Handbuch

Toolbox

Best.-Nr.: VIPA HB74D
Rev. 00/33

Die Angaben in diesem Handbuch erfolgen ohne Gewähr. Änderungen des Inhalts können jederzeit ohne Vorankündigung erfolgen. Die in diesem Handbuch beschriebene Hardware (Software) unterliegt den Bedingungen eines allgemeinen oder besonderen Lizenzvertrags (Einmallizenz) und darf nur in Übereinstimmung mit den Bestimmungen dieses Vertrags verwendet bzw. kopiert werden. Zuwiderhandlungen verpflichten zu Schadenersatz.

Eventuelle Informationen, die erst nach Drucklegung dieses Handbuchs bekannt wurden, finden Sie in einer Datei auf der mitgelieferten Diskette. Um diese, falls sie vorhanden ist, zu lesen, legen Sie die VIPA-Treiberdiskette #1 in Laufwerk A: ein und geben Sie in der Eingabezeile ein:

TYPE README.TXT

unter Windows® verwenden Sie zum Betrachten der Datei bitte den 'NOTEPAD', unter OS/2 den 'E'.

Ohmstraße 4, D-91074 Herzogenaurach
Tel.: +49 (91 32) 744-0
Fax.: +49 (91 32) 744-144
EMail: info@vipa.de
http://www.vipa.de

Hotline: +49 (91 32) 744-114

Über dieses Handbuch

Das Handbuch beschreibt die Handhabung der Utility-Software für die Baugruppen CP386 (CP3) und CP486 (CP4).

Als Betriebssystem wird MS-DOS 5.00 und höher empfohlen. Die Installation und die Beschreibung zu diesem Betriebssystem ist in den zugehörigen Handbüchern von Microsoft bzw. in weiterführenden Büchern aus dem Verlag MARKT UND TECHNIK und DATA BECKER zu finden. Informationen über systeminterne Funktionen sind in dem Buch PC INTERN aus dem Verlag DATA BECKER zu finden.

Für das Betriebssystem MS-DOS wurden von VIPA Utilities und Tools für den Betrieb der CP entwickelt. Die in den Kapiteln beschriebene Software ist auf der VIPA-Diskette CP4-SW593 enthalten.

Überblick

Die in den Kapiteln 1, 2 und 3 beschriebenen Treiber dienen der Kommunikation zwischen der CP-Baugruppe und der SPS. Die drei Treiber können auf allen CP3- bzw. CP4-Baugruppen eingesetzt werden.

In den ersten drei Kapiteln finden Sie Angaben zur Installation der jeweiligen Treiber, und eine Beschreibung der Treiber-Funktionen für die Kommunikation. Auch finden Sie hier die Schnittstellen für PASCAL (nicht CP486NT) und C und eine Beschreibung für den Betrieb unter WINDOWS.

Kapitel 1: Kopplung der SPS mit CP386COM

In diesem Kapitel finden Sie die Beschreibung des CP386COM-Treibers. Dieser Treiber unterstützt den Single-Prozessor-Betrieb. Auch kann hier immer nur ein Blockauftrag abgearbeitet werden.

Kapitel 2: Kopplung der SPS mit CP486COM

Der in diesem Kapitel beschriebene CP486COM-Treiber ist eine Weiterentwicklung des CP386COM. Dieser Treiber ist mehrprozessorfähig. Nach dem FIFO-Prinzip werden die Blockaufträge bearbeitet.

Kapitel 3: Kopplung der SPS mit CP486NT

Hier finden Sie eine Beschreibung des CP486NT. Dieser Treiber ist nur für den Betrieb unter WINDOWS NT ausgelegt. Der Funktionsumfang ist nahezu der gleiche wie beim CP486COM.

Kapitel 4: MS-DOS-Utilities für Silicondisk-Betrieb

Kapitel 4 beinhaltet eine Beschreibung des Silicondisk-Treibers und seine Einbindung in das System. Weiter finden Sie hier die Beschreibung der Programme, die für den Betrieb einer Silicondisk erforderlich sind wie z.B. Formatier-, Lade- und Speicherprogramm.

Kapitel 5: Hilfsprogramme

Dieses Kapitel beinhaltet eine Beschreibung der mitgelieferten Hilfsprogramme zur Rechnerkopplung und zur Visualisierung des Prozeßabbilds. Auch finden Sie hier einen Hinweis auf das System-Testprogramm, das optional erhältlich ist.

Inhaltsverzeichnis

1 Kopplung der SPS mit CP386COM

1 Kopplung der SPS mit CP386COM

1.1 Allgemeine Beschreibung

Der Datentransfer zwischen CP386 und SPS wird durch Hantierungsbausteine auf SPS-Seite und durch Software-Interrupts auf CP-Seite unterstützt. Es stehen folgende Kachelfunktionen zur Verfügung:

Kachel Nr.	Funktion	Bedienung auf SPS-Seite	Bedienung auf CP-Seite
Kachel 0	SPS-Auftrag Daten von CP lesen	Hantierungsbaustein FB3	Interruptservice-Routine
Kachel 1	SPS-Auftrag Daten an CP senden	Hantierungsbaustein FB3	Interruptservice-Routine
Kachel 2	CP-Auftrag Daten von SPS lesen	zykl. aufgerufener Hantierungsbaustein FB1	Softwareinterrupt
Kachel 3	CP-Auftrag Daten an SPS senden	zykl. aufgerufener Hantierungsbaustein FB1	Softwareinterrupt
Kachel 7	Prozeßabbild an CP übertragen	zykl. aufgerufener Hantierungsbaustein FB1	Softwareinterrupt oder direkter Zugriff auf die Kachel

