI-8008:
MCU-3400C/CPCI-8004:

Bei Servo-Achsen kann *mcp* auf einen Wert zwischen -32767 und +32767 gesetzt werden. Dieser Wertebereich entspricht dem Analogausgangsspannungsbereich von -10V bis +10V. Eventuell muss eine projektierte Invertierung des Analogausgangssignals berücksichtigt werden.

Bei Schrittmotorachsen kann mit *mcp* eine Zeitverzögerung spezifiziert werden, mit deren Hilfe ein Schrittsignal für Schrittmotor-Leistungsendstufen generiert wird. Die Frequenz dieses Schrittsignals kann wie folgt berechnet werden:

f_{Pulse} = CLOCK/2/(mcp+1)

Beispiel: mit mcp = 999 und CLOCK = 70MHz
wird f_{Pulse}=35000[Hz]

Der Wert für CLOCK ist 70 MHz für die MCU-3000 / APCI-8001 und 66,66666 MHz für die MCU-3100 / APCI-8008.
Der Wertebereich von *mcp* liegt zwischen -1048574 und +1048574. Das Vorzeichen selektiert die gewünschte Verfahrrichtung und beinflußt das achsspezifische Richtungssignal. Für das Schrittsignal f_{Pulse} ist nur der Betrag von *mcp* maßgebend. Zu beachten ist, dass der Wert 0 von *mcp* ein Schrittsignal von 0Hz bewirkt, d.h. der Motor bleibt stehen.

MCU-6000 / APCI-8401:

Bei Servo-Achsen kann *mcp* auf einen Wert zwischen -2047 und +2047 gesetzt werden. Dieser Wertebereich entspricht dem Analogausgangsspannungsbereich von -10V bis +10V. Eventuell muss eine projektierte Invertierung des Analogausgangssignals berücksichtigt werden.

Bei Schrittmotorachsen kann mit *mcp* eine Zeitverzögerung spezifiziert werden, mit deren Hilfe ein Schrittsignal für Schrittmotor-Leistungsendstufen generiert wird. Die Frequenz dieses Schrittsignals kann wie folgt berechnet werden:

f_{Pulse} = CLOCK/32/(mcp+1)

Beispiel: mit mcp = 99 und CLOCK = 30MHz
wird f_{Pulse}=9375[Hz]

Der Wertebereich von *mcp* liegt zwischen -32767 und +32767. Das

	Vorzeichen selektiert die gewünschte Verfahrrichtung und beinflußt das achsspezifische Richtungssignal. Für das Schrittsignal f_{Pulse} ist nur der Betrag von *mcp* maßgebend. Zu beachten ist, dass der Wert 0 von *mcp* ein Schrittsignal von 0Hz bewirkt, d.h. der Motor bleibt stehen.
BORLAND DELPHI:	procedure wrmcp(var tsrp:TSRP);
C:	void wrmcp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrmcp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].mcp
ANMERKUNG:	Sofern sich das Achssystem in Lageregelung befindet, wirkt sich dieser Befehl höchstens für den Zeitraum eines Abtastintervalles aus, da die Motor-Command-Ports nach der Abarbeitung des PIDF-Filters neu gesetzt werden. PCAP-Befehl *rdmcp()*

4.4.174 wrMDVel – Write Maximum Velocity Skip

BESCHREIBUNG:	Mit diesem Befehl kann der maximale achsspezifische Geschwindigkeitssprung (*MDVEL*) geschrieben werden. Dieser Wert wird von der Look-Ahead Funktionalität der RWMOS-Betriebssystemsoftware verwendet, um die Bahngeschwindigkeit derart zu begrenzen, dass keine der an einer Interpolation beteiligten Achsen, ihren maximal erlaubten Geschwindigkeitssprung überschreitet.
BORLAND DELPHI:	wrMDVel (an: integer; var value: double);
C:	void wrMDVel (long an, double *value);
VISUAL BASIC:	Sub wrMDVel (an As Long, ByVal value As Double)
PARAMETER:	In *an* wird die Achsnummer angegeben, in *value* wird der maximal erlaubte Geschwindigkeitssprung in den jeweils aktivierten Positions- und Zeiteinheiten (PU und TU) übergeben.
ANMERKUNG:	Der Look-Ahead-Modus wird durch Setzen von Bit 0 im MODEREG-Register (siehe Kapitel 6.3.1.5) aktiviert. Mit dem Wert 0 wird die Überwachung der jeweiligen Achse abgeschaltet. In diesem Zusammenhang ist auch Bit 6 von MODEREG zu beachten. **Hinweis:** Im Look-Ahead-Modus müssen die einzelnene Profile mit einer Zielgeschwindigkeit > 0 programmiert werden (i.A. = Maximalgeschwindigkeit), damit der Look-Ahead überhaupt wirksam werden kann.

4.4.175 wrModeReg – Write MODEREG

BESCHREIBUNG:	Mit diesem Befehl wird das Register MODEREG der RWMOS-Betriebssystemsoftware beschrieben.
BORLAND DELPHI:	procedure wrModeReg (var value: integer);
C:	void wrModeReg(long *value);
VISUAL BASIC:	Sub wrModeReg (ByVal value As Long)
PARAMETER:	bitcodierter Wert für ModeReg
ANMERKUNG:	Mit Hilfe von Flags (bits) im ModeReg-Register können unterschiedliche Optionen in RWMOS.ELF aktiviert bzw. gesteuert werden wie z.B. Look-Ahead, S-Profile usw. (siehe Kapitel 6.3.1.5).

4.4.176 wrmpe, write maximum position error

BESCHREIBUNG:	Mit diesem Befehl kann die mit Hilfe des TSW-Programms *mcfg.exe* festgelegte Schleppfehlergrenze {mpe} während der Laufzeit verändert werden. Der achsspezifische maximal erlaubte Schleppfehler wird auf den in *mpe* gesetzten Wert neu festgelegt. Die Wertangabe erfolgt in der achsspezifischen Positionseinheit.
BORLAND DELPHI:	procedure wrmpe(var tsrp:TSRP);
C:	void wrmpe(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrmpe(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].mpe
ANMERKUNG:	Die Schleppfehlerüberwachung findet nur dann statt, wenn ein Wert größer 0.0 spezifiziert wurde und der Regelkreis geschlossen ist. [mcfg / Kapitel 1.7.2.1.9] PCAP-Befehl *rdmpe()*

4.4.177 wrnfrax, write No-Feed-Rate-Axis

BESCHREIBUNG:	Mit diesem Befehl wird das Register NFRAX der RWMOS-Betriebssystemsoftware beschrieben.
BORLAND DELPHI:	wrnfrax (var value: integer);
C:	void wrnfrax (long *value);
VISUAL BASIC:	Sub wrnfrax (ByVal value As Long)
PARAMETER:	bitcodierter Wert für NFRAX
ANMERKUNG:	Im Register NFRAX können bitcodiert sogenannte No-Feed-Rate Achsen definiert werden. Diese Achsen werden bei Interpolationsbefehlen nicht für die Bahngeschwindigkeitsberechnung herangezogen, nehmen aber trotzdem an der Interpolation teil. Dadurch kann ein Einfluß von Hilfsachsen auf die Bahngeschwindigkeit in Interpolationsprofilen verhindert werden. Siehe hierzu auch Funktion rdnfrax Kapitel 4.4.100.

4.4.178 wrrp, write real position

BESCHREIBUNG:	Dieser Befehl setzt das achsspezifische aktuelle Positionsregister auf den in *rp* gesetzten Wert und ist nur im Open-Loop-Mode (keine Lageregelung) wirksam. Die Wertangabe erfolgt in der achsspezifischen Positionseinheit.
BORLAND DELPHI:	procedure wrrp(var tsrp:TSRP);
C:	void wrrp(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrrp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].rp
ANMERKUNG:	Mit diesem Befehl verschiebt sich automatisch der Maschinen-Nullpunkt!

4.4.179 wrsdec, write stop deceleration

BESCHREIBUNG:	Mit diesem Befehl wird die achsspezifische Stopverzögerung *sdec* für den PCAP-Befehl *js()* [Kapitel 4.4.24], SAP-Befehl *JS()* [Kapitel 6.6.26], die mit SMD-projektierten Software-Endlagen [mcfg / Kapitel 1.7.2.1.10] [mcfg / Kapitel 1.7.2.2.3]und die mit LSL_SMD bzw. LSR_SMD projektierten Digital-Eingänge [mcfg / Kapitel 1.7.2.5] gesetzt. Sofern *wrsdec()* nicht zur Ausführung kommt, wird mit dem im TOOLSET-Programm *mcfg.exe* festgelegten Systemparameter gearbeitet. Der Systemparameter kann zu jedem beliebigen Zeitpunkt überschrieben werden.
BORLAND DELPHI:	procedure wrsdec(var tsrp:TSRP);
C:	void wrsdec(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrsdec(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].sdec
ANMERKUNG:	Der aktuell gesetzte Wert von *sdec* kann mit dem PCAP-Befehl *rdsdec()* gelesen werden [Kapitel 4.4.106].

4.4.180 wrsll, write software limit left

BESCHREIBUNG:	Mit diesem Befehl kann die mit Hilfe des TSW-Programms *mcfg.exe* festgelegte achsspezifische linke Software-Endlagen-Position {sll} während der Laufzeit verändert werden. Die linke Software-Endlage wird auf den in *sll* gesetzten Wert neu festgelegt. Die Wertangabe erfolgt in der achsspezifischen Positionseinheit.
BORLAND DELPHI:	procedure wrsll(var tsrp:TSRP);
C:	void wrsll(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrsll(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].sll
ANMERKUNG:	Die gesetzte Software-Endlage wird nur dann berücksichtigt, wenn die Home-Position des entsprechenden Achskanals bereits definiert wurde oder nach Ausführung dieses Befehls gesetzt wird. [mcfg / Kapitel 1.7.2.1.10] PCAP-Befehle *rdsll()*, *shp()*, SAP-Befehl *SHP()*

4.4.181 wrslr, write software limit right

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *wrsll()*, jedoch wird die rechte Software-Endlage mit dem im Parameter *slr* gesetzten Wert neu definiert.
BORLAND DELPHI:	procedure wrslr(var tsrp:TSRP);
C:	void wrslr(struct TSRP far *tsrp);
VISUAL BASIC:	Sub wrslr(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].slr

4.4.182 wrslsp, write Slits / Stepperpulses

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische Auflösung pro Motorumdrehung {slsp} gesetzt werden. Der Defaultwert wird mit Hilfe des TOOLSET Programms *mcfg.exe* festgelegt.
BORLAND DELPHI:	procedure wrslsp (an: integer; var value: double);
C:	void wrslsp (long an, double *value);
VISUAL BASIC:	Sub wrslsp (ByVal an As Long, value As Double)
TSRP-KOMPONENTEN:	keine
ANMERKUNG:	slsp kann mit dem PCAP-Befehl *rdslsp()* gelesen werden. Siehe hierzu auch Achsenqualifizierer slsp. Für slsp sind nur Zahlenwerte > 0.0 erlaubt.

4.4.183 wrtp – write target position

BESCHREIBUNG:	Mit diesem Befehl kann die achsspezifische Zielposition (*tp*) beschrieben werden. Dieser Befehl wird normalerweise nie benötigt und sollte nur in ganz besonderen Fällen verwendet werden.
BORLAND DELPHI:	procedure wrtp(var tsrp:TSRP);
C:	void wrtp(struct TSRP far *tsrp) ;
VISUAL BASIC:	Sub wrtp(DTSRP As TSRP)
TSRP-KOMPONENTEN:	TSRP[n].tp
ANMERKUNG:	Das Schreiben der Zielposition (*tp*) bei der Ausführung von Bewegungskommandos kann unter Umständen zu einem unkontrollierten Prozessverhalten führen und sollte deshalb vermieden werden. Siehe auch PCAP-Befehl *rdtp().*

4.4.184 wrtrac, write trajectory acceleration

BESCHREIBUNG:	Schreiben der RWMOS Systemvariablen TRAC
BORLAND DELPHI:	function wrtrac (var value:double) : integer;
C:	int wrtrac (double *value);
VISUAL BASIC:	Function wrtrac (value As Double) as long
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	TRAC ist die Interpolations Bahnbeschleunigung, die bei Interpolationsbefehlen, welche aus der rw_SymPas Programmierumgebung aufgerufen werden, herangezogen wird (siehe auch *rw_SymPas* System-Parameter in Tabelle 34). Bei der PCAP-Programmierung wird dieser Parameter beim Aufruf in der Datenstruktur LMP, CMP oder HMP übergeben. Lediglich beim PCAP-Aufruf von Motion-Stop (ms) wird diese Beschleunigung verwendet, wenn das Bit 15 (MS_DECEL) im Register ModeReg (Tabelle 38) gesetzt ist.

4.4.185 wrtrovr, write trajectory override

BESCHREIBUNG:	Dieser Befehl setzt den Bahngeschwindigkeitskorrekturwert für alle Interpolationsbefehle (*move*-Befehle). Der Parameter *value* muss einen Wert größer 0.0 haben. Alle Werte kleiner 1.0 resultieren in einer Reduzierung der Bahngeschwindigkeit. Sofern *value* einen Wert größer 1.0 hat, äußert sich dies mit einer Erhöhung der Bahngeschwindigkeit. Der in *value* spezifizierte Korrekturwert wird auf der MCU-G3 in einer System-Variablen zwischengespeichert und ist erst nach Ausführung des PCAP-Befehls *utrovr()*, bzw. SAP-Befehls *UTROVR()* wirksam. Die dort angewählten Achskanäle werden auch während der Bahnfahrt je nach Korrekturfaktor *value* abgebremst bzw. beschleunigt.
BORLAND DELPHI:	procedure wrtrovr(var value:double);
C:	void wrtrovr(double *value);
VISUAL BASIC:	Sub wrtrovr(value As Double)
ANMERKUNG:	Zu beachten ist, dass der spezifizierte Korrekturwert gleichermaßen auf die aktuelle Bahnbeschleunigung wirkt. Ein zu rasches Anheben- bzw. Absenken des Korrekturwertes kann sich in einem Beschleunigungssprung (Ruck) der Achsen äußern. Der Korrekturfaktor sollte deshalb über Verzögerungsschleifen linear bis zum gewünschten Endwert inkrementiert- bzw. dekrementiert werden. Bei der Ausführung des PCAP-Befehl *rs()* oder SAP-Befehl *RS*, wird der Override-Faktor auf den Defaultwert 1.0 initialisiert. PCAP-Befehle *wrtrovr()*, *wrjovr()*, *rdtrovr()* und *rdjovr()*

4.4.186 wrtrovrst, write trajectory override settling time

BESCHREIBUNG:	Mit diesem Befehl läßt sich eine „weiche“ Anpassung des Override-Wertes TROVR nach dem Aufruf von utrovr() realisieren. Im Parameter value wird vor dem Aufruf von utrovr() eine Zeit in Sekunden angegeben, welche der Anpassdauer zwischen den Werten 0 und 1 entspricht. Dadurch können Geschwindigkeitssprünge, verursacht durch Programmierung des Override verhindert werden. Angegebene Zeiten, die kleiner sind als ein Abtastintervall werden nicht berücksichtigt. Mit dem Wert 0 wird die Funktion deaktiviert.
BORLAND DELPHI:	function wrtrovr(var value:double) : integer;
C:	int wrtrovr(double *value);
VISUAL BASIC:	Function wrtrovr(value As Double) as long
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt , Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	Siehe hierzu auch Kommandos rdtrovrst, wrtrovr, rdtrovr, utrovr und rw_SymPas Systemparameter TROVRST Ein Setzen des Jog-Override per wrjovr wird von dieser Funktionaliät nicht beeinflusst. Siehe hierzu auch Bit 25 im Register MODEREG (Kapitel 6.3.1.5). Ein Lesen des Parameters TROVR liefert immer den programmierten Sollwert zurück, auch während der Anpassungsphase. Falls der aktuell wirksame Overridewert während der Anpassungsphase gelesen werden soll, kann auch den aktuellen Jog-Override einer der beteiligten Achsen zurückgegriffen werden.

4.4.187 wrtrvl, write trajectory velocity

BESCHREIBUNG:	Schreiben der RWMOS Systemvariablen TRVL
BORLAND DELPHI:	function wrtrvl (var value:double) : integer;
C:	int wrtrvl (double *value);
VISUAL BASIC:	Function wrtrvl (value As Double) as long
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	TRVL ist die Interpolations Bahngeschwindigkeit, die bei Interpolationsbefehlen, welche aus der rw_SymPas Programmierumgebung aufgerufen werden, herangezogen wird (siehe auch *rw_SymPas* System-Parameter in Tabelle 34). Bei der PCAP-Programmierung wird dieser Parameter beim Aufruf in der Datenstruktur LMP, CMP oder HMP übergeben.

4.4.188 wrtrtvl, write trajectory target velocity

BESCHREIBUNG:	Schreiben der RWMOS Systemvariablen TRTVL
BORLAND DELPHI:	function wrtrtvl (var value:double) : integer;
C:	int wrtrtvl (double *value);
VISUAL BASIC:	Function wrtrtvl (value As Double) as long
RÜCKGABEWERT:	0 bei Erfolg -1 Kommando ist in RWMOS-Version nicht verfügbar -4 Timeout, Ursache unbekannt, Kommunikation mit der Steuerungsbaugruppe ist unterbrochen sonstiger Wert <> 0 unbekannter Fehler bei Befehlsausführung
ANMERKUNG:	TRTVL ist die Interpolations Bahn-Zielgeschwindigkeit, die bei Interpolationsbefehlen, welche aus der rw_SymPas Programmierumgebung aufgerufen werden, herangezogen wird (siehe auch *rw_SymPas* System-Parameter in Tabelle 34). Bei der PCAP-Programmierung wird dieser Parameter beim Aufruf in der Datenstruktur LMP, CMP oder HMP übergeben.

5 Die Programmiersprache rw_SymPas für die Standalone-Applikations-Programmierung

5.1 Einführung

rw_SymPas ist eine Programmiersprache zur Erzeugung von selbständig ablauffähigen CNC-Programmen (Stand-Alone-Applikations-Programmen) für die Positioniersteuerung MCU-G3. Die lexikalische und semantische Grammatik von *rw_SymPas* ist stark an die Programmiersprache *Pascal* angelehnt.

5.2 Lexikalische Grammatik

Dieses Kapitel enthält eine formale Definition der lexikalischen Grammatik von rw_SymPas. Diese befaßt sich mit den wortähnlichen Einheiten einer Sprache, sogenannten »Symbolen« oder »Tokens«. Die semantische Grammatik bestimmt die Regeln, nach denen Symbole zur Bildung von Ausdrücken, Anweisungen oder anderen Einheiten kombiniert werden können.
In *rw_SymPas* ergeben sich die Symbole als Folge der Operationen, die der *NCC*-Compiler mit dem Benutzerprogramm durchführt. Ein *rw_SymPas* Programm ist eine Abfolge von ASCII-Zeichen, die den Quellcode darstellen, der mit einem Texteditor (z.B. CNC-Edit) erstellt wurde. Die grundlegende Programmeinheit in *rw_SymPas* ist die Datei. Sie entspricht einer benannten DOS-Datei im Speicher oder auf der Platte und hat die Namenserweiterung .SRC.

5.2.1 Whitespace

In der lexikalischen Analysephase der Compilierung wird die Quellcodedatei in Symbole und »Whitespace« geparst (d.h. zerlegt). Whitespace ist der Sammelbegriff für Zeichen, die als Trenner gelten: Leerzeichen, Tabulatoren, Zeilenvorschübe und Kommentare. Whitespace dient zur Markierung von Beginn und Ende eines Symbols, aber abgesehen davon wird Whitespace ignoriert.

5.2.2 Kommentare

Kommentare sind Textzeilen, die Erklärungen zum Programm enthalten. Sie werden vor dem Parsen aus dem Quelltext entfernt.
Ein *rw_SymPas* Kommentar ist eine Zeichenfolge, die nach dem Zeichen { steht. Der Kommentar endet bei dem ersten Auffinden des Zeichens }, welches auf das Startsymbol { folgt. Die Verschachtelung von Kommentaren ist nicht zulässig.
Weiterhin ist es möglich einen einzeiligen Kommentar mit zwei Schrägstrichen // anzulegen. Der Kommentar kann an beliebiger Stelle beginnen und erstreckt sich bis zur nächsten Zeile.

5.2.3 Symbole

rw_SymPas kennt folgende Arten von Symbolen

Symbol:
Schlüsselwort
Bezeichner
Qualifizierte Bezeichner
Labels
Konstante
Operator
Interpunktionszeichen (auch Trennzeichen)

5.2.3.1 Schlüsselwörter

Schlüsselwörter sind für spezielle Zwecke reservierte Wörter, die nicht als normale Bezeichnernamen verwendet werden dürfen. In der folgenden Tabelle 23 sind alle *rw_SymPas* Schlüsselwörter aufgelistet.