Tab. 1-1: Übersicht der Kachelfunktionen unter CP386COM

Sie können folgende Datenstrukturen in der SPS auf der CP-Seite ansprechen:

- einzelne Elemente im Format Bit, Byte, Wort und Doppelwort
  DB, DX, Merker, Eingänge, Ausgänge, Timer, Zähler, Anzeigenwort
- Datenblöcke
  DB, DX, FB, FX, OB, PB, SB, BA, BB, BT, BS

Folgende SPS-Zugriffe auf die CP sind möglich:

- Die Kopplung unterstützt alle Arten von MS-DOS-Device-orientierten Zugriffen.
- Die Aufträge die von der SPS abgesetzt werden können, basieren auf den MS-DOS-Device-Funktionen.

Die im folgenden beschriebenen Funktionen stehen ab CP386COM-Version 1.00 (Software SW593 Version 2.x) und Hantierungsbaustein (Software SW973 Version 2.x) zur Verfügung. Das Programm CP386COM wird in der folgenden Beschreibung als COM-Treiber bezeichnet.

1.1.1 MS-DOS-Treiber

Zur Kommunikation über die Kacheln zwischen AG und CP ist in der CP ein spezieller Kommunikationstreiber erforderlich. Dieser Treiber ist speziell zur Kommunikation mit den VIPA-Hantierungsbausteinen ausgelegt. Er stellt einfach zu handhabende Funktionen bereit, die den Umgang mit den Kacheln erleichtern sollen. Wissen über Aufbau und Wirkungsweise der Kacheln ist nicht erforderlich.

Unterstützt werden

- Kacheln 0 und 1 (AG aktiv, CP passiv)
- Kacheln 2 und 3 (AG passiv, CP aktiv)
- Kachel 7 (Prozeßabbild).

Der Treiber stellt automatisch alle Funktionen für die erforderlichen Kacheln bereit; eine weitere Konfiguration ist nicht erforderlich.

1.1.2 Installation des MS-DOS-Treibers

Der Treiber sollte beim Start der CP geladen werden. Hierzu binden Sie den Treiber in Ihre AUTOEXEC.BAT ein. Ein späterer Aufruf ist ebenfalls möglich.

Während des Neustarts werden in der Regel die Hantierungsbausteine zur Synchronisation aufgerufen. Diese warten nur eine bestimmte Zeit auf eine Reaktion der CP.

Wird innerhalb dieser Zeit der Treiber der CP nicht gestartet, so sind AG und CP unsynchronisiert und es ist keine Kommunikation möglich.

Aufruf: CP386com.exe [/ini] [/txx] [/ixx] [/notsr] [/?] [/h]

Der Treiber ist ein speicherresidentes Programm (TSR-Utility) und belegt ca. 26 KByte des Arbeitsspeichers (Programm und Daten). Der Treiber kann immer nur einmal geladen sein. Jeder weitere Aufruf bewirkt die Ausgabe, daß der Treiber bereits installiert ist.

Sie können den Treiber nicht mehr aus dem Arbeitsspeicher entfernen. Zum Entfernen müssen Sie den CP neu booten.

1.1.3 Optionen des Treibers ab Version 1.6

/INI oder /ini

Durch diese Option wird der Kachelbereich der CP neu initialisiert, d.h. die laufenden Aufträge werden abgebrochen und die Kacheln werden gelöscht. Der Treiber wird bei dieser Option nicht installiert und kann so jederzeit zur Initialisierung aufgerufen werden.

/Txx oder /txx

Bei der Installation wird durch diese Option der Timeout des Treibers auf den angegebenen Wert eingestellt.

Gültige Werte für xx: 1s bis 30s.

Standardvorgabe: 10s

/Ixx oder /ixx

Bei der Installation wird durch diese Option die Nummer für den Service-Interrupt zur Kommunikation über die Kachel 2 und 3 eingestellt.

Gültige Werte für xx: 78h, 7Ah bis 7Fh.

Standardvorgabe: 78h.

/NOTSR oder /notsr

Diese Option verhindert, daß der Treiber speicherresident geladen wird.

Das Programm wird nicht beendet, so daß anschließend auch keine weiteren DOS-Befehle mehr eingegeben oder Programme gestartet werden können. Diese Option ist nur sinnvoll, bei ausschließlicher Kommunikation über Kacheln 0 und 1 (CP passiv).

Durch Drücken der Taste F10 und anschließender Bestätigung mit "j" wird der Treiber wieder entfernt.

/?, /H oder /h

Über diesen Parameter werden die möglichen Optionen der Kommandozeile mit den dazugehörigen Wertebereichen als Hilfetext ausgegeben.

Der COM-Treiber ist ausschließlich für CP-Baugruppen der VIPA GmbH entwickelt und kann nur auf diesen Systemen installiert werden. Wird der Treiber auf anderen CP-Systemen geladen, auch mit i386 oder i486 Prozessor, so kann dies ein sofortiges Aufhängen des Rechners zur Folge haben. Datenverluste sind nicht ausgeschlossen.

1.1.4 Belegte Interrupts

Der Treiber verwendet mehrere Software-Interrupts zum Betrieb und zur Kommunikation mit der Anwendungssoftware in der CP-Baugruppe. Die Interrupts sind wie folgt belegt:

- INT 1Ch Timer-Interrupt und INT 28 DOS-Idle-Interrupt

  Für regelmäßige, zyklische Überprüfungen der Kachel wird der sogenannte Ticker-Interrupt mit der Nummer 1Ch, sowie der sog. DOS-Idle Interrupt verwendet. Hiermit kann z.B. regelmäßig überprüft werden, ob das AG die Kacheln neu synchronisieren will.

  Nach Ausführung der CP-spezifischen Funktionen wird die ursprüngliche Interrupt-Behandlungsroutine (Interrupt Service Routine) aufgerufen.

- INT 74h (IRQ 12):

  Die CP verwendet den Hardware-IRQ 12, der den Software-Interrupt 74h belegt. Dieser Interrupt wird immer ausgelöst, wenn im AG BASP aktiv ist oder auf die höchste Speicherzelle jeder Kachel (Byte1023) vom AG geschrieben wurde. Die Verwendung des Interrupts ermöglicht eine schnelle Reaktion der CP auf Anforderungen durch die SPS.

  Nach Bearbeitung der CP-spezifischen Funktionen wird die ursprüngliche Interrupt-Behandlungsroutine aufgerufen. Dadurch können auch mehrere Geräte den IRQ 12 verwenden.

- INT 78h Service-Interrupt:

  Dieser Interrupt ist von der Anwender-Software in der CP zu verwenden, um Funktionen des Treibers aufzurufen wie z.B. den Datenaustausch mit dem AG über die Kacheln 2 und 3. Durch entsprechende Belegung der Prozessor-Register können unterschiedliche Funktionen ausgelöst werden. Wird der Int 78h mit Register-Werten aufgerufen, die für die CP-Kommunikation ungültig sind, so wird die ursprüngliche Interrupt-Behandlungsroutine aufgerufen.

1.1.5 Unterschiedliche Darstellung von Daten im Speicher

Bei der Übertragung von Daten zwischen CP und AG muß die unterschiedliche Darstellung von Worten und Doppelworten (Langworten) auf CP und AG berücksichtigt werden.

Im CP sind im Unterschied zum AG Datenworte in anderer Form im Speicher abgelegt, das höherwertige (High-Byte) und das niederwertige Byte (Low-Byte) sind vertauscht abgespeichert. Bei Doppelworten sind alle 4 Bytes genau im umgekehrter Reihenfolge abgespeichert. Werden zwischen AG und CP Daten vom Typ Wort oder Doppelwort ausgetauscht, so muß natürlich irgendwann eine Vertauschung vorgenommen werden, sonst erhält man nach der Übertragung falsche Daten. Soweit dies sinnvoll möglich ist, übernimmt der COM-Treiber automatisch die Anpassung der Daten.

Bei Datenaustausch über die Kacheln 0 und 1 muß der Anwender die Vertauschung, entweder auf CP- oder auf AG-Seite, selbst vornehmen. Bei den Kacheln 2, 3 und 7 wird die Vertauschung in allen Fällen automatisch vom Treiber durchgeführt.

- Bei der Übertragung von Bytes findet keine Vertauschung statt.
- Bei der Übertragung von Worten werden High- und Low-Byte vertauscht.
- Bei der Übertragung von Langworten werden alle 4 Bytes in ihrer Reihenfolge umgedreht.

Darstellung von Daten im AG

Adresse n	Byte	Darstellung Byte
Adresse n	High-Byte	Darstellung Wort
Adresse n+1	Low-Byte	
Adresse n	High-Byte High-Word	Darstellung Doppelwort
Adresse n+1	Low-Byte High-Word	
Adresse n+2	High-Byte Low-Word	
Adresse n+3	Low-Byte Low-Word	

Darstellung von Daten im CP

Adresse n	Byte	Darstellung Byte
Adresse n	Low-Byte	Darstellung Wort
Adresse n+1	High-Byte	
Adresse n	Low-Byte Low-Word	Darstellung Doppelwort
Adresse n+1	High-Byte Low-Word	
Adresse n+2	Low-Byte High-Word	
Adresse n+3	High-Byte High-Word	

1.2 SPS-Aufträge für die CP (Funktionen für Kachel 0 und 1)

1.2.1 Übersicht

Über die Kacheln 0 und 1 werden SPS-Aufträge abgewickelt. Die CPU hat so die Möglichkeit eine Reihe von MS-DOS-Funktionen zu nutzen.

Die SPS-Aufträge werden, sobald der Treiber installiert ist, im Hintergrund über Interrupts abgearbeitet. Es ist keine weitere Software auf der CP-Seite erforderlich. Sie können mit dem Treiber verschiedene MS-DOS Systemfunktionen aus dem AG heraus aufrufen. Hierbei werden alle Parameter und Informationen transparent zwischen AG und CP ausgetauscht, indem die übergebenen Parameter richtig in die Prozessor-Register eingetragen und zurückgegebene Werte in einer passenden Form an das AG weitergegeben werden.

Es macht Sinn nur einen Teil der MS-DOS-Systemfunktionen aus dem AG heraus aufzurufen, da ein Großteil von Systemaufrufen zu einem Systemabsturz führen können. Zur Vermeidung solcher Fehlfunktionen unterstützt die Treiber-Software nur solche Funktionen, die sinnvoll durch die SPS verwendet werden können. Geben Sie bei einem Aufruf als Funktionsnummer die jeweiligen Funktionsnummern der MS-DOS System-Funktion an.