Tabelle 23: Alle *rw_SymPas* Schlüsselwörter

and	begin	boolean	const
do	double	downto	else
end	for	forward	function
goto	if	integer	label
mod	module	not	or
procedure	repeat	shl	shr
single	then	timer	to
until	var	while	xor

5.2.3.2 Bezeichner

Bezeichner können aus folgenden Elementen bestehen:

Bezeichner
Nicht-Ziffer
Bezeichner Nicht-Ziffer
Bezeichner Ziffer

Nicht-Ziffer: eines der folgenden Zeichen
a b c d e f g h i j k l m n o p q r s t u v w x y z _
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Ziffer: eines der folgenden Zeichen
0 1 2 3 4 5 6 7 8 9

Beispiele:

```
A, AA, AB, A1, A2, _A          // gültig
1A, ?B                         // ungültig
```

5.2.3.2.1 Namen- und Längenbeschränkung

Bezeichner sind beliebige Namen von einer beliebigen Länge für Variablen, Prozeduren, Funktionen, Labelnamen, etc. Bezeichner können die Buchstaben A bis Z, a bis z, den Unterstrich und die Ziffern 0 bis 9 enthalten. Es gibt jedoch folgende Einschränkungen:

- Das erste Zeichen muss ein Buchstabe oder ein Unterstrich sein.
- Nur die ersten 32 Zeichen sind signifikant. Sofern der Bezeichner mehr als 32 Zeichen enthält, werden die restlichen Zeichen verworfen. Bei grossen *rw_SymPas* Programmen sollte man sich auf kurze Namen beschränken, um den Arbeitsspeicher des PC zu entlasten.
- In DIN G-Code Programmen sind keine numerischen Zeichen erlaubt.

5.2.3.2.2 Bezeichner Groß- und Kleinschreibung

In *rw_SymPas* wird zwischen Groß- und Kleinschreibung unterschieden, so dass *Position*, *position* und *positioN* unterschiedliche Bezeichner sind.

5.2.3.2.3 Eindeutigkeit und Gültigkeit von Bezeichnern

Bezeichner können beliebige Namen sein, die den geltenden Regeln entsprechen. Es kann jedoch zu Fehlern kommen, wenn derselbe Name innerhalb desselben Geltungsbereichs für mehrere Bezeichner verwendet wird, die denselben Namensbereich haben. Gleiche Namen sind für verschiedene Namensbereiche zulässig, unabhängig vom Geltungsbereich. Die Definition des Geltungsbereichs von Bezeichnern wird im Kapitel 5.3.2.2 erläutert.

5.2.3.3 Standardbezeichner

rw_SymPas definiert bereits eine Reihe von Bezeichnern, für die deshalb der Name Standardbezeichner verwendet wird. In der folgenden Tabelle 24 sind alle *rw_SymPas* Standardbezeichner aufgelistet.

Tabelle 24: Alle *rw_SymPas* vordefinierten Standardbezeichner

abort	cl	contcnct	disev
enev	ja	jaw	jhi
jhiw	jhl	jhlw	jhr
jhrw	jr	jrw	js
jsw	mca	mcaw	mcr
mcrw	mha	mhaw	mhr
mhrw	mla	mlaw	mlr
mlrw	ms	msw	ol
ra	rs	shp	smca
smcr	smha	smhr	smla
smlr	ssms	ssmsw	startcnct
stop	stepcnct	stopcnct	uf
wt	utrovr		

5.2.3.4 Achsenbezeichner

Jeder Achskanal wird mit Hilfe eines symbolischen Namens referenziert. Dieser Name kann mit bis zu 8 Zeichen vom Benutzer frei gewählt werden. Diese Achsenbezeichner werden von *rw_SymPas* ebenfalls in die Standardbezeichnerliste mitaufgenommen.
Anmerkung: Die automatische Deklaration der Achsenbezeichner weicht vom Standard-Pascal ab.

5.2.3.5 Qualifizierte Bezeichner

Der Bezug auf Bezeichner gleichen Namens, die für verschiedene Achssysteme (von *rw_SymPas*) deklariert sind, geschieht über eine Qualifizierung durch Voranstellen des Achsenbezeichners.
Beispiele:

```
A1.digo := 0;               // alle Ausgänge der MCU-G3 rücksetzen
A2.digo := $FFFFFFFF;       // alle Ausgänge der MCU-G3 setzen
```

Anmerkung: Der Variablenbezug auf qualifizierte Bezeichner weicht vom Standard-Pascal ab.

5.2.3.6 Labels

Für den Aufbau eines Labels gelten dieselben Regeln wie für die Bezeichner. Labels werden ausschließlich im Zusammenhang mit der Anweisung *goto* verwendet.

5.2.3.7 Konstanten

Konstanten sind Symbole, die für feste numerische Werte stehen. *rw_SymPas* kennt zwei Klassen von Konstanten: Gleitkomma und Integer. Der Datentyp einer Konstante wird vom *NCC*-Compiler auf Grund ihres numerischen Wertes und ihres Formats im Quelltext abgeleitet. Tabelle 25 zeigt die formale Definition einer Konstante.

Tabelle 25: Formale Definition einer Konstanten

Konstante: Gleitpunktkonstante Integerkonstante
Gleitpunktkonstante: Fraktionale-Konstante<Exponent> Ziffernfolge Exponent
Fraktionale Konstante: <Ziffernfolge>.Ziffernfolge Ziffernfolge.
Exponent: e<Vorzeichen>Ziffernfolge E<Vorzeichen>Ziffernfolge
Vorzeichen: Eines der folgenden Zeichen + -
Ziffernfolge: Ziffer Ziffernfolge Ziffer
Integer-Konstante: <Vorzeichen>Dezimale Konstante Hexadezimale Konstante
Dezimale-Konstante: Ziffer Dezimale-Konstante Ziffer
Hexadezimale-Konstante: $ Hexziffer Hexadezimale-Konstante Hexziffer
Ziffer: 0 1 2 3 4 5 6 7 8 9
Hexziffer:

0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F

5.2.3.7.1 Integer-Konstanten

Integerkonstanten können dezimale (Basis 10) oder hexadezimale (Basis 16) Zahlen sein. Zu beachten ist, dass für dezimale und nichtdezimale Konstanten unterschiedliche Regeln gelten.

5.2.3.7.1.1 Dezimalkonstanten

Es sind Dezimalkonstanten von -2147483648 bis 2147483647 zugelassen. Konstanten ausserhalb des Bereichs werden automatisch auf den entsprechenden Minimal- bzw. Maximalwert begrenzt.

5.2.3.7.1.2 Hexadezimale Konstanten

Alle Konstanten, die mit dem Dollarzeichen ($) beginnen, werden als hexadezimale Konstante interpretiert. Hexadezimale Konstanten von $80000000 bis $7FFFFFFF sind zugelassen. Konstanten ausserhalb des Bereichs werden auf den entsprechenden Minimal- bzw. Maximalwert begrenzt.

5.2.3.7.2 Gleitpunkt-Konstanten

Eine Gleitpunktkonstante setzt sich aus 4 Bestandteilen zusammen:

- Vorkommastellen
- Dezimalpunkt
- Nachkommastellen
- e oder E und ein vorzeichenbehafteter Integerexponent (optional)

Es können entweder die Vorkomma- oder Nachkommastellen wegfallen (aber nicht beide). Der Dezimalpunkt oder der Buchstabe e (E) kann weggelassen werden (aber nicht beide). Diese Regeln ermöglichen sowohl die konventionelle als auch die wissenschaftliche Notation (mit Exponenten).

5.2.3.7.2.1 Der Typ der Gleitkommakonstanten

Gleitkommakonstanten werden grundsätzlich als Double-Werte behandelt. Sie werden in einem Doppelwort (8 Byte) nach IEE abgelegt. Der Bereich ist $1.7*10^{-308}$ bis $1.7*10^{308}$.

5.2.3.7.2.2 Deklaration von Konstanten

Eine Konstanten-Deklaration vereinbart einen Bezeichner, der innerhalb des entsprechenden Blocks für einen konstanten Wert steht. Beispiel Konstantendeklaration:

```
Const           one = 1;
```

Vorzeichenbehaftete Konstanten stehen für einen Ganzzahl- oder Gleitkommawert. Die Berechnung von Konstanten ist nicht möglich.

5.2.3.7.3 Interpunktionszeichen

Interpunktionszeichen (auch Trennzeichen genannt) sind in *rw_SymPas* wie folgt definiert:

Interpunktionszeichen: eines der folgenden Symbole
() , ; : =

5.2.3.7.3.1 Runde Klammern

() fassen Ausdrücke zusammen, isolieren konditionale Ausdrücke und repräsentieren Prozeduraufrufe und Prozedurparameter:

```
d := c * (a + b);           // Ändern der normalen Reihenfolge
if (d = z) then ...         // Erforderlich bei einer bedingten
                            // Anweisung
proc()                      // Prozeduraufruf ohne Argumente
```

5.2.3.7.3.2 Komma

Das Komma (,) trennt die Elemente in einer Prozedur-Argumentliste:

```
mlr (A1, A2);
```

5.2.3.7.3.3 Strichpunkt

Der Strichpunkt (;) dient als Endekriterium einer Anweisung. Jeder gültige *rw_SymPas* Ausdruck (auch der leere Ausdruck), an dessen Ende ein Strichpunkt steht, wird als Anweisung (Ausdruck-Anweisung) interpretiert.

5.2.3.7.3.4 Gleichheitszeichen

Das Gleichheitszeichen (=) trennt Konstantendeklarationen von den Initialisierungswerten:

```
const one = 1.0;
```

5.3 Semantische Grammatik

In diesem Kapitel wird die formale Definition von der *rw_SymPas* Sprachstruktur erläutert. Diese semantische Grammatik bestimmt die Regeln, nach denen Symbole zur Bildung von Ausdrücken, Anweisungen oder anderen bedeutungstragenden Einheiten kombiniert werden können.

5.3.1 Deklarationen

Im folgenden Abschnitt befindet sich eine kurze Zusammenfassung über Themen, die mit Deklarationen zu tun haben: Objekte, Typen, Blöcke, Lokalität und Geltungsbereich. Lokalität und Geltungsbereich legen diejenigen Programmteile fest, von denen aus ein zulässiger Zugriff auf das mit dem Bezeichner verknüpfte Objekt möglich ist.

5.3.1.1 Objekte

Ein Objekt ist ein identifizierbarer Speicherbereich, in dem ein fester oder variabler Wert (oder eine Menge von Werten) steht. Jedes Objekt hat einen Namen und einen Typ (den sogenannten Datentyp). Der Zugriff auf das Objekt erfolgt über seinen Namen. Dieser Name kann ein einfacher Bezeichner oder ein komplexer Ausdruck sein, der eindeutig auf ein Objekt zeigt. Der Typ wird dazu verwendet um:

- die korrekte Speicherreservierung festzulegen, die zu Beginn erforderlich ist
- durch Überprüfung der Typen sicherzustellen, dass zulässige Zuweisungen erfolgen

Zu den vordefinierten Typen von *rw_SymPas* gehören der Datentyp Boolean, Integerzahlen mit Vorzeichen und Gleitkommazahlen mit unterschiedlicher Genauigkeit.
Deklarationen stellen die Verbindung zwischen Bezeichnern und Objekten her. Jede Deklaration verknüpft einen Bezeichner mit einem Datentyp. Darüber hinaus bestimmen die meisten Deklarationen, die sogenannten Definitionsdeklarationen, auch die Generierung des Objekts (wo und wann) und übernehmen die Zuweisung des Speicherplatzes.

5.3.1.2 Typen

Jede Deklaration einer Variablen muss den Typ dieser Variablen angeben. Der Typ legt den Wertebereich der Variablen fest und bestimmt die Operationen, die mit ihr ausgeführt werden können. Eine Typendefinition vereinbart also einen Bezeichner, der seinerseits für einen bestimmten Typ steht.

Typen-Deklaration:
    Bezeichner = Typ;

Typ:
    Boolean-Typ
    Ganzzahl-Typ
    Gleitpunkt-Typ

5.3.1.2.1 Boolean-Typ

Der Datentyp *Boolean* kann ausschließlich einen der vordefinierten Werte *FALSE* oder *TRUE* annehmen. Dabei gelten folgende Beziehungen:

- *FALSE* < *TRUE*
- Ordinalzahl von *FALSE* = 0
- Ordinalzahl von *TRUE* = 1

5.3.1.2.2 Ganzzahl-Typ

rw_SymPas stellt die Ganzzahl-Typen *Integer* und *Timer* zur Verfügung.

Tabelle 26: Der Ganzzahl-Typ und sein Wertebereich

Typ	Bereich	Format
Integer	-2147483648 .. 2147483647	32 Bit mit Vorzeichen
Timer	0 .. 4294967295	32 Bit ohne Vorzeichen

5.3.1.2.3 Gleitpunkt-Typen (Real-Typen)

rw_SymPas kennt zwei verschiedene Arten von Gleitpunkt-Typen: *Single* und *Double*. Die Typen unterscheiden sich voneinander sowohl durch ihren Wertebereich als auch in der Genauigkeit mit ihnen durchgeführter Operationen.
Anmerkung: Gelegentlich wird auch der Begriff Real-Typ für Gleitpunkt-Typ verwendet.

Tabelle 27: Die Gleitpunkt-Typen und ihre Genauigkeit

Typ	Bereich	Format
Single	$-1.2e^{-38}$.. $3.4e^{38}$	7 bis 8 Stellen
Double	$-2.2e^{-308}$ .. $1.8e^{308}$	15 bis 16 Stellen

5.3.1.2.4 Zuweisungskompatibilität von Typen

Die Zuweisungskompatibilität ist unabdingbar, wenn ein Wert zugewiesen werden soll. Der Wert eines Typs T_2 kann einem Wert T_1 zugewiesen werden (d.h. $T_1:=T_2$), wenn eine der folgenden Bedingungen erfüllt ist:

- T_1 und T_2 sind vom gleichen Typ.
- T_1 hat den Typ Double, T_2 den Wert Integer oder Single.
- T_1 hat den Typ Single, T_2 den Wert Integer.

Wenn keine dieser Bedingungen erfüllt, Zuweisungskompatibilität aber erforderlich ist, meldet der *NCC*-Compiler einen Fehler.

5.3.1.3 Variablen

5.3.1.3.1 Automatische Typ-Konvertierung

rw_SymPas führt eine automatische Typ-Konvertierung durch, sofern mit unterschiedlichen Typen in einem Ausdruck operiert wird. Die Konvertierung wird wie folgt ausgeführt: Integer nach Single oder Integer und Single nach Double. Beispiel:

```
...
Var
     i : Integer;
     s : Single;
     d : Double;
...

d := s * i;      // s und i werden automatisch nach Double konvertiert

s := i;          // i wird automatisch nach Single konvertiert
```

5.3.2 Blöcke, Lokalität und Geltungsbereich

Ein Block besteht aus beliebig angeordneten Deklarationen und Anweisungen. Jeder Block ist Teil einer Prozedur-Deklaration, einer Funktions-Deklaration oder eines Programms. Alle Bezeichner und Labels im Deklarationsteil des Blocks sind in ihrer Wirkung auf diesen Block beschränkt - sie sind lokal zu diesem Block.

5.3.2.1 Syntax

Der syntaktische Aufbau jedes Blocks läßt sich wie folgt darstellen:

Block:
  Deklarationsteil
  Befehlsteil

5.3.2.1.1 Deklarationsteil

Deklarationsteil:
  Label-Deklarationsteil
  Konstanten-Deklarationsteil
  Variablen-Deklarationsteil
  Deklarationsteil Label-Deklarationsteil
  Deklarationsteil Konstanten-Deklarationsteil
  Deklarationsteil Variablen-Deklarationsteil

5.3.2.1.1.1 Label-Deklarationsteil

Im *Label-Deklarationsteil* werden alle Labels vereinbart, die *goto*-Sprungziele im Befehlsteil des betreffenden Blocks darstellen sollen. Jedes Label darf innerhalb des Befehlsteils nur einmal definiert werden (d.h. jedes *goto* muss ein eindeutiges Ziel haben).

Aufbau des Label-Deklarationsteils:

label Labels;

Labels:
LabelName
Labels, LabelName

5.3.2.1.1.2 Konstanten-Deklarationsteil

Der Deklarationsteil für Konstanten beinhaltet alle Vereinbarungen von Konstanten, die lokal zum entsprechenden Block sind.

Aufbau des Konstanten-Deklarationsteils:

const Konstanten-Deklarationen

Konstanten-Deklarationen:
Konstanten-Deklaration
Konstanten-Deklarationen Konstanten-Deklaration

5.3.2.1.1.3 Variablen-Deklarationsteil

Der Deklarationsteil für Variablen beinhaltet alle Variablen-Deklarationen, die lokal zum entsprechenden Block sind.

Aufbau des Variablen-Deklarationsteils:

var Variablen-Deklarationen

Variablen-Deklarationen:
Variablen-Deklaration
Variablen-Deklarationen Variablen-Deklaration

5.3.2.1.2 Befehlsteil

Im Befehlsteil werden alle Operationen definiert, die bei der Aktivierung des Blocks ausgeführt werden.

Befehlsteil:
Verbundanweisung

Zulässige Verbundanweisungen werden im Kapitel 5.3.5.5 erläutert.

Der Befehlsteil des Hauptprogrammblocks hat folgenden Aufbau:

```
begin
        Anweisungsfolge;
end.
```

5.3.2.2 Geltungsbereich

Jeder Bezeichner und jedes Label einer Deklaration vereinbart genau ein Objekt bzw. ein Sprungziel. Deshalb muss ein Bezeichner, genauso wie ein Label, immer im Geltungsbereich seiner Deklaration sein, wenn er im Programm erscheint. Der Geltungsbereich von Bezeichnern und Labels liegt zwischen der eigentlichen Deklaration und dem Ende des dazugehörigen Blocks, wobei alle Blöcke mit eingeschlossen sind, die dieser Block umfaßt. Allerdings gibt es hierzu einige Ausnahmen, die in den folgenden Absätzen erläutert werden.

5.3.2.2.1 Neudeklaration in einem untergeordneten Block

Mit der Annahme, dass ein Block »Aussen« einen Block »Innen« umfaßt, d.h. ihm übergeordnet ist, beschränkt jede Neudeklaration eines Bezeichners von »Aussen« im Block »Innen« den Geltungsbereich dieses Bezeichners auf den Block »Innen«. Anders formuliert: Wenn »Aussen« eine Variable *x* deklariert und »Innen« eine Variable gleichen Namens, dann können Anweisungen im Block »Innen« nicht auf die in »Aussen« deklarierte Variable *x* zugreifen.

5.3.2.2.2 Der Ort einer Deklaration im Block

Bezeichner und Labels müssen deklariert sein, bevor sie in einem Block benutzt werden können. Auf Zugriffsversuche vor ihrer eigentlichen Deklaration reagiert der *NCC*-Compiler mit der Fehlernummer 3.

5.3.2.2.3 Neudeklarationen innerhalb eines Blocks

Bezeichner und Labels können auf der obersten Ebene eines Blocks nur jeweils einmal deklariert werden, es sei denn, ihre Neudeklaration erfolgt innerhalb eines untergeordneten Blocks.

5.3.2.2.4 Standardbezeichner

rw_SymPas bietet eine ganze Reihe vordefinierter Konstanten, Typen, Funktionen und Prozeduren, die so arbeiten, als wären sie innerhalb eines Blocks deklariert worden, der das ganze Programm umschließt. Folglich umfaßt ihr Geltungsbereich das gesamte Programm.

5.3.3 Variablen

5.3.3.1 Die Deklaration von Variablen

Die Variablen-Deklaration enthält eine Liste von Bezeichnern, die ihrerseits für neue Variablen und ihren jeweiligen Typ stehen.

Variablendeklaration:
Bezeichnerliste: Typ;

Bezeichnerliste:
Variablenname
Variablennamen, Variablenname

Typ:
BOOLEAN
INTEGER
SINGLE
TIMER
DOUBLE

Beispiele für gültige Variablendeklarationen sind:

```
var
     on, off:    BOOLEAN;
     eins:       INTEGER;
     dvalue:     DOUBLE;
     ticks:      TIMER;
```

Wenn ein Bezeichner in der Bezeichnerliste eines Deklarationsteils steht, gilt er innerhalb des gesamten Blocks, für den er deklariert wurde. Es kann über den ganzen Block hinweg auf diese Variable Bezug genommen werden, solange nicht in einem untergeordneten Block derselbe Bezeichner für eine andere Variable verwendet wird (»Redeklaration«). Eine redeklarierte Variable benutzt den Namen eines bereits existierenden Bezeichners, stellt aber ansonsten eine eigenständige Einheit dar. Der Wert der ursprünglichen Variablen wird durch die Redeklaration nicht beeinflußt. Variablen die außerhalb von Prozeduren oder Funktionen deklariert sind, werden als *global* bezeichnet. Innerhalb von Prozeduren oder Funktionen deklarierte Variablen sind *lokal*.