Kachel-Nr	Funktions-Nr		Funktion
	hex	dez	
1	0Dh	13	alle Laufwerke zurücksetzen
1	0Eh	14	Laufwerk anwählen
0	19h	25	aktuelles Laufwerk bestimmen
1	39h	57	Verzeichnis anlegen
1	3Ah	58	Verzeichnis löschen
1	3Bh	59	Verzeichnis wechseln
0	47h	71	aktuelles Verzeichnis bestimmen
1	3Ch	60	Datei erzeugen
1	5Ah	90	Datei erzeugen ohne überschreiben
1	3Dh	61	Datei öffnen
1	68h	104	Datei physikalisch auf Disk schreiben (ohne schließen)
1	3Eh	62	Datei schließen
1	41h	65	Datei löschen
1	56h	86	Datei umbenennen
1	42h	66	Datei-Zeiger setzen
0	C2h	194	Datei-Zeiger lesen (keine MS-DOS System-Funktion!)
0	3Fh	63	von Datei oder Device lesen
1	40h	64	auf Datei oder Device schreiben
0	2Ah	42	Datum lesen
0	2Ch	44	Zeit lesen
1	4Bh	75	Programm ausführen
0	30h	48	MS-DOS Version bestimmen
0	59h	89	ausführliche Fehler-Information lesen
1	FFh	255	beliebiger Interrupt

Tab. 1-2: Übersicht MS-DOS Systemfunktionen

1.2.2 Schnittstellenkonzept für die Kacheln 0 und 1

Diese beiden Kacheln dienen zum Senden und Empfangen von Daten aus bzw. in die CP. Für das Senden und Empfangen von Daten sind in der SPS Hantierungsbausteine (SEND bzw. FETCH und RECEIVE) erforderlich. Diese Hantierungsbausteine erstellen dann einen Auftragsblock in der Kachel 0 bzw. Kachel 1. In den Kacheln 0 und 1 kann zu jedem Zeitpunkt jeweils maximal ein Auftrag eingetragen sein. Die Größe der zu übertragenden Daten kann von einem Wort bis zu 504 Worte betragen. Der Aufbau eines Auftragsblocks innerhalb der Kacheln 0 und 1 ist völlig identisch. Durch die Kachelnummer wird zwischen Senden und Empfangen unterschieden.

Auf der CP-Seite ist ein Auftragskatalog hinterlegt. Sobald die CP einen Auftrag in der Kachel 0 bzw. in der Kachel 1 registriert, entnimmt sie dem Auftragsblock die Auftragsnummer und sucht den entsprechenden Parameterblock auf ihrer Seite im Auftragskatalog. In diesem Katalog ist hinterlegt, wie die Daten, die z.B. von der SPS an die CP übertragen werden, zu handhaben sind.

Für das Empfangen gilt entsprechendes, d.h. die CP sucht anhand der Auftragsnummer in der Kachel ob auf ihrer Seite ein Katalog hinterlegt ist. Wenn ja, stellt sie die angeforderten Daten entsprechend ihrem Katalog zur Verfügung.

1.2.3 Ablauf eines Sendeauftrags

Die Anwendersoftware ruft den Hantierungsbaustein SEND auf. Hier parametrieren Sie die Auftragsnummer, die Übertragungslänge in Worten und die Quelle der Daten im AG. Der Hantierungsbaustein überprüft alle Angaben auf Gültigkeit. Sind alle Angaben richtig, so wird noch geprüft ob die Kachel frei ist. Die Kachel ist frei, wenn kein laufender Auftrag abgelegt ist. Bei einer belegten Kachel wird der Sendeauftrag abgelehnt.

Bei einer freien Kachel erstellt der Hantierungsbaustein einen Auftragsblock, legt die zu schreibenden Daten im Anschluß an den Auftragsblock in der Kachel ab und setzt den Auftrags-Status auf "Auftrag läuft". An "Auftrag läuft" erkennt die CP, daß in der Kachel ein neuer Auftrag steht, der bearbeitet werden muß.

Abhängig davon ob der Auftrag ausführbar ist, setzt die CP nach Ausführung des Auftrages die Kennung "Auftrag läuft" zurück und setzt entweder "Auftrag fertig mit Fehler" oder "Auftrag fertig ohne Fehler". Bei "Auftrag fertig mit Fehler" liefert die CP eine entsprechende Fehlernummer mit.

Sie können mit dem Hantierungsbaustein CONTROL den Status des laufenden bzw. des letzten Auftrags ausgeben.

1.2.4 Ablauf eines Empfangsauftrags

Mit dem Hantierungsbaustein FETCH übergibt die Anwender-Software einen Empfangsauftrag an die CP. Auch hier werden wie bei SEND die eingegebenen Parameter überprüft. Der Hantierungsbaustein erstellt einen Auftragsblock in der Kachel 0 und setzt den Status auf "Auftrag läuft". Hier erkennt die CP den neuen Auftrag und bearbeitet diesen wie unter Senden bereits beschrieben. Kann die CP die angeforderten Daten zur Verfügung stellen, so legt sie diese im Anschluß an den Auftragsblock in der Kachel 0 ab und setzt den Status auf "Fertig ohne Fehler". Zusätzlich setzt sie die Kennung "Daten für Receive vorhanden".

Mit dem Hantierungsbaustein CONTROL können Sie zu jedem Zeitpunkt den Status ausgeben. Sind die Daten von der CP bereitgestellt, so können Sie diese mit dem Hantierungsbaustein RECEIVE in das AG übertragen. Bei fehlerfreier Übertragung setzt der Hantierungsbaustein RECEIVE die Kennung "Daten für Receive vorhanden" zurück. Ab diesem Zeitpunkt kann von der Software ein neuer FETCH-Auftrag eingetragen werden.

Sobald während eines laufenden SEND- bzw. RECEIVE-Auftrags ein weiterer Auftrag gestartet wird, wird an die Software die Kennung "Schnittstelle belegt" ausgegeben. Diese Kennung vergeben jedoch nur die Hantierungsbausteine. Sie haben für die Kommunikation mit der CP keinerlei Bedeutung.

Die Schnittstelle (Datenaufbau in den Kacheln) ist in der aktuellen Version nicht für Multiprozessorbetrieb im AG geeignet. Zu jedem Zeitpunkt darf immer nur ein und dieselbe CPU auf eine im AG gesteckte CP zugreifen !

1.2.5 Parametrierung der Hantierungsbausteine

Die Hantierungsbausteine SEND (FB3), CONTROL (FB4), FETCH (FB5) und RECEIVE (FB6) werden folgendermaßen parametriert:

Falls die Kachelnummer der CP nicht mit der im Hantierungsbaustein parametrierten Nummer übereinstimmt, geht die CPU mit QVZ in Stop!

1.2.5.1 FB3 (SEND), Send-Auftrag an CP

FB3 (SEND) übergibt einen Datenblock mit bis zu 504 Worten aus einem DB an die CP. Zur Identifikation wird eine Auftragsnummer an die CP mit übergeben. Das Ergebnis liefert der Hantierungsbaustein über ein Anzeigewort in einem Merkerwort an das Anwenderprogramm. Parametrierungsfehler werden über ein Merkerbyte gemeldet. Der Hantierungsbaustein ist direkt und indirekt parametrierbar. SEND arbeitet mit der relativen Kachelnummer 1.

Mit dem Aufruf von FB3 sind folgende Parameter zu übergeben:

Bez.	Format	Beschreibung
INSS	KY	Kennung Parametrierung/Basiskachel
A-NR	KY	Auftragsnummer und Funktionsnummer
DOSP	KY	DOS-Parameter
ANZW	KY	Merkerwort für Anzeigewort
QT/N	KY	Quellbausteinnummer
QANF	KF	Anfangsadresse im DB
QLAE	KF	Anzahl Datenworte
PAFE	BY	Merkerbyte für Fehlermeldungen

Tab. 1-3: Parameterliste für den Aufruf von FB3

Für die o.g. Parameter sind im einzelnen die folgenden Angaben erforderlich:

INSS IN Kennung, ob direkte oder indirekte Parametrierung

- = 0 direkte Parametrierung über die Formaloperanden
- ≠ 0 indirekte Parametrierung, die Übergabeparameter sind im offenen DB abgelegt

SS Nummer der Basiskachel (muß durch 8 dividierbar sein)

A-NR Bei direkter Parametrierung enthält das linke Byte die Auftragsnummer (1..127) und das rechte Byte die Funktionsnummer für den CP-Treiber.

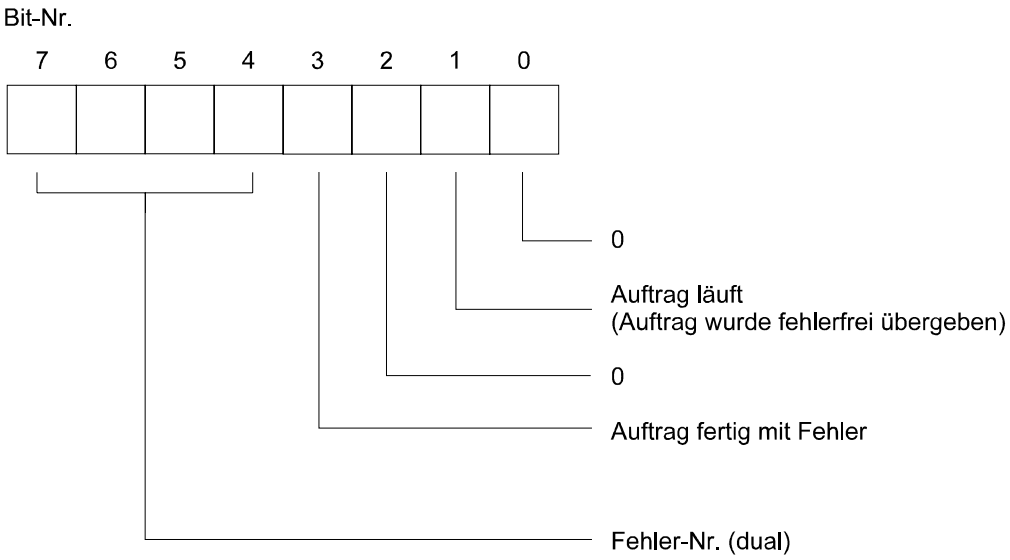
Bei indirekter Parametrierung enthält A-NR die DW-Nr, ab der die Parameter im offenen DB liegen. Hierbei wird der Inhalt des Parameters als Wort ausgewertet.