5.3.3.1.1 Axis-Typ-Deklaration

Mit Hilfe der AXIS-Typ Deklaration können variable Achsen definiert werden. Dies ist insbesondere für die Gestaltung von mehrfach verwendeten Unterprogrammen (Prozeduren / Funktionen) hilfreich in denen immer wiederkehrende Aktionen für verschiedene Achsen ausgeführt werden sollen.

Beispiel:

```
var
    VA : AXIS;                      // variable Achse mit Namen VA
    VA.an := 0;                     // Achsennummer 0 an VA zuweisen (wichtig!)
    ol(VA);                         // open loop von Achse 0
    for VA.an := 0 to 5 do  cl(VA); // close loop Achse 0 .. 5
```

Anmerkung: Auch die Achsnummer der vordefinierten Achsenspezifizierer (A1 .. An), welche bei den Systemparametern definiert sind, können auf diese Weise neu vergeben werden. Dies kann jedoch schnell zu einer unübersichtlichen und fehlerhaften Programmierung führen. Sobald mit variablen Achsen gearbeitet werden soll, empfiehlt es sich entsprechende Variablen mit entsprechendem Symbolcharakter zu verwenden.

Achtung: momentan darf die Deklaration von variablen Achsen nur im Hauptprogramm erfolgen, also nicht in Prozeduren oder Funktionen!

5.3.3.1.2 Timer-Deklaration

Eine Eingabe mit Hilfe der vordefinierten System-Variablen *CLOCK* liefert einen Wert vom Typ *Timer*, welcher die Zeit darstellt. Dieser Wert wird von einer internen Uhr der Steuerung geliefert, die in regelmäßigen Zeitabständen ihren Wert ändert. Dieser Wert läuft zyklisch weiter, d.h. nach dem größten positiven Wert wird als nächstes der kleinste negative Wert geliefert. Das Zeitintervall, in der diese interne Uhr hochgezählt wird, beträgt 64µs.
Mit Hilfe *von CLOCK* kann jederzeit der Zählerstand dieser Uhr einer *Integer* oder *Timer*-Variablen zugewiesen werden. Sofern verschiedene Zeiten miteinander zu vergleichen sind, sollte dieser Vergleich nur mit Hilfe von *Timer*-Variablen durchgeführt werden, da hier ein Timer-Überlauf beim Vergleichsoperator > automatisch berücksichtigt wird. Ebenso wird die Addition und Subtraktion mit *Timer*-Variablen in Modulo-Technik, also vorzeichenlos durchgeführt. Hingegen kann bei *Integer*-Variablen ein Über- bzw. Unterlauf stattfinden, welcher wiederum das Setzen eines Internen Error-Flags nach sich zieht und in bestimmten Situationen zum Abbruch des *rw_MOS*-Betriebssystems führt.
Eine praktische Timer-Anwendung könnte etwa so aussehen:

```
Const
     s := 15625;            // 15625 ticks = 1s
Var
     t: timer

t := CLOCK + 5*s;           // Verzögerung von 5s ab jetzt berechnen
...
repeat
     ...
until CLOCK > t;            // warten bis 5s vorbei
...
```

In diesem Beispiel wird ersichtlich, dass die Addition von *CLOCK* und der Verzögerungszeit bei großen Werten von *CLOCK* zum Überlauf führt. Deshalb empfiehlt es sich für die Deklaration der Variable *t* den *Timer* anstatt den *Integer*-Typ zu verwenden. Ein weiterer Grund ist die Abfrage auf das Erreichen der berechneten Verzögerungszeit. Im Fall eines Überlaufs bei der Berechnung der Verzögerungszeit ist der

Inhalt von *t* nämlich kleiner als *CLOCK*. Dieser Umstand wird durch die Deklaration als *Timer*-Variable ebenfalls richtig behandelt.
Der Wertebereich einer Timer-Variable liegt zwischen 0 und 4294967295. Es können Verzögerungszeiten bis zu 38h realisiert werden.

5.3.3.2 Umwandlung von Variablentypen

Der Bezug auf eine Variable eines bestimmten Typs kann in den Bezug auf eine Variable eines anderen Typs umgewandelt werden.

Typ Umwandlung:
Typbezeichner (Variablenbezug)

Typbezeichner:
BOOLEAN
INTEGER
SINGLE
DOUBLE

Einige Beispiele für die Umwandlung von Variablen-Typen:

```
var
     B : BOOLEAN;
     I : INTEGER;
     D : DOUBLE;

     B := BOOLEAN (I);
     B := BOOLEAN (D);
     D := B;
     I := INTEGER (D);
     I := B;
```

5.3.4 Ausdrücke

Ausdrücke bestehen aus Operatoren und Operanden. Die meisten Operatoren von *rw_SymPas* verknüpfen zwei Operanden und werden deshalb als binär bezeichnet. Die restlichen Operatoren arbeiten mit nur einem Operanden, daher bezeichnet man sie als unär. Binäre Operatoren benutzen die herkömmliche algebraische Form wie z.B. *a+b*. Ein unärer Operator steht immer unmittelbar vor seinem Operanden, wie bei *-b*. Bei umfangreichen Ausdrücken regelt die in Tabelle 28 gezeigte *Rangfolge* der Operatoren die Reihenfolge der Berechnung. Es gelten drei grundlegende Regeln:

- Ein Operand zwischen zwei Operatoren von unterschiedlichem Rang ist immer an den höherrangigen Operator gebunden.
- Ein Operand zwischen gleichrangigen Operatoren ist immer an den Operator gebunden, der links von ihm steht.
- Ausdrücke in Klammern werden als einzelner Operand betrachtet und immer als erstes ausgewertet.

Tabelle 28: Rangfolge der Operatoren

Operatoren	Rangfolge	Kategorie
-, +, not	1 (am höchsten)	unär
*, /, mod, shl, shr, and	2	multiplizierend
+, -, or, xor	3	addierend
=, <>, <, >, <=, >=	4	relational

Operationen von gleichem Rang werden normalerweise von links nach rechts durchgeführt.

5.3.4.1 Syntax von Ausdrücken

Die Rangfolge der Operatoren folgt der Syntax von Ausdrücken, die sich aus Faktoren, Termen und einfachen Ausdrücken zusammensetzen. Faktoren lassen sich durch folgende Syntax darstellen:

Faktor:
- Variablenbezug
- vorzeichenlose Konstante
- (Ausdruck)
- *not* Faktor
- Typ-Umwandlung (Werte)

vorzeichenlose Konstante:
- vorzeichenloser numerischer Wert
- Zeichenfolge
- Konstanten-Bezeichner

Die folgenden Angaben stellen gültige Faktoren dar:

Dummy	Variablenbezug
15	vorzeichenlose Konstante

5.3.4.2 Operatoren

Operatoren werden in vier Gruppen unterschieden: Arithmetische, logische, boolsche und relationale Operatoren.

5.3.4.3 Arithmetische Operatoren

Folgende Tabellen zeigen die Operanden- und Ergebnistypen binärer bzw. unärer arithmetischer Operationen.

Tabelle 29: Binäre arithmetische Operatoren

Operator	Operation	Operandentyp	Ergebnistyp
+	Addition	Integer, Real	Integer, Real
-	Subtraktion	Integer, Real	Integer, Real
*	Multiplikation	Integer, Real	Integer, Real
/	Division	Integer, Real	Integer, Real
mod	Modulo	Integer	Integer

Anmerkung: Sofern einer der Operanden den Typ *Timer* hat, werden Addition und Subtraktion mit Hilfe der Modulo-Technik durchgeführt. Es erfolgt keine Überlaufprüfung, da die *Timer*-Werte zyklisch sind. Weitere Angaben zum *Timer*-Typ sind im Kapitel 5.3.3.1.2 enthalten.

Tabelle 30: Unäre arithmetische Operatoren

Operator	Operation	Operandentyp	Ergebnistyp
+	Identität	Integer, Real	Integer, Real
-	Negation	Integer, Real	Integer, Real

Wenn beide Operanden eines Operators +, -, *, /, oder *mod* einen Integer-Typ haben, hat das Ergebnis ebenfalls den Typ *Integer*. Ist einer der Operanden eines Operators +, -, * oder / vom Typ *Real*, dann hat das Ergebnis ebenfalls den Typ *Real*.
Der Operator *mod* liefert den Rest der Division seiner Operanden zurück, also:

```
i mod j = i - (i/j)*j;
```

5.3.4.4 Logische Operatoren

Tabelle 31 gibt die Typen der Operanden und Ergebnisse logischer Operationen wieder.

Tabelle 31: Logische Operationen

Operator	Operation	Operandentyp	Ergebnistyp
not	bitweise Negation	Integer	Integer
and	bitweises UND	Integer	Integer
or	bitweises ODER	Integer	Integer
xor	bitweise Antivalenz	Integer	Integer
shl	Linksschieben	Integer	Integer
shr	Rechtsschieben	Integer	Integer

Anmerkung: *not* ist ein unärer Operator.

Die Operationen i *shl* j und i *shr* j verschieben den Wert von *i* um *j* Bitpositionen nach links bzw. nach rechts, entsprechen also einer Multiplikation bzw. Division mit 2^j.

5.3.4.5 Boolsche Operatoren

Tabelle 32 gibt die Typen der Operanden und Ergebnisse boolscher Operationen wieder.

Tabelle 32: Boolsche Operatoren

Operator	Operation	Operandentyp	Ergebnistyp
not	logische Negation	Boolean	Boolean
and	logisches UND	Boolean	Boolean
or	logisches ODER	Boolean	Boolean
xor	logische Antivalenz	Boolean	Boolean

Anmerkung: Der Operator **not** ist auch hier unär.

Bei Operanden des Typs *Boolean* bestimmt die normale boolsche Logik das Ergebnis dieser Operationen. Beispielsweise ergibt *a and b* nur dann *TRUE* (wahr), wenn *a* und *b* wahr sind.

5.3.4.6 Relationale Operatoren

Tabelle 33 gibt die Operandentypen und die Ergebnisse relationaler Operationen wieder.

Tabelle 33: Relationale Operatoren

Operator	Operation	Operandentyp	Ergebnistyp
=	gleich	Integer, Real	Boolean
<>	ungleich	Integer, Real	Boolean
<	kleiner als	Integer, Real	Boolean
>	größer als	Integer, Real	Boolean
<=	kleiner gleich	Integer, Real	Boolean
>=	größer gleich	Integer, Real	Boolean

Anmerkung: Sofern einer der Operanden den Typ *Timer* hat wird die Operation *größer als* (>) mit Hilfe der Modulo-Technik durchgeführt. Es erfolgt keine Überlaufprüfung, da die *Timer*-Werte zyklisch sind. Weitere Angaben zum *Timer*-Typ sind im Kapitel 5.3.3.1.2 enthalten.

5.3.5 Anweisungen

Der engl. Begriff *statement* steht für alle Konstrukte, die eine durch die MCU-G3 ausführbare Aktion vereinbaren. In diesem Hanbuch wird der Terminus »Anweisung« als Oberbegriff für Anweisungen (wie *begin, end* oder *for*) und *Befehle* (wie *goto*, Zuweisungen, Prozedur-Aufrufe usw.) verwendet.
Jeder Anweisung (d.h. jeder Vereinbarung einer ausführbaren Aktion) kann ein Label vorangestellt werden, auf das wiederum ein Bezug mit *goto* möglich ist: ein *goto* zu diesem Label bewirkt einen direkten Sprung zu dieser Anweisung und ihre Ausführung.

Aufbau einer Anweisung:

Label: Anweisung

Anweisung:
Zuweisung
Prozeduranweisung
goto-Anweisung

5.3.5.1 Zuweisungen

Zuweisungen ersetzen den momentanen Wert einer Variablen mit einem neuen Wert, der über einen Ausdruck angegeben wird.

Aufbau einer Zuweisung:

Variablenbezug **:=** Ausdruck

5.3.5.2 Prozedur- oder Funktionsaufrufe

Durch Angabe eines Prozedurbezeichners wird eine Prozedur aufgerufen, die mit diesem Bezeichner deklariert wurde. Die Übergabe von Parametern an die Prozedur wird erst ab mcfg.exe V2.5.3.97 (ncc.exe/dll V2.5.3.73) unterstützt. Durch Angabe eines Funktionsbezeichners wird eine Funktion aufgerufen, die mit diesem Bezeichner deklariert wurde. Die Verwendung von anwenderdefinierten Funktionen wird erst ab mcfg.exe V2.5.3.97 (ncc.exe/dll V2.5.3.73) und RWMOS.ELF ab V2.5.3.126 unterstützt.

5.3.5.3 Die goto-Anweisung

führt einen Sprung zum angegebenen Label aus: Das Programm wird an dem Punkt fortgesetzt, der unmittelbar auf das Label folgt. Die Syntax von *goto* ist:

goto Label

Bei Verwendung von *goto* müssen folgende Regeln beachtet werden:

- Das Label, auf das *goto* Bezug nimmt, muss im selben Block stehen wie die *goto*-Anweisung selber. Es ist nicht möglich, mit *goto* beliebig zwischen Prozeduren / Funktionen hin- und herzuspringen.
- Der Bezug auf einen strukturierten Anweisungsblock von einem Programmteil außerhalb dieses Blocks (d.h ein Sprung auf eine tiefere Verschachtelungsebene) kann unvorhersehbare Folgen haben. *rw_SymPas* kann derartige Fehler nicht erkennen.

5.3.5.4 Strukturierte Anweisungen

bestehen aus mehreren ineinander verschachtelten Ebenen, die ihrerseits Anweisungen enthalten. Sie werden entweder in der Reihenfolge ihres Erscheinens (Verbund-Anweisungen), bedingt (konditionale Anweisungen) oder wiederholt (Wiederholungsanweisungen bzw. Schleifen) ausgeführt.

Strukturierte Anweisung:
Blockbefehl
bedingte Anweisung
Wiederholungsanweisung

5.3.5.5 Verbundanweisungen

Die Verbundanweisungen legen fest, dass die einzelnen darin enthaltenen Komponenten in der Reihenfolge ausgeführt werden, in der sie im Quelltext erscheinen. Alle im Verbund enthaltenen Anweisungen werden als einziger Block behandelt und genügen so den Forderungen an Stellen, wo die Syntax von *rw_SymPas* nur eine einzelne Anweisung zulässt. Anfang und Ende eines Verbundes werden durch *begin* und *end* gekennzeichnet, die einzelnen Komponenten sind durch Semikolons voneinander getrennt.

Die Verbundanweisung lässt sich wie folgt darstellen:

begin Anweisungsfolge **end;**

Anweisungsfolge:
Anweisung;
Anweisungsfolge Anweisung;

Beispiel:

```
// ...
var
        i: Integer;
        j: Integer;
        temp: Integer;
// ...
begin
        if (i > 0) then i := 0;
        else begin
                // j und i vertauschen
                temp := i;
                i := j;
                j := temp;
        end;
end.
```

5.3.5.6 Bedingte Anweisungen

Bedingte Anweisungen bieten eine oder mehrere Optionen und wählen eine ihrer Komponenten (ggf. auch keine) zur Anweisung aus.

5.3.5.6.1 Die if-Anweisung

lässt sich wie folgt darstellen:

if (Bedingter-Ausdruck) **then** w-Anweisung <**else** f-Anweisung>

Die Klammern um *Bedingter-Ausdruck* sind nicht unbedingt erforderlich. Das Ergebnis von *Bedingter-Ausdruck* muss den Standardtyp *Boolean* haben. Ist *Bedingter-Ausdruck* TRUE, so wird *w-Anweisung* ausgeführt; andernfalls wird *w-Anweisung* ignoriert.
Ist das optionale *else f-Anweisung* vorhanden und *Bedingter-Ausdruck* wahr, so wird *w-Anweisung* ausgeführt; andernfalls wird *w-Anweisung* ignoriert und *f-Anweisung* ausgeführt.
Die Anweisungen *f-Anweisung* und *w-Anweisung* können selbst *if-Anweisungen* sein, wodurch verschachtelte Bedingungstest in nahezu beliebiger Tiefe ermöglicht werden. Bei verschachtelten *if..else*-Konstrukten muss sehr sorgfältig vorgegangen werden - es ist darauf zu achten, dass die korrekten Anweisungen ausgewählt werden. *Else*-Mehrdeutigkeiten werden gelöst, indem ein *else* dem letzten auf derselben Schachtelungstiefe aufgetretenen *if*-ohne-*else* zugeordnet wird. Für *w-Anweisung* und *f-Anweisung* sind auch Verbundanweisungen zulässig.

5.3.5.7 Schleifen

Schleifen (oder Wiederholungsanweisungen) legen die wiederholte Ausführung bestimmter Programmteile fest.

Schleife:
- while-Anweisung
- repeat-Anweisung
- for-Anweisung

5.3.5.7.1 Die while-Anweisung

Das Format für eine **while**-Anweisung lautet:

while (Bedingungsausdruck) **do** w-Anweisung;

Die Klammern um *Bedingungsausdruck* sind nicht unbedingt erforderlich. Die Schleifenanweisung *w-Anweisung* wird solange ausgeführt, bis der *Bedingungsausdruck* den Wert FALSE ergibt. Der *Bedingungsausdruck* wird vorher ausgewertet und getestet. Ergibt sich ein Wert, der TRUE ist (wahr), wird *w-Anweisung* ausgeführt. Wenn das Programm auf keine Sprunganweisungen trifft, die das Verlassen der Schleife zur Folge haben, wird der *Bedingungsausdruck* erneut ausgewertet. Dieser Vorgang wiederholt sich solange, bis *Bedingungsausdruck* den Wert FALSE ergibt. Gibt es keine Sprunganweisungen, muss *w-Anweisung* den Wert von *Bedingungsausdruck* beeinflussen oder *Bedingungsausdruck* selbst muss sich während der Auswertung ändern, damit Endlosschleifen vermieden werden. Für *w-Anweisung* sind auch Verbundanweisungen zulässig.

5.3.5.7.2 Die repeat-Anweisung

Das Format für eine *repeat*-Anweisung lautet:

repeat r-Anweisung **until** (Bedingungsausdruck);

Die Klammern um *Bedingungsausdruck* sind nicht unbedingt erforderlich. Die Anweisung *r-Anweisung* wird ausgeführt, solange *Bedingungsausdruck* den Wert FALSE (falsch) hat. Im Gegensatz zur *while*-Anweisung wird *Bedingungsausdruck* nicht vor, sondern nach jeder Ausführung der Schleifenanweisung getestet. *r-Anweisung* wird daher mindestens einmal ausgeführt.
Für *r-Anweisung* sind auch Verbundanweisungen zulässig.

5.3.5.7.3 Die for-Anweisung

Das Format für eine *for*-Anweisung lautet:

for Laufvariable := Startwert **to/downto** Endwert **do** f-Anweisung;

Die Laufvariable muss der Bezeichner einer Variablen des Typs *integer* sein, die entweder innerhalb desselben Blocks wie die *for*-Anweisung lokal oder global zum gesamten Programm deklariert ist. Die Definition einer Schleife mit *for* schließt die Festlegung eines *Start*- und *Endwert*es mit ein. Beide Werte müssen ebenfals vom Typ *integer* sein, der zu dem der Laufvariablen zuweisungskompatibel ist.
Beim Start der Schleife wird die Laufvariable auf den *Startwert* gesetzt und für jeden Schleifendurchlauf um eins erhöht bzw. erniedrigt - solange, bis der *Endwert* erreicht ist. Bei jedem Durchlauf wird die im Rumpf der Schleife enthaltene *f-Anweisung* bzw. Verbundanweisung einmal ausgeführt. Wenn die Endbedingung der Schleife bereits vor dem ersten Durchlauf gegeben ist (d.h. *Endwert* < *Startwert* bzw. *Endwert* > *Startwert* bei Verwendung von *downto*), dann werden Schleife und dazugehörender Rumpf komplett übersprungen.

5.3.6 Prozeduren und Funktionen

Prozeduren und Funktionen stellen formal gesehen zusätzliche Ebenen innerhalb des Hauptprogramm-Blocks dar, also eine Verschachtelung. Eine Prozedur wird durch einen Prozeduraufruf (d.h. die Angabe eines Bezeichners) aktiviert und liefert keinen direkten Wert zurück. Eine Funktion wird bei der Berechnung eines Ausdrucks aktiviert, in der ihr Bezeichner erscheint, und hat normalerweise ein Ergebnis, das für diesen Aufruf mit dem Funktionsbezeichner gleichgesetzt werden kann.

5.3.6.1 Prozedurdeklarationen

Eine Deklaration, die mit dem reservierten Wort *procedure* eingeleitet wird, verbindet einen Bezeichner und einen Block von Anweisungen zu einer Prozedur. Derartig deklarierte Prozeduren können durch Angabe ihres Bezeichners aktiviert (d.h. aufgerufen) werden. Eine Prozedurdeklaration hat folgenden formalen Aufbau:

Prozedurkopf; Prozedurblock;

Der Prozedurkopf benennt die Prozedur (d.h. ordnet ihr einen Bezeichner zu). Ab mcfg.exe V2.5.3.97 (ncc.exe/dll V2.5.3.73) ist an dieser Stelle die Deklaration von Parametern möglich. Ein Datenaustausch zwischen dem Hauptprogramm und einer Prozedur kann über globale Variablen oder über diese Parameter durchgeführt werden. Durch die Angabe ihres Bezeichners wird eine Prozedur aktiviert: die im Befehlsteil der entsprechenden Prozedurdeklaration definierten Aktionen werden ausgeführt.
Eine Prozedur, die ihre eigene Anweisung als Bestandteil ihres Befehlsteils enthält, wird rekursiv ausgeführt, d.h. sie ruft sich selber wiederholt auf. In diesem Zusammenhang muss ein geeignetes Kriterium für den Abbruch der Rekursion gefunden werden, bevor der interne CNC-Task-Stack überläuft.
Das Schachteln von Prozeduren in *rw_SymPas* ist nur möglich, wenn der Prozedurkopf mit optionalen Parametern als *forward* deklariert wurde. Hierzu muss nach dem Prozedurkopf das Schlüsselwort *forward* ebenfalls gefolgt von einem Semikolon erscheinen. Eine forward-Deklaration ist nur dann erforderlich, wenn die Prozedur vor dem Abschluss der eigentlichen Deklaration (mit Prozedurblock) verwendet werden soll. Zwischen Prozedurkopf und Prozedurblock können lokale Variablen und lokale Labels deklariert werden.
Die forward-Deklaration ist ebenfalls erst möglich ab mcfg.exe V2.5.3.97 (ncc.exe/dll V2.5.3.73).

Beispiele:

```
procedure ProcA; forward;
...
procedure ProcA;
begin
        (Prozedurblock)
end;

procedure ProcB (ParamA, ParamB : integer; ParamC : double); forward;
...
procedure ProcB (ParamA, ParamB : integer; ParamC : double);
begin
        (Prozedurblock mit Verwendung von ParamA, ParamB und ParamC)
end;

procedure ProcC (ParamA : integer);
var locVarA, locVarB : integer;
    locVarC : double;
begin
        (Prozedurblock)
end;
```

5.3.6.2 Funktionsdeklarationen

Anmerkung: Die nach *Pascal*-Standard implementierte Funktionendeklaration ist in *rw_SymPas ab* V2.5.3.97 (ncc.exe/dll V2.5.3.73) möglich. Zur Ausführung ist rwmos.elf ab V2.5.3.126 erforderlich, ansonsten wird ein rw_SymPas Programm mit dem Laufzeitfehler 4 beim Rückspung aus der Funktion beendet. Ausserdem gibt es verschiedene vordefinierte System-Funktionen.

Die Funktionen (Systemfunktionen und anwenderdefinierte Funktionen) werden bei der Berechnung von Ausdrücken aktiviert, in denen ihr Bezeichner erscheint, und stehen dort stellvertretend für den von ihnen zurückgelieferten Wert. Ein Funktionsbezeichner kann überall anstelle eines Operanden in einem Ausdruck eingesetzt werden, solange der Typ des Funktionsergebnisses mit dem des ersetzten Operanden kompatibel ist. Zuweisungen an einen Funktionsbezeichner sind nicht erlaubt. Der Aufruf einer Funktion geschieht über die Angabe ihres Bezeichners, gefolgt von einer Liste aktueller Parameter, die in Typ und Reihenfolge den formalen Parametern der entsprechend vordefinierten Funktion entsprechen müssen. Der Rückgabewert einer Funktion muss verwendet werden, ansonsten wird der Übersetzungsfehler 91 angezeigt.

Die Deklaration einer anwenderspezifischen Funktion, die mit dem reservierten Wort *function* eingeleitet wird, verbindet einen Bezeichner, Typdeklarationen und einen Block von Anweisungen zu einer Funktion. Derartig deklarierte Funktionen können durch Angabe ihres Bezeichners aktiviert (d.h. aufgerufen) werden. Eine Funktionsdeklaration hat folgenden formalen Aufbau:

Funktionskopf; Funktionsblock;

Der Funktionskopf benennt die Funktion (d.h. ordnet ihr einen Bezeichner zu) und definiert Parameter und Funktionstyp. Ein Datenaustausch zwischen dem Hauptprogramm und einer Funktion kann über globale Variablen oder über Parameter und den Rückgabewert durchgeführt werden. Durch die Angabe ihres Bezeichners wird eine Funktion aktiviert: die im Befehlsteil der entsprechenden Funktionsdeklaration definierten Aktionen werden ausgeführt.
Eine Funktion, die ihre eigene Anweisung als Bestandteil ihres Befehlsteils enthält, wird rekursiv ausgeführt, d.h. sie ruft sich selber wiederholt auf. In diesem Zusammenhang muss ein geeignetes Kriterium für den Abbruch der Rekursion gefunden werden, bevor der interne CNC-Task-Stack überläuft.
Das Schachteln von Funktionen in *rw_SymPas* ist nur möglich, wenn der Funktionskopf mit optionalen Parametern und Funktionstyp als *forward* deklariert wurde. Hierzu muss nach dem Funktionskopfkopf das Schlüsselwort *forward* ebenfalls gefolgt von einem Semikolon erscheinen. Eine forward-Deklaration ist nur dann erforderlich, wenn die Funktion vor dem Abschluss der eigentlichen Deklaration (mit Funktionsblock) verwendet werden soll. Im Funktionsblock muss der Rückgabewert an eine Variable mit der Bezeichnung des Funktionsnamens zugewiesen werden. Diese Variable für den Rückgabewert, kann innerhalb der Funktion als lokale Variable verwendet werden. Wenn ein Rekursiver Aufruf der Funktion erfolgen soll, muss der Funktionsname mit einem runden Klammernpaar verwendet werden, welches die Parameter enthält. Bei Funktionen ohne Parameter muss in diesem Fall der Funktionsaufruf durch ein leeres rundes Klammernpaar gekennzeichnet werden.
Zwischen Funktionskopf und Funktionsblock können lokale Variablen und lokale Labels deklariert werden (siehe Beispiel bei den Prozeduren).

Beispiele:

```
function FuncA  : integer; forward;
...
function FuncA  : integer;
begin
        (Funktionsblock)
        FuncA := Funktionsrückgabewert;
end;

function FuncB (ParamA, ParamB : integer; ParamC : double) : double; forward;
...
function FuncB (ParamA, ParamB : integer; ParamC : double) : double; forward;
begin
        (Funktionsblock mit Verwendung von ParamA, ParamB und ParamC)
        FuncB := Funktionsrückgabewert;
end;
```

Funktionsrückgabewert kann in obigen Beispielen ein beliebiger Ausdruck sein.

5.3.7 Die Syntax eines *rw_SymPas*-Programmes

Ein *rw_SymPas*-Programm hat eine ähnliche Form wie eine Prozedurdeklaration. Die Unterschiede liegen lediglich im Programmkopf.

rw_SymPas-Programm:

Programmkopf; Programmblock**.**

5.3.7.1 Der Programmkopf

Der Programmkopf legt den Namen eines Programms fest, hat aber keine besondere Bedeutung.

Programmkopf:
program Bezeichner

Beispiel:

```
program Test;
```

5.3.7.2 Der Programmblock

Programmblock:
Implementationsteil
Prozedur- / Funktions- Befehlsteil
Initialisierungsteil

Implementationsteil:
Konstanten-Deklaration
Variablen-Deklaration
Implementationsteil Konstanten-Deklaration
Implementationsteil Variablen-Deklaration

Initialisierungsteil:
begin
Befehlsteil
end

Der Initialisierungsteil ist der letzte Bestandteil eines *rw_SymPas*-Programms und stellt das Hauptprogramm dar. Er besteht aus einem mit **begin** eingeleiteten Block, der Anweisungen enthält und durch ein abschliessendes **end** beendet wird. Der gesamte Programmblock wird mit dem Zeichen **„.“** abgeschlossen.

6 Standalone-Applikations-Programmierung

6.1 Einführung

Die Programmiersprache *rw_SymPas* ist mit einem umfangreichen Befehlssatz ausgestattet, mit dessen Hilfe eine flexible und effiziente Programmerstellung ermöglicht wird. Die Prozeduraufrufe werden bis auf wenige Ausnahmen nach *Pascal*-Konvention durchgeführt.
Da die Prozedurnamen und auch die Funktionsweise der einzelnen Prozeduren für die beiden Programmiermethoden Standalone-Applikations-Programmierung [SAP] und PC-Applikations-Programmierung [PCAP] identisch sind, erfolgt eine detaillierte Beschreibung nur bei den Befehlen der PCAP-Programmierung.
Die Auflistung der einzelnen Befehle erfolgt in alphabetischer Reihenfolge.

6.2 rw_SymPas-Beispielprogramme

Die in der MCU-G3 TOOLSET Software enthaltenen *rw_SymPas* Beispielprogramme zeigen die einfache Anwendung nachfolgend beschriebener Funktionen. Die Quelltexte der Beispielprogramme sind durch Kommentare selbsterklärend. Deshalb wird an dieser Stelle auf eine detaillierte Programmbeschreibung dieser Beispiele verzichtet. Die Beispielprogramme haben alle den Dateierweiterungsnamen .SRC und sind im Unterverzeichnis SAP der MCU-G3 TOOLSET Software Diskette abgelegt.

6.3 Abkürzungen, System-Parameter, Achsenspezifizierer und Achsenqualifizierer

Für die nachfolgend abgedruckte SAP-Funktionenreferenzliste werden hier zunächst die einzelnen Abkürzungen und Typen erklärt, welche zum Teil als Parameter für die verschiedenen Funktionen dienen.

6.3.1 System-Parameter

Die von der *rw_SymPas*-Programmiersprache vordefinierten Systemparameter werden tabellarisch aufgelistet und deren Funktionsweise erläutert. Zu beachten ist, dass der *NCC*-Compiler bei diesen Parametern zwischen Groß- und Kleinbuchstaben unterscheidet.

Tabelle 34: *rw_SymPas* vordefinierte System-Parameter

Name	Typ	Kurzwortbedeutung	Funktion
BOARD TYPE	integer	Board-Typ	Hadwarevariante der Steuerungstyps (siehe Kapitel 4.4.18)
CI0..CI999	integer	Common Integer 0..999	1000 vordefinierte Integer-Variablen zum Datenaustausch oder zur Synchronisation mit einem parallel ablaufendem PC-ApplikationsProgramm. Weitere Infos bei den PCAP-Befehlen rdci() u. wrci().
CD0.. CD999	double	Common Double 0..999	1000 vordefinierte Double-Variablen. Sonst wie CI0..CI999. Weitere Informationen bei den PCAP-Befehlen rdcd() und wrcd().
CFLAG	integer	ControllerFlags	Zugriff auf das Register ControllerFlags (siehe Kapitel 6.3.1.4)
DTCA1	double	Distance-to-Center A1	Mittelpunktsangabe für Helix-Profile und 3D-Kreise für die X-Kreisachse
DTCA2	double	Distance-to-Center A2	Mittelpunktsangabe für Helix-Profile und 3D-Kreise für die Y-Kreisachse
DTCA3	double	Distance-to-Center A3	Mittelpunktsangabe für 3D-Kreise für die Z-Kreisachse
ERROR REG	integer	error register	Bitcodiertes Fehlerregister, in welchem interne Fehlerzustände von RWMOS.ELF angezeigt werden.
IRQPC	boolean	Interrupt Request PC	PC-Interrupt-Anforderung, aktiv wenn TRUE
LEDGN	boolean	Led green	Grüne Leuchtdiode auf MCU-G3, eingeschaltet wenn TRUE
LEDRD	boolean	Led red	Rote LED, sonst wie LEDGN
LEDYL	boolean	Led yellow	Gelbe LED, sonst wie LEDGN
LET	double	Latch End Time	Zeitpunkt für die Dauer der Aufzeichnung für die grafische Systemanalyse in Sekunden, vom Zeitpunkt LST an. Siehe hierzu die Befehle LPR und LPRS. Grundsätzlich werden immer 1000 Werte aufgezeichnet. Der in LET angegebene Wert wird immer in ganzzahlige Vielfache von 1000 * T_A aufgerundet. T_A ist standardmässig 1.28ms.
LST	double	Latch Start Time	Zeitpunkt für den Beginn der Aufzeichnung für die grafische Systemanalyse in Sekunden, vom Zeitpunkt des Aufrufs an. Siehe hierzu die Befehle LPR und LPRS.
MODEREG	integer	Mode Register	Bitcodiertes Register zur Steuerung der Betriebssystem-Funktionalität (siehe Kapitel 6.3.1.5)
NFRAX	integer	No-Feed-Rate-Axis	In dieser Variable können Achsen bitcodiert definiert werden, die bei Interpolationsbewegungen nicht für die Bahngeschwindigkeitsberechung herangezogen werden.

Tabelle 34 (Fortsetzung): *rw_SymPas* vordefinierte System-Parameter

Name	Typ	Kurzwortbedeutung	Funktion
NOA	integer	Number of Axis	Diese Systemvariable enthält die Anzahl der tatsächlich im System vorhandenen Achsen und darf nicht beschrieben werden.
OS VERSION	integer	Betriebssystem–Versionsinformation	Mit dem vordefinierten Systemparameter *OSVERSION* (Typ) kann die aktuelle Betriebssystemversionsnummer des *rwmo.elf*-Files zur Laufzeit in einem SAP-Programm abgefragt werden. Die Versionsnummer ist in eine Haupt und eine Nebennummer gegliedert, wobei die Hauptnummer in 1000er Schritten und die Nebennummer in 1er Schritten hoch gezählt wird. Die Versionsnummer 253042 z.B. bedeutet dann Hauptnummer 2.5.3 und Nebennummer 042.
PHI	double		Verfahrwinkel für Kreis- und Helix-Profile
PN1	double	Plane-Normal	Flächen-Normale für MCA3D Kommando. Weitere Informationen beim Kommando MCA3D.
PN2	double	Plane-Normal	Flächen-Normale für MCA3D Kommando. Weitere Informationen beim Kommando MCA3D.
PN3	double	Plane-Normal	Flächen-Normale für MCA3D Kommando. Weitere Informationen beim Kommando MCA3D.
PU	integer	Position Unit	Index für Positions-Einheit (Tabelle 35)
SSFP	integer	Spool-Special-Function-Parameter	Funktionsparameter für Spezialfunktionen in der Spooler-Betriebsart. Weitere Informationen beim SAP-Kommando *SSF*
TRAC	double	Trajectory Acceleration	Bahnbeschleunigung für Linear-, Kreis- und Helix-Profile. Die Einheit dieses Parameters ist in TU und PU festgelegt.
TROVR	double	Trajectory Override	Bahngeschwindigkeitskorrekturwert
TROVRST	double	Trajectory Override Settling Time	Zeit für Anpassung des Bahngeschwindigkeitskorrekturwertes
TRTVL	double	Trajectory Target Velocity	Bahnzielgeschwindigkeit für Linear, Kreis- und Helix-Profile. Die Einheit dieses Parameters ist in TU und PU festgelegt.
TRVL	double	Trajectory Velocity	Bahngeschwindigkeit für Linear, Kreis- und Helix-Profile. Die Einheit dieses Parameters ist in TU und PU festgelegt.
TU	integer	Time Unit	Index für Zeit-Einheit (Tabelle 36)

6.3.1.1 PC-Interrupt-Generierung

Mit Hilfe des System-Parameters *IRQPC* ist es möglich, auf dem PC einen Hardware-Interrupt auszulösen. Diese Möglichkeit gestattet eine effiziente Methode für den Einsatz der beiden Programmiermethoden PC-Applikations- und Standalone-Applikations-Programmierung. Mit Hilfe eines Stand-Alone-Programms kann ein weitgehend autarker Prozeßablauf erfolgen, der nur im Bedarfs- oder Fehlerfall das parallel ablaufende PC-Programm unterbrechen muss. Die Unterbrechung geschieht dann mit Hilfe dieser Interrupt-Generierung. Nach Erkennen des Hardware-Interrupts durch das PC-Programm könnte dann z.B. mit Hilfe der oben aufgeführten Common Variablen ein Datenaustausch zwischen den beiden parallel ablaufenden Programmen erfolgen.

Anmerkung zur PCI-Interruptverwaltung: Der Hardware-Interrupt (PCI-Interrupt) wird Dank der auf der MCU-G3 integrierten Plug and Play-Eigenschaften automatisch ermittelt und durch den Systemtreiber mcug3.dll verwaltet. Der Anwender muß lediglich eine PCAP-Benutzerroutine mit vorgegebenem Aufbau

definieren und diese dem Treiber bekannt machen. Sobald die MCU-G3 einen Hardware-Interrupt auslöst, wird die entsprechende Benutzerroutine automatisch aufgerufen. Der mcug3.dll-Treiber ist so gestaltet, dass auch andere PCI-Interrupts, die die gleiche Interruptquelle benutzen, aufgerufen werden. Die im Lieferumfang enthaltene Treibersoftware stellt Funktionen bereit um auf einfachem Wege eine Interrupt-Service-Routine zu installieren und bei Bedarf auch wieder zu deinstallieren (Kapitel 4.4.33 und 4.4.34).

6.3.1.2 Systemparameter für die Einheiten-Verarbeitung

Bei sämtlichen *move*-Befehlen der *rw_SymPas*-Programmiersprache erfolgt die Angabe der Beschleunigungs- *(TRAC)*, Geschwindigkeits- *(TRTVL, TRVL)* und Positionsparameter jeweils in angewählten Weg- und Zeiteinheiten. Mit den beiden nachfolgend aufgeführten Systemparametern ist es möglich, die Weg- (PU) und Zeit-Einheit (TU) jederzeit umzuschalten.

Tabelle 35: System-Parameter PU

Wert	Einheit	Kurzwortbedeutung
0	mm	Millimeter
1	inch	Inch
2	m	Meter
3	rev	Revolution
4	deg	Degree
5	rad	Radiant
6	counts	Counts
7	steps	Steps

Tabelle 36: System-Parameter TU

Wert	Einheit	Kurzwortbedeutung
0	sec	Seconds
1	min	Minutes
2	tsample	Sampling Time

Anmerkung: Die Default-Werte für TU und PU werden in dem Menü [Setup][Set CNC-specific parameters] in der CNC-Editor-Umgebung festgelegt.
Die gewählten Einheiten werden nur für Interpolationsbefehle (alle *move*-Befehle) herangezogen! Sofern es sich um achsspezifische Bewegungskommandos (alle *jog*-Befehle) handelt, werden die in *mcfg.exe* spezifizierten Achs-Einheiten berücksichtigt. Hier ist keine Umschaltung während der Laufzeit möglich.

6.3.1.3 ERRORREG

In diesem bitcodierten Register werden Laufzeitfehler der RWMOS-Betriebssystemsoftware angezeigt. Die Belegung der Bits kann Tabelle 18 in Kapitel 4.4.71.1 entnommen werden.

Das Register ERRORREG wird nur durch beschreiben mit 0 oder durch einen Systemboot zurückgesetzt.

6.3.1.4 ControllerFlags

Dies ist ein achsspezifisches bitcodiertes Register, mit dessen Hilfe sich Spezialoptionen im Lageregler der Steuerungsbaugruppen aktivieren lassen. Es gibt Optionen für das Verhalten des Lagereglers während des Verfahrens und im Stillstand. Dieses Register ist nur bei Servo-Achsen mit PID-Filtercharakteristik wirksam, bei Stepper-Achsen hat es keine Wirkung. Auf das Register kann zugegriffen werden mit den dll-Funktionen wrControllerFlags (Kapitel 4.4.147) und rdControllerFlags (Kapitel 4.4.59). Aus der rw_SymPas Programmierung kann auf das Register mit dem Achsenqualifizierer CFLAGS zugegriffen werden.