DOSP DOS-Parameter, der bei bestimmten Funktionen mitübergeben wird.

ANZW linkes Byte Reserve

rechtes Byte Enthält die MW-Nr., in dem das Anzeigenwort abgelegt werden soll (zulässig sind MW0..MW198)

Das Anzeigenwort kann beim SEND folgende Informationen enthalten:



Die Fehlernummern sind dual codiert.

Fehlernummer:	1	Schnittstelle belegt von SPS (Auftrag läuft)
	6	Schnittstelle belegt von CP

QT/N linkes Byte Reserve

rechtes Byte Quellbausteinnummer (2...255), es wird die DB-Nr. des Bausteins mit den zu übertragenden Daten angegeben

QANF Anfangsadresse im DB (0...32761), es wird die DW-Nr. angegeben, ab der die zu übertragenden Daten im DB abgelegt sind.

QLAE Anzahl der zu übertragenden Datenworte (1...504)

PAFE Merkerbyte, in dem die PAFE-Meldung an das Programm übergeben wird (zulässig 0...255)

Fehlerrückmeldung des Hantierungsbausteins:

= 0 es ist kein Fehler aufgetreten

≠ 0 es ist ein Fehler aufgetreten, Fehlernummer im PAFE-Byte:

- 3 Die Basiskachelnummer ist nicht durch 8 teilbar
- 5 Die Kachel ist noch nicht von der CP synchronisiert
- 10 ungültige Auftragsnummer (außerhalb von 1...127)
- 12 kein DB für indirekte Parametrierung aufgeschlagen
- 13 Quellbaustein ist nicht vorhanden
- 14 Quellbaustein ist zu kurz
- 15 QLAE ist ungültig (außerhalb von 1...504)
- 16 DB für indirekte Parametrierung zu kurz
- 18 ungültige Quellbaustein-Nr. (außerhalb von 2...255)
- 19 ungültige Quellanfangsadresse (außerhalb von 0...32761)
- 20 ungültige Merkerwort-Nr. für ANZW (außerhalb von 0...198)

Ablage der Parameter in einem DB bei indirekter Parametrierung:

A-NR zeigt auf den Beginn

DL	DR
INSS	
A-NR	F-NR
DOSP	
ANZW	
QT/N	
QANF	
QLAE	

Hinweis

PAFE ist nicht indirekt parametrierbar.

Bei allen anderen Formaloperanden (SS, DOSP, ANZW, QT/N, QANF, QLAE) kann 0 parametriert werden, da diese bei indirekter Parametrierung nicht ausgewertet werden.

1.2.5.2 FB4 (CONTROL), Status des CPs ausgeben

Dieser Hantierungsbaustein liefert den Status eines Schreib- oder Lese-Auftrags. Zur Identifikation wird eine Auftragsnummer mit an die CP übergeben. Den Auftrags-Status liefert der Hantierungsbaustein über ein Anzeigewort in einem Merkerwort an das Anwenderprogramm. Parametrierungsfehler werden über ein Merkerbyte gemeldet. Der Hantierungsbaustein ist direkt und indirekt parametrierbar. CONTROL arbeitet mit den relativen Kachelnummer 0 und 1.

Mit dem Aufruf von FB4 sind folgende Parameter zu übergeben:

Bez.	Format	Beschreibung
INSS	KY	Kennung Parametrierung/Basiskachel
A-NR	KF	Auftragsnummer und Funktionsnummer
DOSP	KY	DOS-Parameter
RWAW	KY	Auftragsart/Anzeigewort
PAFE	BY	Merkerbyte für Fehlermeldungen

Tab. 1-4: Parameterliste für den Aufruf von FB4

Für die o.g. Parameter sind im einzelnen die folgenden Angaben erforderlich:

INSS IN Kennung, ob direkte oder indirekte Parametrierung

- = 0 direkte Parametrierung über die Formaloperanden
- ≠ 0 indirekte Parametrierung, die Übergabeparameter sind im offenen DB abgelegt

SS Nummer der Basiskachel (muß durch 8 dividierbar sein)

A-NR Bei direkter Parametrierung enthält das rechte Byte die Auftragsnummer (0..127). Bei indirekter Parametrierung enthält A-NR die DW-Nr, ab der die Parameter im offenen DB liegen. Dabei wird der Inhalt des Parameters als Wort ausgewertet.

Bei Auftragsnummer 1...127 wird der Status des entsprechenden Auftrags gelesen. Bei Auftragsnummer 0 wird der Status des gerade laufenden, bzw. des zuletzt ausgeführten Auftrags gelesen.

DOSP linkes Byte: Reserve

rechtes Byte: enthält die MW-Nr., in dem der DOS-Parameter abgelegt werden soll.