Tabelle 37: Beschreibung des Registers ControllerFlags

Bit # / Hex	Bez.	Erläuterung
0 / 0001H	***MoveControl***	Aktivierung aller Flags Move...
1 / 0002H	***MoveZero***	Ablöschen des Integralanteil des Achsreglers, sobald die Achse verfahren wird.
2 / 0004H	***MoveDeadBand***	Integralanteil des Achsreglers begrenzen, sobald die Achse verfahren wird und sich die Regeldifferenz der Achse ausserhalb des Totbandes befindet. Das Totband wird im ControllerParams Feld [8][0] in Digits angegeben. (siehe auch Kapitel 4.4.122)
3 / 0008H	***MoveFix***	Integralanteil des Achsreglers festhalten und nicht weiter aufintegrieren, sobald die Achse verfahren wird.
4 / 0010H		not used
5 / 0020H		not used
6 / 0040H		not used
7 / 0080H		not used
8 / 0100H	***StopControl***	Aktivierung aller Flags Stop...
9 / 0200H	***StopZero***	Ablöschen des Integralanteil des Achsreglers, sobald die Achse steht.
10 / 0400H	***StopDeadBand***	Integralanteil des Achsreglers begrenzen, sobald die Achse steht und sich die Regeldifferenz der Achse ausserhalb des Totbandes befindet. Das Totband wird im ControllerParams Feld [8][0] in Digits angegeben. (siehe auch Kapitel 4.4.122)
11 / 0800H	***StopFix***	Integralanteil des Achsreglers festhalten und nicht weiter aufintegrieren, sobald die Achse steht.
12 - 31		not used

Move... bedeutet, dass die entsprechenden Flags ihre Wirkung entfalten, sobald die entsprechende Achse verfährt. Stop... bedeutet, dass die entsprechenden Flags ihre Wirkung entfalten, sobald die entsprechende Achse sich in Lageregelung befindet. Wenn StopControl und MoveControl auf FALSE gesetzt sind, sind alle anderen Bits unwirksam. Zur Anzeige und zur testweisen Einstellung dieser Flags gibt es das Hilfsprogramm McuControlInit.exe, welches im Handbuch BHB beschrieben wird.

6.3.1.5 MODEREG

Mit Hilfe dieses bitcodierten Registers können diverse Optionen der Betriebssystemsoftware RWMOS.ELF eingestellt werden. Defaultmässig sind alle Bits auf 0 eingestellt.

Hinweis: Wenn in diesem Register ein Bit gesetzt oder rückgesetzt werden soll, muss i.A. darauf geachtet werden, dass die anderen Bits nicht verändert werden. Dazu ist es notwendig, vor Schreiben auf MODEREG, dieses zunächst zu lesen, den Inhalt mit Hilfe boolescher Operationen zu bearbeiten und dann wieder zurückzuschreiben. Setzen von Bits führt man mit boolescher Oder-Verknüpfung, Rücksetzen von Bits mit boolescher Und-Verknüpfung durch. Bits die i.M. nicht belegt sind, dürfen nicht verwendet werden, da diese für zukünftige Erweiterungen reserviert sind.

Tabelle 38: Bitcodierung MODEREG

Bit # / Hex	Bez.	Erläuterung
0 / 0001H	***LookAhead***	Mit diesem Bit wird die Look-Ahead-Funktionalität der RWMOS-Betriebssystemsoftware aktiviert. Hierbei wird die angegebene Zielgeschwindigkeit von Interpolationsprofilen so begrenzt, dass der maximale achsspezifische Geschwindigkeitssprung MDVEL bei keiner Achse überschritten wird und dass am Ende der Interpolationsfahrt alle Achsen stehen. Dieser Modus bezieht sich nur auf die Kommandos SMLA, SMLR, SMCA, SMCR, SMHA, SMHR. Des weiteren wird in diesem Modus die Bahngeschwindigkeit in Kreis-Befehlen so begrenzt, dass bei keiner der beteiligten Achsen die achsspezifische Maximum-Acceleration (MaxAcc) überschritten wird. Die maximale Geschwindigkeit ergibt sich aus √(Kreisradius * >MaxAcc). Siehe hierzu auch Kapitel 4.4.171, 4.4.93 und 6.3.3.
1 / 0002H	***S-Profil***	Durch Setzen dieses Bits werden die Beschleunigungs- und Bremsrampen mit S-förmigem Geschwindigkeitsanstieg / -abfall ausgeführt. Diese Option kann mit dem Achsenqualifizierer JERKREL parametriert werden.
2 / 0004H		frei für zukünftige Verwendung
3 / 0008H	***WkzRadKorr***	Werkzeug-Radius-Korrektur ein (nur bei optionTC) Zur Werkzeug-Radius-Korrektur existiert ein gesondertes Handbuch.
4 / 0010H		frei für zukünftige Verwendung
5 / 0020H	***AutoSpool***	Wenn Verfahrprofile in SAP-Programmen gespoolt werden, wird bei aktiver Option jeweils der Spooler geprüft. Wenn der Spooler voll ist, wird bei den im aktuellen Profil selektierten Achsen die Spoolerabarbeitung automatisch gestartet. Weitere Profile werden nur eingetragen, wenn wieder Speicherplatz frei ist. Diese Option wird für folgende Verfahrbefehle unterstützt: SMLA, SMLR, SMCR, SMCA, SMHA, SMHR und G01 bei DIN66025
6 / 0040H	***NoTriangle***	Dreiecksprofile im LookAhead-Modus werden unterdrückt. Falls ein Teilprofil die programmierte Bahngeschwindigkeit nicht erreichen kann, wird die aktuelle Startgeschwindigkeit beibehalten. Dadurch wird bei kurzen Verfahrprofilstücken ein glatterer Lauf ohne ständige Beschleunigungs- und Bremsphasen erreicht.
7 / 0080H	***ChkMaxVel***	In diesem Modus, wird bei gespoolten Linear-und Kreis-Interpolationsbefehlen die Bahngeschwindigkeit und die Bahnbeschleunigung aller Achsen so begrenzt, dass keine Achse die in MAXVEL und MAXACC gesetzen Maximalwerte überschreitet. In diese Überwachung werden auch No-Feedrate-Achsen mit einbezogen (siehe Tabelle 34 – NFRAX).
8 / 0100H	***ExactTargetPos***	Normalerweise wird am Ende eines jeden Verfahrprofils, welches die Zielgeschwindigkeit 0 hat, die Sollposition auf ganzzahlige Werte der Systemauflösung gerundet. Dies ist bei Schrittmotorsystemen z.B. ein Schritt oder bei Enkodersystemen ein Enkoder-Zählimpuls. Dadurch kann sich, beim hintereinanderhängen von Relativprofilen ein Fehler aufsummieren. Diese Rundung kann durch Setzen dieses Bits abgeschaltet werden.
9 / 0200H	***ShortestRotatoric Distance***	Wenn dieses Bit gesetzt ist, werden bei rotatorischen Achsen Jog-Absolut-Befehle (JA) in die Richtung ausgeführt, in die der kürzeste Verfahrweg notwendig ist.

Tabelle 38 (Fortsetzung): Bitcodierung MODEREG

Bit # / Hex	Bez.	Erläuterung
10 / 0400H	***RotatoricUnit***	Wenn dieses Bit gesetzt ist, wird die Zielposition / der Verfahrweg in der achsspezifischen rotatorischen Einheit angegeben bei rotatorischen Achsen die mit translatorischen Achsen per Interpolationsbefehl verfahren werden sollen.
11 / 0800H	***ForbidTargetVel***	Wenn dieses Bit gesetzt ist, wird ein Systemreset (rs) durchgeführt, wenn Verfahrprofile mit einer Zielgeschwindigkeit <> 0 beendet werden. In diesem Fall wird in der Systemvariablen ErrorReg das Bit 1 gesetzt.
12 / 1000H	***CenterAlwaysRel***	Kreismittelpunkte bei G-Codes G02 und G03 immer als Relativkoordinaten interpretieren
13 / 2000H	***NoLsmCheck***	Durch Setzen dieses Flags, kann die automatische Spoolerüberwachung bei G01 des G-Code-Interpreters (McuWIN) abgeschaltet werden.
14 / 4000H		frei für zukünftige Verwendung
15 / 8000H	***MS_DECEL***	Beim Kommandos MotionStop (ms) als Bremsbeschleunigung den Wert von TRAC verwenden unter Berücksichtigung der aktiven Interpolationseinheiten
16 / 1 0000H bis 23/80 0000H		frei für zukünftige Verwendung
24 / 0100 0000H	***SimulationMode***	Mit diesem Bit kann die Steuerung in den Simulationsmodus versetzt werden. In diesem Modus, werden keine Stellgrößen an die Antriebssysteme ausgegeben, der Verlauf der Istposition wird simuliert. **Vorsicht:** Eine Drift der Achsen muss vom Anwender verhindert werden, da in diesem Modus der Lageregler nicht aktiv ist.
25 / 0200 0000H	***OvrMode***	Wenn dieses Bit gesetzt ist, wird beim Aufruf des Kommandos ctru der Jog-Override der selektierten Achsen nicht beeinflusst.
26 / 0400 0000H	***StopAtWriteln***	Wenn dieses Bit gesetzt ist, bewirkt das SAP-Kommando writeln einen Stopp der jeweiligen Task. In diesem Fall wird im Register running der Datenstruktur CNCTS (Kapitel 4.3.2.10) zusätzlich das Bit 2 gesetzt. Dieser Modus kann zur lückenlosen Verarbeitung von Ausgabestrings in einem überlagerten Programm verwendet werden.
27 / 0800 0000H	***ClearZeroPosition***	Wenn dieses Bit gesetzt ist, wird die mit szpa / szpr gesetzte Nullpunktverschiebung gelöscht. Die aktuellen Positionswerte bleiben jedoch erhalten. Bei gesetztem Bit kann somit z.B. jederzeit die Referenzposition mit szpr verschoben werden.
28 / 1000 0000H	***JSatSAF***	JOG-Stop at Spooler-Asynchronous-Flag: Wenn dieses Bit gesetzt ist, werden im Falle des Auftretens eines SAF-Flags im AXST-Register alle Achsen mit der programmierten Stop-Deceleration angehalten. Im Fehlerfall wird auch das Bit 19 im ErrorReg gesetzt.
29 / 2000 0000H	***InhibitProfile Refuse***	Normalerweise werden Interpolations-Verfahrprofileohne Verfahrweg oder mit Geschwindigkeit/Beschleunigung = 0 automatisch verworfen und eine Fehlermeldung im fwsetup Monitor Screen wird generiert. Mit diesem Bit kann die Ausgabe einer Fehlermeldung beim Verwerfen von Verfahrprofilen unterdrückt werden.
folgende Bits		derzeit nicht belegt, reserviert für zukünftige Verwendung

6.3.2 Achsen-Spezifizierer

Die verschiedenen Achskanäle werden mit einem symbolischen Namen referenziert. Diese Namen können im Programm *mcfg.exe* vom Anwender frei gewählt werden. In der Programmiersprache *rw_SymPas* werden diese Namen automatisch vordefiniert und dienen im Anwenderprogramm bei verschiedenen Befehlen als Parameter. Zu beachten ist, dass der *NCC*-Compiler bei den Achsen-Spezifizierern zwischen Groß- und Kleinschreibung unterscheidet.

6.3.3 Achsen-Qualifizierer

Die nachfolgend aufgeführten Systemparameter verstehen sich als Achsen-Qualifizierer und sind deshalb für alle im System vorhandenen Achskanäle und damit für alle Achsen-Spezifizierer verfügbar. Mit Hilfe dieser Parameter können verschiedene achsspezifische Daten abgefragt bzw. gesetzt werden. Zu beachten ist, dass der *NCC*-Compiler bei diesen Parametern zwischen Klein- und Großschreibung unterscheidet. Der Bezug auf einen Achsen-Qualifizierer wird durch Angabe eines Achsen-Spezifizieres, das Zeichen '.' und dem Achsen-Qualifizierer hergestellt. Dies wird mit nachfolgendem Beispiel ersichtlich:

```
...
var
     input: integer;
...
input := A2.digi;            // Digitaleingänge von Achskanal 2
                             // einlesen
...
```

Tabelle 39: Achsen-Qualifizierer

Name	Typ	Kurzwortbedeutung	Funktion
an	integer	axis number	Der Achsenqualifizierer an beinhaltet die Achsnummer des entsprechenden Achsenbezeichners. Der Qualifizierer kann im Zusammenhang mit „variablen" Achsnamen verwendet werden. [Kapitel 5.3.3.1.1]
asms	integer	ASM Status Word	nur bei MCU-6000
aux	double	Auxiliary Register	Der Inhalt dieses Registers ist Optionsabhängig. Falls das System die optionEV (Encoder-Verification) enthält, kann über diese Variable auf den Enkoder-Zählerstand bei Schrittmotorsystemen zugegriffen werden. In diesem Fall ist die Einheit des Registers Counts.
axst	integer	axis status	Error-, Status- und Profil-Flags (wortweise)
digi	integer	digital inputs	Digital-Eingänge der MCU-G3 (wortweise) Verschiedene Flags dieses Registers können durch Zuweisung eines beliebigen Wertes an dieses Register gelöscht werden [Kapitel 4.4.60.1]
digo	integer	digital outputs	Digital-Ausgänge der MCU-G3 (wortweise)
dp	double	desired position	Soll-Position des Achskanals

Tabelle 39 (Fortsetzung): Achsen-Qualifizierer

Name	Typ	Kurzwortbedeutung	Funktion
dpoffset	double	desired position offset	In diesem Register kann ein Positionsoffset für den Lageregler in der achsspezifischen Benutzereinheit eingetragen werden. Dieses Register kann für eine überlagerte Lageregelung z.B. bei Steppern mit Enkoderverifikation verwendet werden. Dieses Register steht für Steppersysteme ab RWMOS-Version 2.5.2.23, für Servosysteme ab RWMOS-Version 2.5.2.29 zur Verfügung.
dv	double	desired velocity	Soll-Geschwindigkeit des Achskanals
dvoffset	double	desired velocity offset	In diesem Register kann eine Änderungs-geschwindigkeit des Positionsoffset (dpoffset) für den Lageregler in der achsspezifischen Benutzereinheit eingetragen werden.
effradius	double	Effektiv Radius	Bei Teilnahme von rotatorischen Achsen an translatorischen Interpolationsfahrten: achsenspezifischer Parameter für Umrechung von rotatorischen in translatorische Größen, (Mantelflächenbearbeitung) [Kapitel 2.3.4]
jerkrel	double	Jerk Relativ	Parameter für S-Geschwindigkeitsprofile
jovr	double	jog override	Geschwindigkeitsfaktor
jtvl	double	jog target velocity	Zielgeschwindigkeit für jog-Befehle
jvl	double	jog velocity	Geschwindigkeit für jog-Befehle
kd	double		PIDF-Filter-Koeffizient für Differentation
kfca	double		PIDF-Filter-Koeffizient zur Vorwärtskompensation für Beschleunigung
kfcv	double		PIDF-Filter-Koeffizient zur Vorwärtskompensation für Geschwindigkeit
ki	double		PIDF-Filter-Koeffizient für Integration
kp	double		PIDF-Filter-Koeffizient für Verstärkung
kpl	double		PIDF-Filter-Koeffizient zur zus. Phasen-Voreilung
lp	double	latched position	gelatchter Positionswert
lpndx	double	latched position index	gelatchter Positionswert mit Index-Signal (Nullspur)
lsm	integer	left spool memory	freier Spoolbereich [Bytes]
maxacc	double	maximum acceleration	Achsspezifische Maximalbeschleunigung im ChkMaxVel-Modus
maxvel	double	maximum velocity	Achsspezifische Maximalgeschwindigkeit im ChkMaxVel-Modus
mcis	integer	Move Commands in Spooler	In diesem Register wird angezeigt, wie viele Verfahrkommandos derzeit im Spooler enthalten sind. Dadurch kann der Abarbeitungszustand des Spoolers festgestellt werden. Diese Information kann verwendet werden, wenn der aktuelle Bearbeitunszustand nach Unterbrechung fortgesetzt werden soll (siehe auch PCAP-Kommando rdMCiS).
mcp	integer	Motor Command Port	Servo-Motoren: Sollwert für Analog-Port Stepper-Motoren: Schrittsignal für Schrittmotor-Leistungsendstufen Zusätzliche Beschreibung bei den Kommandos wrmcp (Kapitel 0) und rdmcp (Kapitel 4.4.96).

Tabelle 39 (Fortsetzung): Achsen-Qualifizierer

Name	Typ	Kurzwortbedeutung	Funktion
mdvel	double	maximum velocity skip	Achsspezifischer maximaler Geschwindigkeitssprung im Look-Ahead-Modus
mpe	double	maximum position error	maximal erlaubter Schleppfehler
poserr	double	position error	In diesem Register wird der aktuelle achspezifische Schleppfehler in der Benutzereinheit angezeigt. Dies ist der in Echtzeit berechnete Wert dp – rp.
pprev	double	Pulses Per Revolution	In diesem Register kann die Anzahl der Enkoderimpulse pro Umdrehung (antriebsseitig) gelesen werden. Bei Stepperachsen und bei Linearachsen bzw. wenn die Nennereinheit von slsp eine Lineareinheit ist wird hier 0 zurückgeliefert. Gegenüber slsp ist hier eine eventuelle Inpulsvervierfachung berücksichtigt.
rp	double	real position	Ist-Position des Achskanals
rv	double	real velocity	Ist-Geschwindigkeit des Achskanals [Kapitel 4.4.105], kann nur gelesen, nicht zugewiesen werden
sdec	double	stop deceleration	Stopverzögerung des Achskanals
sf	integer	special function	Applikationsspezifisches Sonderregister.
epc	integer	EEPROM programming cycles	Anzahl der Programmierzyklen
gcr	integer	gear configuration register	Mit Hilfe dieses Registers, kann die Gear-Funktionalität der MCU-3000 gesteuert werden. Die Verwendung dieses Registers wird auch bei der Beschreibung der Gear-Funktionalität in G3-Resourcen-Interface.pdf beschrieben.
gf	double	gear factor	Über diese Variable kann auf den achsspezifischen Getriebefaktor zugegriffen werden. Eine Zuweisung an diesen Wert darf nun in besonderen Fällen erfolgen.
gfaux	double	Gear Faktor Aux-Register	Angabe eines Übersetzungsverhältnis zwischen aux-Register und Sollposition für Systeme mit Puls-Richtungs-Schnittstelle **Beispiel:** 1000 Schritte / Umdrehung Enkoder mit 500 Strichen ergibt mit Vervierfachung 2000 Impulse pro Umdrehung. Hier muss der Wert 0,5 in gfaux eingetragen werden.
ifs	integer	interface status	Interface Status-Flags der MCU-G3 (wortweise) Verschiedene Flags dieses Registers können durch Zuweisung eines beliebigen Wertes an dieses Register gelöscht werden [Kapitel 4.4.78.1]
hac	double	home acceleration	Beschleunigung für home-Befehle
hvl	double	home velocity	Geschwindigkeit für home-Befehle
ipw	double	in position window	Positionsabhängiges Zielfenster
jac	double	jog acceleration	Beschleunigung für jog-Befehle
sll	double	software limit left	linke Software-Endlage
slr	double	software limit right	rechte Software-Endlage
slsp	double	Slits or Stepper Pulses	In diesem Register kann die Anzahl der Enkoderstriche pro Umdrehung (antriebsseitig) bzw. die Anzahl der Schritte pro Umdrehung bei Schrittmotoren gelesen und gesetzt werden. Vervierfachung und Einheiten entsprechen den in mcfg gesetzten Werten.

Tabelle 39 (Fortsetzung): Achsen-Qualifizierer

Name	Typ	Kurzwortbedeutung	Funktion
tp	double	target position	Ziel-Position des Achskanals
zerooffset	double	Zero-Offset	Absolutwert der Nullpunktverschiebung siehe auch PCAP-Kommandos szpa und szpr bzw. azo

Eine detaillierte Funktionsbeschreibung dieser Qualifizierer (Tabelle 39) kann bei den entsprechenden *rdxxxx()*- bzw *wrxxxx()*-Befehlen in der Funktionenreferenzliste der PCAP-Programmierung nachgelesen werden. *Beispielsweise wird die Bedeutung des Qualifizierers digo beim Befehl wrdigo() erklärt.*
Ausnahme: Die PIDF-Filterkoeffizienten werden gemeinsam mit dem SAP-Befehl *UF()* wirksam. Das Lesen bzw. Beschreiben dieser Koeffizienten erfolgt auf PCAP-Ebene mit den Befehlen *rdf()* und *uf()*.

6.3.4 Strukturierte Achsen-Qualifizierer

Die nachfolgend aufgeführten Systemparameter verstehen sich als strukturierte Achsen-Qualifizierer und sind deshalb für alle im System vorhandenen Achskanäle und damit für alle Achsen-Spezifizierer verfügbar. Mit Hilfe dieser Parameter können verschiedene achsspezifische Daten bitweise abgefragt bzw. gesetzt werden. Zu beachten ist, dass der *NCC*-Compiler bei diesen Parametern zwischen Klein- und Großschreibung unterscheidet. Der Bezug auf einen strukturierten Achsen-Qualifizierer wird aus folgendem Beispiel ersichtlich:

```
...
const
     enable = 1;
var
     input: boolean;
...
input := A2.digib.enable;    // Digitaleingang 1 von Achskanal 2 (I1)
                             // einlesen
A1.digob.7 := TRUE;          // Digitalausgang 7 (O7) setzen
...
```

Tabelle 40: Strukturierte Achsen-Qualifizierer

Name	Typ	Kurzwortbedeutung	Funktion
digib	boolean	digital-input-bit	Digital-Eingänge der MCU-G3 (bitweise)
digob	boolean	digital-output-bit	Digital-Ausgänge der MCU-G3 (bitweise)
ifsb	boolean	interface-status-bit	Status-Flags der MCU-G3 (bitweise)
axstb	boolean	axis status-bit	Error-, Status- und Profil-Flags (bitweise)

Die Funktion dieser Qualifizierer kann bei den entsprechenden *rdxxxxb()*- bzw *wrxxxxb()*-Befehlen in der Funktionenreferenzliste der PCAP-Programmierung nachgelesen werden. Beispielsweise wird die Bedeutung des Qualifizierers *digib* beim Befehl *rddigib()* erklärt.
Anmerkung: Die Bit-Zählweise bei den strukturierten Achsenqualifizieren beginnt bei 1!

6.3.5 Abkürzungen

Vorab werden einige in der Funktionen-Referenzliste verwendete Abkürzungen erläutert:

Tabelle 41: Abkürzungen

Name	Beschreibung
A1	Symbolischer Name für den ersten Achskanal. Dieser Name kann in mcfg.exe frei gewählt werden. Wird hauptsächlich bei Beispielen verwendet.
A2	Symbolischer Name für den zweiten Achskanal. Sonst wie A1.
Spec	Achsen-Spezifizierer wie z.B. A1 oder A2
Qual	Achsen-Qualifizierer wie z.B. digi, digib, digo, digob, axst usw.
Pos	Positionssollwert (Datentyp double)
Event	Prozedur mit Funktion als Eventhandler

6.4 Reservierte Prozedur-Namen mit Event-Funktion

In *rw_SymPas* sind eine Reihe von Prozedur-Namen mit Ereignis-Funktion vordefiniert. Sofern Prozedurdefinitionen mit diesen Prozedur-Namen im Anwenderprogramm erfolgen, kann die CNC-Task mit einem Freigabebefehl dazu veranlasst werden, diese Prozeduren beim Eintreffen eines prozedurspezifischen Ereignisses (Event) automatisch aufzurufen. Diese Prozeduren werden deshalb auch als Event-Handler bezeichnet.
Anmerkung: Die Events werden nach jeder Ausführung einer rw_SymPas-Anweisung abgeprüft. Hierbei ist unbedingt zu beachten, daß die entsprechenden Events nicht mehr überprüft werden, . wenn eine Task beendet ist. Wenn eine dauerhafte Eventüberwachung notwendig ist, muß die entsprechende Task in einer Endlosschleife verbleiben.

6.4.1 Event-Prozedur EVTOASM

Die Definition der Prozedur EVTOASM und Freigabe des entsprechenden Ereignisses bewirkt den automatischen Aufruf dieser Prozedur, sofern ein ASM-2003-Timeout-Fehler eintrifft. Sofern mehrere ASM-2003 im System vorhanden sind, kann mit dem Statusregister *axst* geprüft werden, welches ASM-2003 den Fehler verursacht. Diese Funktion ist nur bei der MCU-6000 (APCI-8401) sinnvoll.

6.4.2 Event-Prozedur EVEO

Die Definition der Prozedur EVEO und Freigabe des entsprechenden Ereignisses bewirkt den automatischen Aufruf dieser Prozedur. Das Ereignis EO (Emergency Out. Noto-Aus) tritt dann auf, wenn ein mit EO-Funktion projektierter Digital-Eingang aktiviert wird [mcfg / Kapitel 1.7.2.5]. Sofern mehrere EO-Eingänge im System vorhanden sind, kann mit dem Statusregister *axst* geprüft werden, welcher EO-Eingang den Fehler verursacht. Nachfolgend wird ein einfaches Beispielprogramm für die Implementierung eines EO-Handlers aufgelistet:

```
...
procedure EVEO;     // vordefinierter Name für
                    // Timeout-EVENT-Handler
begin
     CI0 := 999;    // Common Variable
                    // signalisiert Pogrammabbruch
     abort;         // Benutzerprogramm abbrechen
end;

...
begin
     ...
     CI0 := 0;      // Common Variable
                    // löschen
     enev(EVEO);    // Timeout-Handler freigeben
     ...
end.
```

6.4.3 Event-Prozedur EVDNR

Die Funktionsweise der Event-Prozedur EVDNR ist ebenfalls mit EVEO identisch, jedoch wird diese Prozedur beim Eintreffen des Ereignisses Drive Not Ready (Antrieb nicht bereit) automatisch abgearbeitet. Das Ereignis DNR tritt dann auf, wenn ein mit DR-Funktion projektierter Digital-Eingang inaktiv wird [mcfg / Kapitel 1.7.2.5].

6.4.4 Event-Prozedur EVLSH

Die Funktionsweise der Event-Prozedur EVLSH ist ebenfalls mit EVEO identisch, jedoch wird diese Prozedur beim Eintreffen des Ereignisses Limit Switch Hardware (Hardware-Endschalter) automatisch abgearbeitet. Das Ereignis LSH tritt dann auf, wenn ein mit LSL_SMD, LSL_TOM, LSL_SMA, LSR_SMD, LSR_TOM oder LSR_SMA-Funktion projektierter Digital-Eingang aktiviert wird [mcfg / Kapitel 1.7.2.5].

6.4.5 Event-Prozedur EVLSS

Die Funktionsweise der Event-Prozedur EVLSS ist ebenfalls mit EVEO identisch, jedoch wird diese Prozedur beim Eintreffen des Ereignisses Limit Switch Software (Software-Endlage) automatisch abgearbeitet. Das Ereignis LSS tritt dann auf, wenn die aktuelle Position eines Achssystems einen im TOOLSET-Programm *mcfg.exe* spezifizierten Grenzwert überschreitet und der entsprechende Grenzwert mit der Funktion TOM oder SMA projektiert wurde [mcfg / Kapitel 1.7.2.5].

6.4.6 Event-Prozedur EVMPE

Die Funktionsweise der Event-Prozedur EVMPE ist ebenfalls mit EVEO identisch, jedoch wird diese Prozedur beim Eintreffen des Ereignisses Maximum Position Error (Maximaler Schleppfehler) automatisch abgearbeitet. Das Ereignis MPE tritt dann auf, wenn der Regelkreis geschlossen ist und die Differenz von Soll- und Ist-Position eines Achssystems den im TOOLSET-Programm *mcfg.exe* spezifizierten Grenzwert überschreitet [mcfg / Kapitel 1.7.2.1.9].

6.4.7 Event-Prozedur EVUI

Die Funktionsweise der Event-Prozedur EVUI ist ebenfalls mit EVEO identisch, jedoch wird diese Prozedur beim Eintreffen des Ereignisses User Input automatisch abgearbeitet. Das Ereignis UI tritt dann auf, wenn ein mit UI projektierter Digital-Eingang aktiviert wird [mcfg / Kapitel 1.7.2.5]. Der Benutzer hat die Möglichkeit mit UI-projektierten Digital-Eingängen benutzerspezifische Spezial-Funktionen im SAP-Programm aufzubauen. Das alternative zyklische Abfragen (Polling) kann entfallen.

6.4.8 Priorität und Abarbeitungsreihenfolge der Event-Prozeduren

Es ist möglich, dass verschiedene Ereignisse zum gleichen Zeitpunkt eintreffen. In diesem Fall ist die Priorität wie folgt gegeben:

MCU-3000 / APCI-8001:MCU-3100 / APCI-8008: MCU-3400C / CPCI-8004:

Name der Prozedur	Priorität
EVEO	höchste Priorität
EVDNR	↓
EVLSH	
EVLSS	
EVMPE	
EVUI	niedrigste Priorität

MCU-6000 / APCI-8401:

Name der Prozedur	Priorität
EVTOASM	höchste Priorität
EVEO	
EVDNR	↓
EVLSH	
EVLSS	
EVMPE	
EVUI	niedrigste Priorität

Sofern momentan eine Event-Prozedur (Event 1) in Bearbeitung ist, wird das Eintreffen eines anderen Events (Event 2) mit niederer oder höherer Priorität ignoriert und erst nach Abarbeitung des momentanen Eventhandlers (Event 1) durchgeführt. Event 2 muss dann aber noch aktiv sein!
Anmerkung: Nach den *STOP* und *ABORT* SAP-Befehlen und während der Ausführung des *WT()* SAP-Befehls werden keine Event-Handler abgearbeitet.

6.5 SAP-Satz-Befehle

In nachfolgend aufgelisteter Befehls-Referenzliste sind eine Reihe von Befehlen enthalten, mit denen ein Programmaufbau mit Satzstruktur erzielt werden kann. Dies sind alle Befehle, deren Befehlsname mit dem Zeichen 'W' endet. Dazu gehören beispielsweise die SAP-Befehle *MLAW()*, *JAW()* oder *SSMSW().* Diese Befehle warten automatisch das Profilende aller beteiligten Achsen ab, d.h. die nächste Anweisung wird erst beim Erreichen der Zielpositionen der angewählten Achsen abgearbeitet. Dazu prüft die CNC-Task zyklisch die Profil-Ende-Flags dieser Achsen ab und setzt das Programm ggf. bei der nächsten Anweisung fort. Bei diesem Abprüfen werden auch die oben freigegebenen EVENT-Handler berücksichtigt und im Bedarfsfall automatisch abgearbeitet.
Anmerkung: Eine andere Möglichkeit der Profilende-Abprüfung ist das Auswerten des *axst*-Achsenqualifizierers.

6.6 SAP-Befehls-Referenzliste *rw_SymPas*

6.6.1 Aufbau der Referenzliste

Die Referenzliste ist wie folgt aufgebaut:

KURZWORTBEDEUTUNG, BESCHREIBUNG	Dies ist der Name, mit dessen Hilfe die nachfolgend beschriebene Funktion aufgerufen wird. Hier steht die ausführliche Beschreibung des entsprechenden Funktionsnamens.
FUNKTIONSPARAMETER:	Sofern die Funktion eine Parameterübergabe verlangt, werden diese hier aufgeführt.
SYSTEMPARAMETER:	Verschiedene Funktionen werden unter Berücksichtigung verschiedener Systemparameter ausgeführt. Diese werden hier aufgeführt.
SIMULTANFUNKTION:	Bei verschiedenen Funktion ist es gestattet, eine oder mehrere Achsen zu spezifizieren, für welche die entsprechende Funktion auszuführen ist.
REFERENZEN:	Verweise auf andere Funktionen und Kapitel.
DEKLARATION:	Die formale Deklaration von vordefinierten Systemfunktionen, benutzerdefinierte Elemente sind kursiv gesetzt.
ERGEBNISTYP:	Der Typ des zurückgelieferten Wertes (nur bei Systemfunktionen).
BESCHREIBUNG	Klartextbeschreibung des Befehls.
ANMERKUNG:	Immer wiederkehrende Anmerkungen und Erläuterungen verweisen hier auf die entsprechenden Kapitel.
BEISPIEL:	Ein Beispiel für die entsprechende Funktion.

6.6.2 ABORT, abort

BESCHREIBUNG:	Dieser Befehl bewirkt das Abbrechen eines laufenden SAP-Programmes. Im Gegensatz zur *STOP*-Anweisung kann das Programm nicht mit dem PCAP-Befehl *contcnct()* bzw. SAP-Befehl *CONTCNCT()* fortgesetzt werden. Dies ist nur durch den PCAP-Befehl *startcnct()* bzw. PCAP-Befehl *STARTCNCT()* möglich.
ANMERKUNG:	Nach Ausführen des Befehls werden die freigegebenen EVENT-Handler-Prozeduren nicht mehr abgearbeitet.
BEISPIEL:	*ABORT;*

6.6.3 ABS, absolute function

BESCHREIBUNG:	Die Funktion liefert den absoluten Wert von *value* zurück.
DEKLARATION:	abs(*value*:double)
ERGEBNISTYP:	double
BEISPIEL:	*...* *var* *d1, d2: double;* *...* *d1 := -5.0;* *d2 := ABS(d1);* *// d2 := 5.0*

6.6.4 ACOS, arc cosine function

BESCHREIBUNG:	Die Funktion liefert den Arcuscosinus von *value* zurück. Das Argument *Value* muss im Bereich [-1..+1] liegen. Der Rückgabewert hat die Einheit Rad und liegt in den Grenzen [0..pi].
DEKLARATION:	acos(*value*:double)
ERGEBNISTYP:	double

6.6.5 ASIN, arc sine function

BESCHREIBUNG:	Die Funktion liefert den Arcussinus von *value* zurück. Das Argument *Value* muss im Bereich [-1..+1] liegen. Der Rückgabewert hat die Einheit Rad und liegt in den Grenzen [-pi/2..+pi/2].
DEKLARATION:	asin(*value*:double)
ERGEBNISTYP:	double

6.6.6 ATAN, arc tangent function

BESCHREIBUNG:	Die Funktion liefert den Arcustangens von *value* zurück. Der Rückgabewert hat die Einheit Rad und liegt in den Grenzen [-pi/2..+pi/2].
DEKLARATION:	atan(*value*:double)
ERGEBNISTYP:	double

6.6.7 AZO, activate zero offsets

BESCHREIBUNG:	PCAP-Befehl *azo()*
FUNKTIONSPARAMETER:	Integer-Konstante im Wertebereich 0..4
BEISPIEL:	*const Offsets1 = 1;* *azo(Offsets1); // Nullpunktverschiebungen Satz 1 aktivieren*

6.6.8 CL, close loop

BESCHREIBUNG:	PCAP-Befehl *cl()* [Kapitel 4.4.6]
FUNKTIONSPARAMETER:	*Spec*
SIMULTANFUNKTION:	ja
BEISPIEL:	*CL(A1, A2); // Achskanäle 1 und 2 in Lageregelung bringen*

6.6.9 CLV

BESCHREIBUNG:	PCAP-Befehl *clv()* [Kapitel 4.4.9]
FUNKTIONSPARAMETER:	*Spec*
SIMULTANFUNKTION:	ja
BEISPIEL:	*clv (A1, A2); // Achskanäle 1 und 2 in Lageregelung bringen* *js (A1, A2); // danach die Achsen sofort stoppen*

6.6.10 CONTCNCT, continue CNC-Task

BESCHREIBUNG:	Dieser Befehl setzt die im Parameter übergebene CNC-Task fort.
FUNKTIONSPARAMETER:	Integer-Konstante im Bereich 0..3
ANMERKUNG:	Der Befehl kann benutzt werden, um ein gestopptes SAP-Programm fortzusetzen. Ein SAP-Programm welches mit dem SAP-Befehl *ABORT* angehalten wurde, kann nur mit dem SAP-Befehl *STARTCNCT()* oder PCAP-Befehl *startcnct()* neu gestartet, also nicht fortgesetzt, werden. Das selbsttätige Fortsetzen einer gestoppten Task ist ebenfalls nicht möglich.
BEISPIEL:	*...* *const* *TASK0 = 0;* *...* *CONTCNCT(TASK0); // Task 0 fortsetzen* *CONTCNCT(1); // Task 1 fortsetzen*

6.6.11 COS, cosine function

BESCHREIBUNG:	Die Funktion liefert den Cosinus von *value* zurück. Das Argument *Value* wird als Winkel in der Einheit Rad (0..2Pi = 0..360 Grad) interpretiert.
DEKLARATION:	cos(*value*:double)
ERGEBNISTYP:	double
ANMERKUNG:	*Sin()*, *Tan()*-Funktion
BEISPIEL:	*...* *var* *d1, d2: double;* *...* *d1 := 3.1415;* *d2 := COS(d1); // d2 := -1.0 (gerundet)*

6.6.12 COSH, hyperbolic cosine function

BESCHREIBUNG:	Die Funktion liefert den hyperbolischen Cosinus von *value* zurück.
DEKLARATION:	cos(*value*:double)
ERGEBNISTYP:	double

6.6.13 DISEV, disable event

BESCHREIBUNG:	sperrt den spezifizierten Event-Handler
FUNKTIONSPARAMETER:	*Event*
REFERENZEN:	Kapitel 6.4. und SAP-Befehl *ENEV()*
BEISPIEL:	*DISEV(EVEO); // Emergency Out Handler ignorieren*

6.6.14 ENEV, enable event

BESCHREIBUNG:	gibt den spezifizierten Event-Handler frei
FUNKTIONSPARAMETER:	*Event*
REFERENZEN:	Kapitel 6.4. und SAP-Befehl *DISEV()*
BEISPIEL:	*ENEV(EVEO); // Emergency Out Handler freigeben*
ANMERKUNG:	Der freigegebene Event-Handler ist nicht mehr aktiv, wenn die Task beendet ist bzw. angehalten wurde.

6.6.15 EXP, exponential function

BESCHREIBUNG:	Die Funktion liefert den Wert e^{value} zurück, wobei *e* die Basis des natürlichen Logarithmus ist (2.718281...).
DEKLARATION:	exp(*value*:double)
ERGEBNISTYP:	double
ANMERKUNG:	Funktion *Ln()*

6.6.16 JA, jog absolute

BESCHREIBUNG:	Der oder die ausgewählten Achskanäle werden auf die angegebenen Positionssollwerte absolut verfahren. Dazu wird der Motor mit der achsspezifischen Beschleunigung *jac* auf die Geschwindigkeit *jvl* hochbeschleunigt und auf die spezifizierte Zielposition *Pos* verfahren. Zusätzlich kann mit dem Parameter *jtl* eine Zielgeschwindigkeit spezifiziert werden. Die Angabe der Bahnparameter erfolgt in den achsenspezifischen Einheiten.
FUNKTIONSPARAMETER:	*Spec* und *Pos*
SYSTEMPARAMETER:	Qualifizierer: *jac, jvl* und *jtvl*
SIMULTANFUNKTION:	ja
REFERENZEN:	PCAP-Befehl *ja()*, SAP-Befehl *JAW()*
ANMERKUNG:	PCAP-Befehl *ja()*
BEISPIEL:	*JA(A1:=100.0); // Achse 1 absolut auf Position 100 verfahren* *JA(A1:=100.0, A2:=100.0);*

6.6.17 JAW, jog absolute waiting

BESCHREIBUNG:	Dieser Befehl ist mit dem SAP-Befehl *JA()* und PCAP-Befehl *ja()* identisch, wartet jedoch zusätzlich das Profilende aller beteiligten Achsen ab. Durch den Einsatz dieses Befehls erhält das SAP-Programm eine satzähnliche Gestalt wie sie bei den handelsüblichen CNC-Steuerungen zu finden ist.
FUNKTIONSPARAMETER:	*Spec, Pos*
SYSTEMPARAMETER:	Qualifizierer: *jac, jvl* und *jtvl*
SIMULTANFUNKTION:	ja
REFERENZEN:	*JA*
ANMERKUNG:	Der Benutzer sollte durch den Einsatz von EVENT-Handlern sicherstellen, dass der Antrieb auch in Ausnahmesituationen korrekt bedient wird, da das CNC-Programm gerade bei Positioniervorgängen mit großem Zeitbedarf entsprechend lange bei diesem Befehl verweilt.
BEISPIEL:	*JAW(A2:=-1000.0); // Achse 1 absolut auf Position -1000.0 verfahren* *// und warten bis das Profilende erreicht wird* *JAW(A1:=1e3, A2 := 1.3e4);*

6.6.18 JHI, jog home index

BESCHREIBUNG:	Der Referenzsuchlauf auf die Nullspur (Index) des Drehgebers oder des Linearmaßstabes aller selektierten Achskanäle wird gestartet. Der Suchlauf wird abgebrochen, sobald der in *Pos* spezifizierte Verfahrweg bzw. Winkel überschritten wird.
FUNKTIONSPARAMETER:	*Spec, Pos*
SYSTEMPARAMETER:	Qualifizierer: *hac* und *hvl*
SIMULTANFUNKTION:	ja
REFERENZEN:	PCAP-Befehl *jhi()*, SAP-Befehl *JHIW()*
BEISPIEL:	*JHI(A1 := 1.0, A2 := 1.5); // Referenzsuchlauf von Achse 1 und 2* *// starten.*