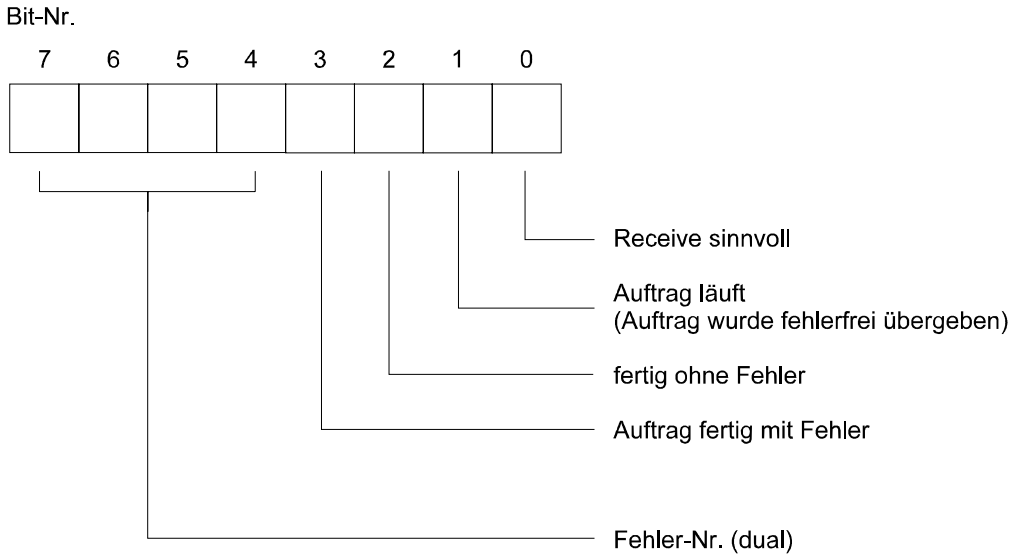
RWAW

RW 0 = Control für Leseauftrag in Kachel 0

1 = Control für Schreibauftrag in Kachel 1

AW enthält die MW-Nr., in dem das Anzeigenwort abgelegt werden soll (zulässig sind MW0..MW198)

Das Anzeigenwort kann bei CONTROL folgende Informationen enthalten:



Die Fehlernummern sind dual codiert.

Fehlernummer
- 4 undefinierter Auftrags-Status auf der CP
- 5 kein Auftrag unter dieser Auftragsnummer
- 6 Schnittstelle belegt von CP

Die Bits 8-15 enthalten eine eventuelle Fehlernummer der CP

PAFE

Merkerbyte, in dem die PAFE-Meldung an das Programm übergeben wird (zulässig 0...255).

Fehlerrückmeldung des Hantierungsbausteins:

= 0 es ist kein Fehler aufgetreten

≠ 0 es ist ein Fehler aufgetreten, Fehlernummer im PAFE-Byte:

- 3 Die Basiskachelnummer ist nicht durch 8 teilbar
- 4 Die Kachel ist nicht vorhanden (Quittungsverzug bei Zugriff auf die Kachel)
- 5 Die Kachel ist noch nicht von der CP synchronisiert
- 10 ungültige Auftragsnummer (außerhalb von 1...127)
- 12 kein DB für indirekte Parametrierung aufgeschlagen
- 16 DB für indirekte Parametrierung zu kurz
- 20 ungültige Merkerwort-Nr. für RWAW (außerhalb von 0...198)
- 21 ungültige Merkerwort-Nr. für DOSP (außerhalb von 0...198)

Ablage der Parameter in einem DB bei indirekter Parametrierung:

A-NR zeigt auf den Beginn

DL DR
INSS
A-NR
DOSP
RWAW

Hinweis

PAFE ist nicht indirekt parametrierbar.

Bei allen anderen Formaloperanden (SS, DOSP, RWAW) kann 0 parametriert werden, da diese bei indirekter Parametrierung nicht ausgewertet werden.

1.2.5.3 FB5 (FETCH), Datenanforderung an CP

FB5 (FETCH) übergibt eine Datenanforderung an den CP. Zur Identifikation wird eine Auftragsnummer mit an den CP mitübergeben. Das Ergebnis liefert der Hantierungsbaustein über ein Anzeigewort in einem Merkerwort an das Anwenderprogramm. Parametrierungsfehler werden über ein Merkerbyte gemeldet. Der Hantierungsbaustein ist direkt und indirekt parametrierbar. FETCH arbeitet mit der relativen Kachelnummer 0.

Mit dem Aufruf von FB5 sind folgende Parameter zu übergeben:

Bez.	Format	Beschreibung
INSS	KY	Kennung Parametrierung/Basiskachel
A-NR	KF	Auftragsnummer und Funktionsnummer
DOSP	KY	DOS-Parameter
LAE	KF	Anzahl Datenworte
ANZW	KY	Anzeigewort
PAFE	BY	Merkerbyte für Fehlermeldungen

Tab. 1-5: Parameterliste für den Aufruf von FB5

Für die o.g. Parameter sind im einzelnen die folgenden Angaben erforderlich:

INSS IN Kennung, ob direkte oder indirekte Parametrierung

- = 0 direkte Parametrierung über die Formaloperanden
- ≠ 0 indirekte Parametrierung, die Übergabeparameter sind im offenen DB abgelegt

SS Nummer der Basiskachel (muß durch 8 dividierbar sein)

A-NR Bei direkter Parametrierung enthält das linke Byte die Auftragsnummer (1..127) und das rechte Byte die Funktionsnummer für den CP-Treiber.

Bei indirekter Parametrierung enthält A-NR die DW-Nr, ab der die Parameter im offenen DB liegen. Dabei wird der Inhalt des Parameters als Wort ausgewertet.