6.6.19 JHIW, jog home index waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *jhi()* und SAP-Befehl *JHI()*. Zusätzlich wird das Profilende der beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	Qualifizierer: *hac* und *hvl*
SIMULTANFUNKTION:	ja
ANMERKUNG:	SAP-Befehl *JA()*
BEISPIEL:	*JHIW(A1 := 5.0);*

6.6.20 JHL, jog home left

BESCHREIBUNG:	Der Referenzsuchlauf auf einen mit REF projektierten Digitaleingang aller selektierten Achskanäle wird in die linke Verfahrrichtung gestartet.
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	Qualifizierer: *hac* und *hvl*
SIMULTANFUNKTION:	ja
REFERENZEN:	PCAP-Befehl *jhl()*, SAP-Befehl *JHLW()*
BEISPIEL:	*JHL(A1);*

6.6.21 JHLW, jog home left waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *jhl()* und SAP-Befehl *JHL()*. Zusätzlich wird das Profilende der beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	Qualifizierer: *hac* und *hvl*
SIMULTANFUNKTION:	ja
REFERENZEN:	PCAP-Befehl *jhl()*, SAP-Befehl *JHL()*
ANMERKUNG:	SAP-Befehl *JA()*
BEISPIEL:	*JHLW(A2);*

6.6.22 JHR, jog home right

BESCHREIBUNG:	Der Referenzsuchlauf auf einen mit REF projektierten Digitaleingang aller selektierten Achskanäle wird in die rechte Verfahrrichtung gestartet.
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	Qualifizierer: *hac* und *hvl*
SIMULTANFUNKTION:	ja
REFERENZEN:	PCAP-Befehl *jhr()*, SAP-Befehl *JHRW()*
BEISPIEL:	*JHR(A2);*

6.6.23 JHRW, jog home right waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *jhr()* und SAP-Befehl *JHR()*. Zusätzlich wird das Profilende der beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	Qualifizierer: *hac* und *hvl*
SIMULTANFUNKTION:	ja
ANMERKUNG:	SAP-Befehl *JA()*
BEISPIEL:	*JHRW(A1);*

6.6.24 JR, jog relative

BESCHREIBUNG:	Die Beschreibung erfolgt beim PCAP-Befehl *jr()*.
FUNKTIONSPARAMETER:	*Spec, Pos*
SYSTEMPARAMETER:	Qualifizierer: *jac, jvl* und *jtvl*
SIMULTANFUNKTION:	ja
BEISPIEL:	*JR(A1 := 100);*

6.6.25 JRW, jog relative waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *jr()* und SAP-Befehl *JR()*. Zusätzlich wird das Profilende der beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec, Pos*
SYSTEMPARAMETER:	Qualifizierer: *jac, jvl* und *jtvl*
SIMULTANFUNKTION:	ja
REFERENZEN:	JR

6.6.26 JS, jog stop

BESCHREIBUNG:	Die Beschreibung erfolgt beim PCAP-Befehl *js()*.
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	Qualifizierer: *sdec*
SIMULTANFUNKTION:	ja
BEISPIEL:	*JS(A1);*

6.6.27 JSW, jog stop waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *js()* und SAP-Befehl *JS()*. Zusätzlich wird das Profilende der beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	Qualifizierer: *sdec*
SIMULTANFUNKTION:	ja
BEISPIEL:	*JSW(A1);*

6.6.28 LN, natural logarithm function

BESCHREIBUNG:	Die Funktion liefert den natürlichen Logarithmus von *value* zurück, d.h. der Wert, mit dem die Konstante 2.71828... potenziert werden muss, um *value* zu erhalten.
DEKLARATION:	ln(*value*:double)
ERGEBNISTYP:	double
ANMERKUNG:	Wert kleiner/gleich 0.0 für *value* sind mathematisch nicht definiert. Die Funktion hat in diesem Fall keinen gültigen Rückgabewert. Funktion *Exp()*

6.6.29 LPR, latch position registers

BESCHREIBUNG:	Daten-Aufzeichnung eines Bewegungsvorganges für eine Achse starten (siehe grafische Systemanalyse in mcfg).
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	PU, TU, LST, LET
SIMULTANFUNKTION:	nein
BEISPIEL:	LPR (A1);

6.6.30 LPRS, latch position registers synchronous

BESCHREIBUNG:	Synchrone Daten-Aufzeichnung eines Bewegungsvorganges für mehrere Achsen starten (siehe grafische Systemanalyse in mcfg).
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	PU, TU, LST, LET
SIMULTANFUNKTION:	ja
BEISPIEL:	LPRS (A1, A2, A3);

6.6.31 MCA, move circular absolute
SMCA, spool motion circular absolute

BESCHREIBUNG:	PCAP-Befehl *mca()*, *smca()*
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI*
BEISPIEL:	*MCA(A1 := 50.0, A2 := 0.0, PHI := 720.0);* *SMCA(A1 := 0.0, A2 := 10.0, PHI := 0.1);*

6.6.32 MCAW, move circular absolute waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem SAP-Befehl *MCA()*, jedoch wird hier zusätzlich das Profilende der beiden beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI*
REFERENZEN:	PCAP-Befehl *mca()*

6.6.33 MCA3D, move circular absolute three-dimendional
SMCA3D, spool move circular absolute three-dimendional

BESCHREIBUNG:	PCAP-Befehl *mca3d()*, *smca3d()*
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI, PN1, PN2, PN3*
BEISPIEL:	*MCA3D(A1 := 50.0, A2 := 0.0, A3 := 0.0, PN1 = 1.0, PN2 =0. 0, PN3 = 1.0, PHI := 720.0); // Kreis 45 Grad um A2 gedreht*

6.6.34 MCA3DW, move circular absolute three-dimendional waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem SAP-Befehl *MCA3D()*, jedoch wird hier zusätzlich das Profilende der beiden beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI, PN1, PN2, PN3*
REFERENZEN:	SAP-Befehl *mca3d()*

6.6.35 MCR3D, move circular relative three-dimendional
SMCR3D, spool move circular relative three-dimendional

BESCHREIBUNG:	PCAP-Befehl *mcr3d()*, *smcr3d()*
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI, PN1, PN2, PN3*
BEISPIEL:	*MCR3D(A1 := 50.0, A2 := 0.0, A3 := 0.0, PN1 = 1.0, PN2 =0. 0, PN3 = 1.0, PHI := 720.0); // Kreis 45 Grad um A2 gedreht*

6.6.36 MCR3DW, move circular relative three-dimendional waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem SAP-Befehl *MCR3D()*, jedoch wird hier zusätzlich das Profilende der beiden beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI, PN1, PN2, PN3*
REFERENZEN:	SAP-Befehl *mcr3d()*

6.6.37 MCR, move circular relative
SMCR, spool motion circular relative

BESCHREIBUNG:	PCAP-Befehl *mcr()*, *smcr()*
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI*
BEISPIEL:	*MCR(A1 := 50.0, A2 := 0.0, PHI := 360.0);* *SMCR(A1 := 0.0, A2 := 10.0, PHI := 45.0);*

6.6.38 MCRW, move circular relative waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem SAP-Befehl *MCR()*, jedoch wird hier zusätzlich das Profilende der beiden beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI*

6.6.39 MHA, move helical absolute
SMHA, spool motion helical absolute

BESCHREIBUNG:	PCAP-Befehl *mha()*, *smha()* *Falls der Kreis hier per Zielpunktsangabe definiert werden soll, müssen die Zielkoordinaten den* Achsenspezifizierern*, die Mittelpunktskoordinaten den Systemparametern DTCA1 und DTCA2 zugewiesen werden. Bei Kreisangabe per Verfahrwinkel werden den* Achsenspezifizierern der Kreisachsen die Mittelpunktskoordinaten zugewiesen.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI, (ggf. DTCA1, DTCA2)*
BEISPIELE:	*MHA(A1 := 50.0, A2 := 0.0, PHI := 720.0, A3 := 10);* *SMHA(A1 := 0.0, A2 := 10.0, PHI := 0.1, A3 := 10);* *// Vollkreis im Gegenuhrzeigersinn mit Radius 10* *MHR(A1 := 0.0, A2 := 0.0, PHI := 0.0, A3 := 10, DTCA1 :=-10, DTCA2 := 0);* *// Halbkreis im Uhrzeigersinn mit Radius 10* *SMHR(A1 := 20.0, A2 := 0.0, PHI := -1e-100, A3 := 10, DTCA1 := 10, DTCA2 := 0);*

6.6.40 MHAW, move helical absolute waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem SAP-Befehl *MHA()*, jedoch wird hier zusätzlich das Profilende aller beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI, (ggf. DTCA1, DTCA2)*

6.6.41 MHR, move helical relative
SMHR, spool motion helical relative

BESCHREIBUNG:	PCAP-Befehl *mhr()*, *smhr()* *Eine Zielpunktsangabe für den Kreis ist hier nicht möglich.*
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI, (ggf. DTCA1, DTCA2)*

6.6.42 MHRW, move helical relative waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem SAP-Befehl *MHR()*, jedoch wird hier zusätzlich das Profilende aller beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL, PHI, (ggf. DTCA1, DTCA2)*

6.6.43 MLA, move linear absolute
SMLA, spool motion linear absolute

BESCHREIBUNG:	Die Beschreibung erfolgt bei dem PCAP-Befehl *mla()* bzw. *smla()*.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL*
SIMULTANFUNKTION:	ja
BEISPIEL:	*MLA(A1:=1000.0, A2:=3.2e2);* *SMLA(A1:=100.0, A2:=-335.0);*

6.6.44 MLAW, move linear absolute waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem SAP-Befehl *MLA()*, jedoch wird hier zusätzlich das Profilende der beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL*
SIMULTANFUNKTION:	ja
BEISPIEL:	*MLAW(A1:=-0.3e3, A2:=100.4);*

6.6.45 MLR, move linear relative
SMLR, spool motion linear relative

BESCHREIBUNG:	Die Beschreibung erfolgt bei dem PCAP-Befehl *mlr()* bzw. *smlr()*.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL*
SIMULTANFUNKTION:	ja
BEISPIEL:	*MLR(A1:=2000.0, A2:=3.2e2);* *SMLR(A1:=300.0, A2:=-35.3);*

6.6.46 MLRW, move linear relative waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem SAP-Befehl *MLRW()*, jedoch wird hier zusätzlich das Profilende der beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*, *Pos*
SYSTEMPARAMETER:	*TRAC, TRVL, TRTVL*
SIMULTANFUNKTION:	ja
BEISPIEL:	*MLRW(A1:=-3.45e3, A2:=100.4e-1);*

6.6.47 MS, motion stop

BESCHREIBUNG:	Die Beschreibung erfolgt beim PCAP-Befehl *ms()* [Kapitel 4.4.39].
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	keine
SIMULTANFUNKTION:	ja
BEISPIEL:	*MS(A1, A2);*

6.6.48 MSW, motion stop waiting

BESCHREIBUNG:	Dieser Befehl ist identisch mit dem PCAP-Befehl *ms()* und SAP-Befehl *MS()*, jedoch wird hier zusätzlich das Profilende der beteiligten Achsen abgewartet.
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	keine
SIMULTANFUNKTION:	ja
BEISPIEL:	*MSW(A1, A2);*

6.6.49 OL, open loop

BESCHREIBUNG:	PCAP-Befehl *ol()* [Kapitel 4.4.41]
FUNKTIONSPARAMETER:	*Spec*
SIMULTANFUNKTION:	ja
BEISPIEL:	*OL(A1, A2); // Lageregelkreis von A1 und A2 öffnen*

6.6.50 POWER

BESCHREIBUNG:	Die Funktion liefert den Wert von *base* hoch *exponent* zurück.
DEKLARATION:	sqrt(*base, exponent* : double)
ERGEBNISTYP:	double
ANMERKUNG:	Funktion ist verfügbar ab RWMOS.ELF V2.5.3.93
BEISPIEL:	*...* *var* *d1, d2: double;* *...* d1 := 2.0; d2 := 3.0; d2 := POWER(d1, d2); // d2 := 8.0

6.6.51 RA, reset axis

BESCHREIBUNG:	PCAP-Befehl *ra()*
FUNKTIONSPARAMETER:	*Spec*
SIMULTANFUNKTION:	ja
BEISPIEL:	*RA(A1, A2); // Achsen A1 und A2 zurücksetzen*

6.6.52 RAMS, reset ASM-2003 status register

Funktionsbeschreibung gilt nur für MCU-6000 / APCI-8401	
BESCHREIBUNG:	Die Beschreibung erfolgt beim PCAP-Befehl *rams()* [Kapitel 4.4.43].
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	keine
BEISPIEL:	*RASM(A1, A2);* *Dieser Befehl kann nur beim System MCU-6000 verwendet werden.*

6.6.53 RDCBD, read COMMON BUFFER double function

BESCHREIBUNG:	Die Funktion liefert einen Gleitpunktwert mit doppelter Genauigkeit (double) aus dem CNC-taskspezifischen COMMON BUFFER zurück. Der Parameter *offset* ist ein Byte-Offset bezogen auf das erste Element (Element 0) des COMMON BUFFER. Der Double-Datentyp belegt 8 Bytes im COMMON BUFFER. Um einen korrekten Zugriff durch das MCU-G3-CPU-System zu ermöglichen, muss *offset* immer doppelwortgerichtet sein, d.h. einen durch 8 teilbaren Wert haben.
DEKLARATION:	RDCBD(*offset*:integer)
ERGEBNISTYP:	double
ANMERKUNG:	Die CNC-Task-spezifische Buffergröße beträgt 1000 Bytes. PCAP-Befehle *rdcbcnct() und wrcbcnct()*, SAP-Befehle *RDCBx()* und *WRCBx()*
BEISPIEL:	*...* *var* *cbd: double;* *...* *cbd := RDCBD(500); // Double-Variable einlesen ab offset 500*

6.6.54 RDCBI, read COMMON BUFFER integer function

BESCHREIBUNG:	Die Funktion liefert einen Integerwert aus dem CNC-taskspezifischen COMMON BUFFER zurück. Der Parameter *offset* ist ein Byte-Offset bezogen auf das erste Element (Element 0) des COMMON BUFFER. Der Integer-Datentyp belegt 4 Bytes im COMMON BUFFER. Um einen korrekten Zugriff durch das MCU-G3-CPU-System zu ermöglichen, muss *offset* immer wortgerichtet sein, d.h. einen durch 4 teilbaren Wert haben.
DEKLARATION:	RDCBI(*offset*:integer)
ERGEBNISTYP:	integer
ANMERKUNG:	Die CNC-Task-spezifische Buffergröße beträgt 1000 Bytes. PCAP-Befehle *rdcbcnct() und wrcbcnct()*, SAP-Befehle *RDCBx()* und *WRCBx().*
BEISPIEL:	*...* *var* *cbi: integer;* *...* *cbi := RDCBI(500); // Integer-Variable einlesen ab offset 500*

6.6.55 RDCBS, read COMMON BUFFER single function

BESCHREIBUNG:	Die Funktion liefert einen Gleitpunktwert mit einfacher Genauigkeit (single) aus dem CNC-taskspezifischen COMMON BUFFER zurück. Der Parameter *offset* ist ein Byte-Offset bezogen auf das erste Element (Element 0) des COMMON BUFFER. Der Single-Datentyp belegt 4 Bytes im COMMON BUFFER. Um einen korrekten Zugriff durch das MCU-G3-CPU-System zu ermöglichen, muss *offset* immer wortgerichtet sein, d.h. einen durch 4 teilbaren Wert haben.
DEKLARATION:	RDCBS(*offset*:integer)
ERGEBNISTYP:	single
ANMERKUNG:	Die CNC-Task-spezifische Buffergröße beträgt 1000 Bytes. PCAP-Befehle *rdcbcnct() und wrcbcnct()*, SAP-Befehle *RDCBx()* und *WRCBx()*
BEISPIEL:	*...* *var* *cbs: single;* *...* *cbs := RDCBS(500); // Single-Variable einlesen ab offset 500*

6.6.56 RPTODP, Real-Position to Desired-Position

BESCHREIBUNG:	PCAP-Befehl *RPtoDP()*
ANMERKUNG:	Die betreffenden Achsen dürfen sich nicht in einem Verfahrprofil befinden, d.h. das Profilendeflag im Achsenstatusregister muss gesetzt sein.
BEISPIEL:	*RPTODP (X, Z);*

6.6.57 RS, reset system

BESCHREIBUNG:	PCAP-Befehl *rs()*
ANMERKUNG:	Sofern dieser Befehl ausgeführt wird, ist keine Kontrolle durch das Standalone-Applikations-Programm mehr möglich, da die CNC-Task durch diesen Befehl angehalten wird.
BEISPIEL:	*RS*; *// komplettes Achssytem zurücksetzen*

6.6.58 SHP, set home position

BESCHREIBUNG:	PCAP-Befehl *shp()*
FUNKTIONSPARAMETER:	*Spec, Pos*
SIMULTANFUNKTION:	ja
BEISPIEL:	*SHP(A2:=1000.0);*

6.6.59 SIN, sine function

BESCHREIBUNG:	Die Funktion liefert den Sinus von *value* zurück. Das Argument *Value* wird als Winkel in der Einheit Rad (0..2Pi = 0..360 Grad) interpretiert.
DEKLARATION:	sin(*value*:double)
ERGEBNISTYP:	double
ANMERKUNG:	*Cos()*, *Tan()*-Funktion
BEISPIEL:	*...* *var* *d1, d2: double;* *...* *d1 := 3.1415;* *d2 := SIN(d1);* *// d2 := 0.0 (gerundet)*

6.6.60 SINH, hyperbolic sine function

BESCHREIBUNG:	Die Funktion liefert den hyperbolischen Sinus von *value* zurück.
DEKLARATION:	cos(*value*:double)
ERGEBNISTYP:	double

6.6.61 SQR, square function

BESCHREIBUNG:	Die Funktion liefert das Quadrat (»square«) von value zurück.
DEKLARATION:	sqr(*value*:double)
ERGEBNISTYP:	double
ANMERKUNG:	Verfügbar nur in RWMOS und Compilerversionen ab 05.10.2007
BEISPIEL:	*...* *var* *d1, d2: double;* *...* *d1* := 9.0; d2 := SQR(d1); // d2 := 81.0

6.6.62 SQRT, square root function

BESCHREIBUNG:	Die Funktion liefert die Quadratwurzel (»square root«) von value zurück.
DEKLARATION:	sqrt(*value*:double)
ERGEBNISTYP:	double
ANMERKUNG:	Negative Werte von *value* sind mathematisch nicht definiert. Die Funktion hat in diesem Fall keinen gültigen Rückgabewert.
BEISPIEL:	*...* *var* *d1, d2: double;* *...* *d1* := 9.0; d2 := SQRT(d1); // d2 := 3.0

6.6.63 SSF, Spool-Special-Function

BESCHREIBUNG:	Dieses Kommando ermöglicht dem Anwender auch andere Kommandos als Verfahrbefehle im Spooler einzutragen. In der Systemvariable *SSFP* wird das auszuführende Kommando eingetragen. Der Wert *Value* wird als Parameter bei der jeweiligen Achse eingetragen. Die verfügbaren Kommandos sind beim PCAP-Kommando ssf aufgeführt.
FUNKTIONSPARAMETER:	*Spec*, *Value*
SYSTEMPARAMETER:	*SSFP*
SIMULTANFUNKTION:	ja
KOMMANDOS:	siehe PCAP-Kommando ssf Kapitel 4.4.130.
BEISPIEL:	SSF(A1:=999, SSFP = 1); // CI1 mit 999 beschreiben SSF(A1:=1, A4:=2, SSFP := 1001); // O1 bei Achse 1 und O2 bei Achse 4 // setzen SSF(A1:=0, A2:=0, A3:=0, SSFP:=1000); // Spoolerhalt bei A1, A2 und A3

6.6.64 SSMS, start spooled motions synchronous

BESCHREIBUNG:	Mit Hilfe von *spool*-Befehlen können Kommandos an die einzelnen Achskanäle der MCU-G3 übertragen werden. Diese werden in einer Warteschlange eingetragen. Der Befehl *SSMS()* veranlaßt den Synchronstart für die Spoolerbefehlsabarbeitung aller in *AS* spezifizierten Achsen.
FUNKTIONSPARAMETER:	*Spec*
SIMULTANFUNKTION:	ja
REFERENZEN:	PCAP-Befehl *ssms()*, SAP-Befehl *SSMSW()*
BEISPIEL:	*...* *SMLA(A1:=1000.0, A2:=1000.0); // Verfahrbefehl spoolen* *SMLR(A1:=200.0, A2:=500.0); // Verfahrbefehl spoolen* *...* *SSMS(A1, A2)*; *// Spooler starten*

6.6.65 SSMSW, start spooled motions synchronous waiting

BESCHREIBUNG:	Synchronstart aller angewählten Achsen und abwarten, bis sämtliche gespoolten Bewegungsprofile dieser Achsen komplett abgefahren sind und das Profilende aller beteiligten Achsen erreicht wird.
FUNKTIONSPARAMETER:	*Spec*
SIMULTANFUNKTION:	ja
REFERENZEN:	SAP-Befehl *SSMS()*
ANMERKUNG:	SPOOL-Modus
BEISPIEL:	*...* *SMLR(A1:=1000.0, A2:=1000.0); // Verfahrbefehl spoolen* *SMLR(A1:=200.0; A2:=500.0); // Verfahrbefehl spoolen* *...* *SSMSW(A1, A2)*; *// Spooler starten*

6.6.66 STARTCNCT, start CNC-Task

BESCHREIBUNG:	Dieser Befehl startet die im Parameter übergebene CNC-Task und führt das dort abgelegte SAP-Programm vom Programmbeginn an aus.
FUNKTIONSPARAMETER:	Integer-Konstante im Bereich 0..3
ANMERKUNG:	Ein SAP-Programm kann sich mit diesem Befehl auch selbsttätig vom Programmbeginn an starten.
BEISPIEL:	*...* *const* *Task1 = 1;* *...* *STARTCNCT(Task1);*

6.6.67 STOP, stop

BESCHREIBUNG:	Dieser Befehl bewirkt den Programmstop des momentan ablaufenden Standalone-Applikations-Programms. Zusätzlich wird die entsprechende CNC-Task (Task 0, 1, 2, oder 3) in den Idle-Zustand gebracht. Das Applikations-Programm kann mit Hilfe des *contcnct()*-PCAP-Befehls, des CONTCNCT()-SAP-Befehls oder im TOOLSET-Programm *mcfg.exe* wieder fortgesetzt werden.
ANMERKUNG:	Eventuell freigegebene EVENT-Handling-Prozeduren werden nach Ausführen des Stop-Befehls nicht mehr abgearbeitet. Der Antrieb sollte vor Ausführung dieses Befehls deshalb in einen sicheren Betriebszustand gebracht werden.
BEISPIEL:	*STOP; // Stoppt das SAP-Programm*

6.6.68 STEPCNCT, step CNC-Task

BESCHREIBUNG:	Dieser Befehl führt eine Programmzeile aus in der angegebenen CNC-Task.
FUNKTIONSPARAMETER:	Integer-Konstante im Bereich 0..3
ANMERKUNG:	Eventuell freigegebene EVENT-Handling-Prozeduren werden nach Ausführung der Programmzeile in der entsprechend selektierten Task nicht mehr abgearbeitet. Vor Ausführung dieses Befehls muss ein gültiges Programm geladen sein. Siehe dazu auch PCAP Kommando *stepcnct* (Kapitel 4.4.133).
BEISPIEL:	*...* *const* *Task3 = 3;* *...* *STEPCNCT(Task3);*

6.6.69 STOPCNCT, stop CNC-Task

BESCHREIBUNG:	Dieser Befehl hält die im Parameter übergebene CNC-Task an und damit auch das dort abgelegte SAP-Programm.
FUNKTIONSPARAMETER:	Integer-Konstante im Bereich 0..3
ANMERKUNG:	Eventuell freigegebene EVENT-Handling-Prozeduren werden nach Ausführung des *STOPCNCT()* von der entsprechend selektierten Task nicht mehr abgearbeitet. Siehe dazu auch Kapitel 6.6.68.
BEISPIEL:	*...* *const* *Task3 = 3;* *...* *STOPCNCT(Task3);*

6.6.70 STOPTOSS

BESCHREIBUNG:	Dieser Befehl versetzt die im Parameter übergebene CNC-Task vom Stop-Zustand in den Step-Zustand, ohne jedoch eine Programmzeile in der angegebenen Task auszuführen.
FUNKTIONSPARAMETER:	Integer-Konstante im Bereich 0..3
ANMERKUNG:	Wenn sich die angegebene Task nicht im Stop-Modus befindet hat das Kommando keinen Einfluss. Dieses Kommando wird vorzugsweise zur Single-Step Abarbeitung unter Verwendung mehrerer SAP Programmtasks benötigt. Eventuell zuvor gespoolte Verfahrkommandos werden bei den aktuell verwendeten Achsen ausgeführt.
BEISPIEL:	*...* *const* *Task3 = 3;* *...* *STOPTOSS(Task3);*

6.6.71 SZPA – Set Zero Position Absolut

BESCHREIBUNG:	Setzen eines absoluten virtuellen Nullpunktes. Dieses Kommando ist beim PCAP-Kommando szpa dokumentiert (Kapitel 4.4.135). Die Verwendung von SZPA bei Schrittmotorsystemen ist erst ab RWMOS.ELF Version 2.5.2.32 möglich.
FUNKTIONSPARAMETER:	*Spec, Pos*
SIMULTANFUNKTION:	ja
REFERENZEN:	PCAP-Befehle *szpa()*, szpr(), SAP-Befehl *SZPR*
BEISPIEL:	SZPA (X := 100, Y := -20);

6.6.72 SZPR – Set Zero Position Relativ

BESCHREIBUNG:	Relatives Verschieben des virtuellen Nullpunktes. Dieses Kommando ist beim PCAP-Kommando szpr dokumentiert (Kapitel 4.4.136). Die Verwendung von SZPR bei Schrittmotorsystemen ist erst ab RWMOS.ELF Version 2.5.2.32 möglich.
FUNKTIONSPARAMETER:	*Spec, Pos*
SIMULTANFUNKTION:	ja
REFERENZEN:	PCAP-Befehle *szpa()*, szpr(), SAP-Befehl *SZPA*
BEISPIEL:	SZPR (X := 100, Y := -20);

6.6.73 TAN, tangent function

BESCHREIBUNG:	Die Funktion liefert den Tangens von *value* zurück. Das Argument *Value* wird als Winkel in der Einheit Rad (0..2Pi = 0..360) Grad interpretiert.
DEKLARATION:	tan(*value*:double)
ERGEBNISTYP:	double
ANMERKUNG:	*Sin()*, *Cos()*-Funktion
BEISPIEL:	*...* *var* *d1, d2: double;* *...* *d1 := 0.5;* *d2 := TAN(d1); // d2 := 0.5463 (gerundet)*

6.6.74 TANH, hyperbolic tangent function

BESCHREIBUNG:	Die Funktion liefert den hyperbolischen Tangens von *value* zurück.
DEKLARATION:	tan(*value*:double)
ERGEBNISTYP:	double

6.6.75 UF, update filter

BESCHREIBUNG:	PCAP-Befehl *uf()*
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	Qualifizierer: *kp, ki, kd, kpl, kfca, kfcv*
SIMULTANFUNKTION:	ja
ANMERKUNG:	Zur Aktualisierung der PIDF-Filter-Koeffizienten müssen alle oben aufgeführten Qualifizierer vor Ausführung des Befehls initialisiert werden.
BEISPIEL:	*...* *A1.kp := 5.0; // Proportionalverstärkung ändern* *A1.ki := 0.0;* *A1.kd := 0.0;* *A1.kpl := 0.0;* *A1.kfca := 0.0;* *A1.kfcv := 0.0;* *UF(A1);* *...*

6.6.76 UTROVR, update trajectory override

BESCHREIBUNG:	PCAP-Befehl *utrovr()*
FUNKTIONSPARAMETER:	*Spec*
SYSTEMPARAMETER:	*TROVR*
SIMULTANFUNKTION:	ja
BEISPIEL:	*...* *TROVR := 0.9; // Bahngeschwindigkeitsoverride = -10%* *UTROVR(A1, A2); // reduzierte Bahngeschwindigkeit für Achsen A1 und A2* *...*

6.6.77 WRCBI, write COMMON BUFFER integer procedure

BESCHREIBUNG:	Die Prozedur beschreibt eine Speicherzelle vom Typ integer mit dem Wert *value* im CNC-taskspezifischen COMMON BUFFER. Der Parameter *offset* ist ein Byte-Offset bezogen auf das erste Element (Element 0) des COMMON BUFFER. Der Integer-Datentyp belegt 4 Bytes im COMMON BUFFER. Um einen korrekten Zugriff durch das MCU-G3-CPU-System zu ermöglichen, muss *offset* immer wortgerichtet sein, d.h. einen durch 4 teilbaren Wert haben.
DEKLARATION:	WRCBI(*offset*:integer; *value*:integer)
ANMERKUNG:	Die CNC-Task-spezifische Buffergröße beträgt 1000 Bytes. PCAP-Befehle *rdcbcnct() und wrcbcnct()*, SAP-Befehle *RDCBx()* und *WRCBx()*
BEISPIEL:	*WRCBI(500, -1000); // Integer-Variable beschreiben ab offset 500* *// mit Wert -1000*

6.6.78 WRCBS, write COMMON BUFFER single procedure

BESCHREIBUNG:	Die Prozedur beschreibt eine Speicherzelle vom Typ single (Gleitpunkt-Zahl mit einfacher Genauigkeit) mit dem Wert *value* im CNC-taskspezifischen COMMON BUFFER. Der Parameter *offset* ist ein Byte-Offset bezogen auf das erste Element (Element 0) des COMMON BUFFER. Der Single-Datentyp belegt 4 Bytes im COMMON BUFFER. Um einen korrekten Zugriff durch das MCU-G3-CPU-System zu ermöglichen, muss *offset* immer wortgerichtet sein, d.h. einen durch 4 teilbaren Wert haben.
DEKLARATION:	WRCBS(*offset*:integer; *value*:single)
ANMERKUNG:	Die CNC-Task-spezifische Buffergröße beträgt 1000 Bytes. PCAP-Befehle *rdcbcnct() und wrcbcnct()*, SAP-Befehle *RDCBx()* und *WRCBx().*
BEISPIEL:	*WRCBS(500, 3.99); // Single-Variable beschreiben ab offset 500* *// mit Wert 3.99*

6.6.79 WRCBD, write COMMON BUFFER double procedure

BESCHREIBUNG:	Die Prozedur beschreibt eine Speicherzelle vom Typ double (Gleitpunkt-Zahl mit doppelter Genauigkeit) mit dem Wert *value* im CNC-taskspezifischen COMMON BUFFER. Der Parameter *offset* ist ein Byte-Offset bezogen auf das erste Element (Element 0) des COMMON BUFFER. Der Double-Datentyp belegt 8 Bytes im COMMON BUFFER. Um einen korrekten Zugriff durch das MCU-G3-CPU-System zu ermöglichen, muss *offset* immer doppelwortgerichtet sein, d.h. einen durch 8 teilbaren Wert haben.
DEKLARATION:	WRCBD(*offset*:integer; *value*:double)
ANMERKUNG:	Die CNC-Task-spezifische Buffergröße beträgt 1000 Bytes. PCAP-Befehle *rdcbcnct() und wrcbcnct()*, SAP-Befehle *RDCBx()* und *WRCBx()*
BEISPIEL:	*WRCBD(500, 100.2e-128); // Double-Variable beschreiben ab offset 500* *// mit Wert 100.2e-128*

6.6.80 WRITE

BESCHREIBUNG:	Anhängen eines Teilstrings an die aktuelle taskspezifische Stringausgabe.
FUNKTIONSPARAMETER:	*diverse*
ANMERKUNG:	Die Funktion kann mit einer unbestimmten Anzahl von Parametern aufgerufen werden, die vom Typ Stringkonstante, Integer, Double oder Boolean sein können. Stringkonstanten sind Zeichenketten, die in rw_SymPas durch Hochkommata, in der G-Code Programmierung durch Anführungszeichen begrenzt sind. Die einzelnen Parameter werden durch Kommata getrennt. Numerische oder boolesche Parameter können auch Ausdrücke sein. Der Aufruf dieser Funktion setzt Bit 0 in der Systemvariablen tskinfo. Informationen über den Zustand der Stringausgabe siehe Kapitel 4.4.13. Das Lesen des Taskspezifisischen Ausgabestrings erfolgt mit der PCAP-Funktion gettskstr(), siehe Kapitel 4.4.14.
BEISPIEL RW_SYMPAS:	write ('Dies ist ein String: ', CI0); write ('Istposition: ', A1.rp);
BEISPIELG-CODES:	N0100 write “Dies ist ein String“, CI0

6.6.81 WRITELN

BESCHREIBUNG:	Anhängen eines Teilstrings an die aktuelle taskspezifische Stringausgabe und Abschließen des Ausgabestrings.
FUNKTIONSPARAMETER:	*diverse*
ANMERKUNG:	Die Funktion kann mit einer unbestimmten Anzahl von Parametern aufgerufen werden, die vom Typ Stringkonstante, Integer, Double oder Boolean sein können. Stringkonstanten sind Zeichenketten, die in rw_SymPas durch Hochkommata, in der G-Code Programmierung durch Anführungszeichen begrenzt sind. Die einzelnen Parameter werden durch Kommata getrennt. numerische oder boolesche Parameter können auch Ausdrücke sein. Wenn nach dieser Anweisung write oder writeln erneut aufgerufen wird, wird der bisherige Ausgabestring überschrieben. Der Aufruf dieser Funktion setzt Bit 1 in der Systemvariablen tskinfo. Informationen über den Zustand der Stringausgabe siehe Kapitel 4.4.13. Das Lesen des Taskspezifisischen Ausgabestrings erfolgt mit der PCAP-Funktion gettskstr(), siehe Kapitel 4.4.14. Wenn Bit 26 im Register MODEREG (Kapitel 6.3.1.5) gesezt ist, bewirkt dieses Kommando einen Stopp der jew. CNC-Task.
BEISPIEL RW_SYMPAS:	writeln (‘Dies ist ein String: ’, CI0); writeln (‘Istposition: ’, A1.rp);
BEISPIELG-CODES:	N0100 writeln “Dies ist ein String“, CI0

6.6.82 WT, wait timer

BESCHREIBUNG:	Die als Parameter übergebene Verweilzeit abwarten, bis das SAP-Programm wieder fortgesetzt wird. Dieser Befehl versetzt die CNC-Task in einen inaktiven Zustand und benötigt deshalb keine CPU-Zeit. Um das Master-CPU-System zu entlasten kann dieser Befehl ggf. in Warteschleifen o.ä. eingesetzt werden.
FUNKTIONSPARAMETER:	Integer-Wert mit Einheit 64µs
ANMERKUNG:	Die EVENT-Handling-Prozeduren werden während der Ausführung dieses Befehls nicht abgearbeitet. Sollen diese jedoch überwacht werden, so kann dies beispielsweise durch mehrere *WT()*-Aufrufe mit kürzeren Verweilzeiten (evtl. in einer Schleife) erzwungen werden.
BEISPIEL:	*...* *CONST sec = 15625;* *...* *WT(5*sec); // 5s warten* *...*

6.7 Compilerbefehle

Wie der Name vermuten läßt, weist ein Compilerbefehl den Compiler während der Übersetzung eines Quelltextes an, bestimmte Operationen auszuführen (oder zu unterlassen). In *rw_SymPas* wird ein Compilerbefehl wie folgt aktiviert:

Innerhalb des SAP-Quelltextprogramms wird innerhalb eines Kommentars eine spezielle Syntax formuliert: Direkt auf die öffnende Klammer ({) folgen ein Dollarzeichen ($) und der Name des Befehls, der aus einem oder mehreren Buchstaben besteht. Diese »Kommentare« können - von einigen Ausnahmen abgesehen - an jeder Stelle des Quelltextes erscheinen, an der auch ein normaler Kommentar zulässig ist.

6.7.1 Include-Datei

BESCHREIBUNG:	Dieser Compilerbefehl weist den Compiler an, die durch *Dateiname* bezeichnete Datei einzulesen. Der Compiler verhält sich dabei im Prinzip so, als ob der gelesene Text anstelle des {$I}-Befehls stünde. *rw_SymPas* erlaubt die Verschachtelung von Include-Dateien bis zu 15 Ebenen. Eine via {$I} eingefügte Datei kann also ihrerseits weitere Dateien einfügen, die wiederum {$I}-Befehle enthalten. **Anmerkung:** Sofern in der *mcfg.exe* NCC-Editor-Umgebung bereits eine Include-Datei in einem der drei Editor-Fenster eröffnet wurde, wird der SAP-Quelltext dieses Editors - und nicht der Inhalt der entsprechenden Datei - eingebunden.
SYNTAX:	{$I Dateiname}

6.7.2 Task-Auswahl

BESCHREIBUNG:	Mit diesem Compilerbefehl kann angegeben werden, in welcher Task (*TaskNr*, Werte 0..3) das entsprechende SAP-Programm ablaufen soll. Die Information wird im Autocodefile *„filename.cnc“* abgelegt. Mit Hilfe des PCAP-Befehls *txbf2()* wird dieses File automatisch in die richtige Task übertragen.
SYNTAX:	{$TASK *TaskNr*}
ANMERKUNG:	Sofern das entsprechende SAP-Programm diese Anweisung nicht enthält, wird die momentan selektierte Tasknummer zur Übersetzung herangezogen. Erfolgt aber der ($TASK)-Befehl, wird die entsprechend angewählte Tasknummer auch Default-Tasknummer für alle nachfolgenden Anzeige-, Start- und Stoppbefehle. Kapitel 3.2.1.
BEISPIEL:	*...* *const* *Task1 = 1;* *...* *{$TASK Task1}; // oder* *{$TASK 1}*

6.7.3 Full-System Compilierung

BESCHREIBUNG:	Mit diesem Compilerbefehl kann die Compiler-Option FULLSYSTEM angewählt werden.
SYNTAX:	{$FULLSYSTEM}
ANMERKUNG:	Sofern das entsprechende SAP-Programm diese Anweisung nicht enthält, wird die aktuell selektierte Option zur Übersetzung herangezogen. Diese Anweisung sollte am Anfang des Quelltextfiles stehen. Dieses Kommando ist ab mcfg V2.5.2.13 und ncc ab V2.5.2.9 verfügbar.

6.8 SAP Laufzeitfehler

Bei der Ausführung von Stand-Alone-Programmen können verschiedene Fehler auftauchen. In diesem Fall wird die entsprechende Task angehalten. In der Datenstruktur CNCTS (siehe Kapitel 4.3.2.10) wird dann eine Fehlernummer und die Zeilennummer eingetragen, in welcher der Fehler aufgetreten ist. Nachfolgend sind die Fehlernummern und möglichen Ursachen aufgelistet.

Tabelle 42: SAP Laufzeitfehler

Fehler # dez / hex	Beschreibung
1 / 0001	Die arithmetische Operation ist ungültig für den verwendeten Datentyp-
2 / 0002	Ungültiger Datentyp.
4 / 0004	Ungültiger interner Operationscode. Dies kann auf ein Kompatibilitätsproblem zwischen mcfg/ncc und RWMOS.ELF zurückzuführen sein.
8 / 0008	Stack-Überlauf. Programm zu groß oder internes Problem der RWMOS Betriebssytemsoftware.
16 / 0010	Stack-Unterlauf. Dieser Fehler deutet auf ein internes Problem der RWMOS Betriebssytemsoftware hin.
32 / 0020	Unbekannter Event-Handler. Dies kann auf ein Kompatibilitätsproblem zwischen mcfg/ncc und RWMOS.ELF zurückzuführen sein.
64 / 0040	Ungültiges NC-Kommando. Dies kann auf ein Kompatibilitätsproblem zwischen mcfg/ncc und RWMOS.ELF zurückzuführen sein.
128 / 0080	Adressverletzung im NC-Programm. Dieser Fehler deutet auf ein internes Problem der RWMOS Betriebssytemsoftware hin.
256 / 0100	Adressverletzung im NC-Programm durch falsche Parameterangabe.
512 / 0200	Fehler bei Verwendung des MCUG3 AT-Interface.
1024 / 0400	Eine Common-Variable ausserhalb des gültigen Bereichs wurde adressiert.
2048 / 0800	Üngültiger Index bei einem Double-Zugiff auf den Common Buffer (nicht durch 8 teilbar).
4096 / 1000	Üngültiges SAP Kommando. Dieser Fehler deutet auf ein Kompatibilitätsproblem zu Steuerungen einer anderen Generation hin.
8192 / 2000	Fehler bei Schnittgeschwindigkeitsinterpolation
16384 / 4000	Verwendung unerlaubter Achsen bei Interpolation mit G-Codes
32768 / 8000	ungültiger Parameter bei arithmetischer Operation z.B. mod 0
10000 hex	zu viele SSF Kommandos in einer Spline-Zeile (max. 8 sind möglich)
20000 hex	Rekursionstiefe bei G-Code Unterprogrammen mit Programmwiederholung überschritten (O-Parameter)