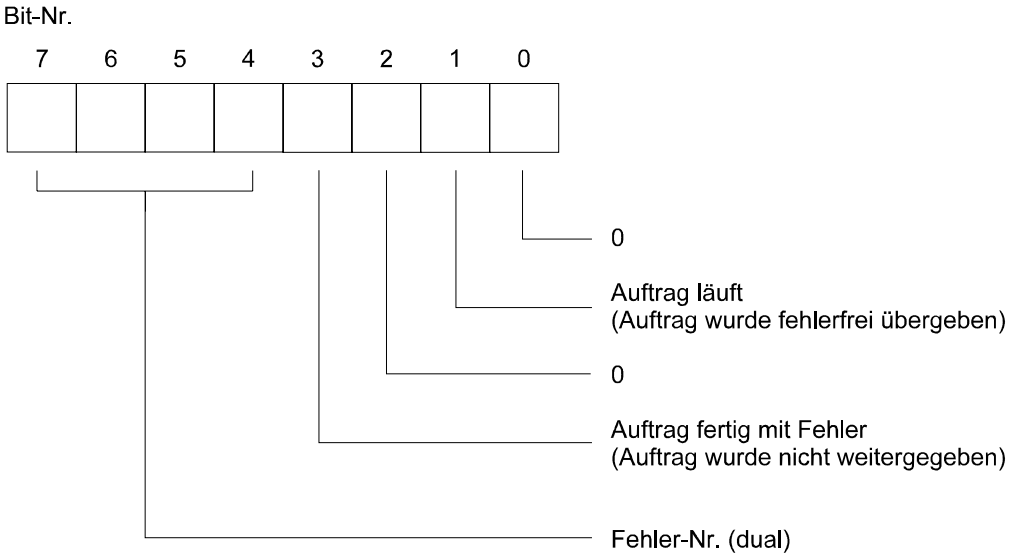
DOSP DOS-Parameter, der bei bestimmten Funktionen mitübergeben wird.

LAE Anzahl der zu lesenden Datenworte, entsprechend der übergebenen Funktionsnr. für den DOS-Treiber, z.B. Anzahl der Daten, die aus einem File gelesen werden sollen.

ANZW linkes Byte Reserve

rechtes Byte Enthält die MW-Nr., in dem das Anzeigenwort abgelegt werden soll (zulässig sind MW0..MW198)

Das Anzeigenwort kann beim FETCH folgende Informationen enthalten:



Die Fehlernummern sind dual codiert.

Fehlernummer: 1 Schnittstelle belegt von SPS (Auftrag läuft)
6 Schnittstelle belegt von CP

PAFE Merkerbyte, in dem die PAFE-Meldung an das Programm übergeben wird (zulässig 0...255).

Fehlerrückmeldung des Hantierungsbausteins:

= 0 es ist kein Fehler aufgetreten

≠ 0 es ist ein Fehler aufgetreten, Fehlernummer im PAFE-Byte:

- 3 Die Basiskachelnummer ist nicht durch 8 teilbar
- 5 Die Kachel ist noch nicht von der CP synchronisiert
- 10 ungültige Auftragsnummer (außerhalb von 1...127)
- 12 kein DB für indirekte Parametrierung aufgeschlagen
- 16 DB für indirekte Parametrierung zu kurz
- 20 ungültige Merkerwort-Nr. für ANZW (außerhalb von 0...198)

Ablage der Parameter in einem DB bei indirekter Parametrierung:

A-NR zeigt auf den Beginn

DL	DR
INSS	
A-NR	F-NR
DOSP	
LAE	
ANZW	

Hinweis

PAFE ist nicht indirekt parametrierbar.

Bei allen anderen Formaloperanden (SS, DOSP, LAE, ANZW) kann 0 parametriert werden, da diese bei indirekter Parametrierung nicht ausgewertet werden.

1.2.5.4 FB6 (RECEIVE), Daten von CP empfangen

FB6 (RECEIVE) empfängt einen Datenblock mit bis zu 504 Worten von der CP und legt diesen in einen DB ab. Vor dem Aufruf des RECEIVE-Bausteins müssen Sie mit dem FETCH-Hantierungsbaustein die Daten im CP anfordern. Zur Identifikation wird eine Auftragsnummer an den CP mitübergeben. Der CP liefert diese Auftragsnummer zusammen mit den Daten zurück.

Das Ergebnis liefert der Hantierungsbaustein über ein Anzeigewort in einem Merkerwort an das Anwenderprogramm. Parametrierungsfehler werden über ein Merkerbyte gemeldet. Der Hantierungsbaustein ist direkt und indirekt parametrierbar. RECEIVE arbeitet mit der relativen Kachelnummer 0.

Mit dem Aufruf von FB6 sind folgende Parameter zu übergeben:

Bez.	Format	Beschreibung
INSS	KY	Kennung Parametrierung/Basiskachel
A-NR	KF	Auftragsnummer und Funktionsnummer
ANZW	KY	Anzeigewort
ZT/N	KY	Zielbausteinnummer
ZANF	KF	Anfangsadresse Zielbaustein
ZLAE	KF	Anzahl der Datenworte
PAFE	BY	Merkerbyte für Fehlermeldungen

Tab. 1-6: Parameterliste für den Aufruf von FB6

Für die o.g. Parameter sind im einzelnen die folgenden Angaben erforderlich:

INSS IN Kennung, ob direkte oder indirekte Parametrierung

- = 0 direkte Parametrierung über die Formaloperanden
- ≠ 0 indirekte Parametrierung, die Übergabeparameter sind im offenen DB abgelegt

SS Nummer der Basiskachel (muß durch 8 dividierbar sein)

A-NR Bei direkter Parametrierung enthält A-NR die Auftragsnummer (0..127).

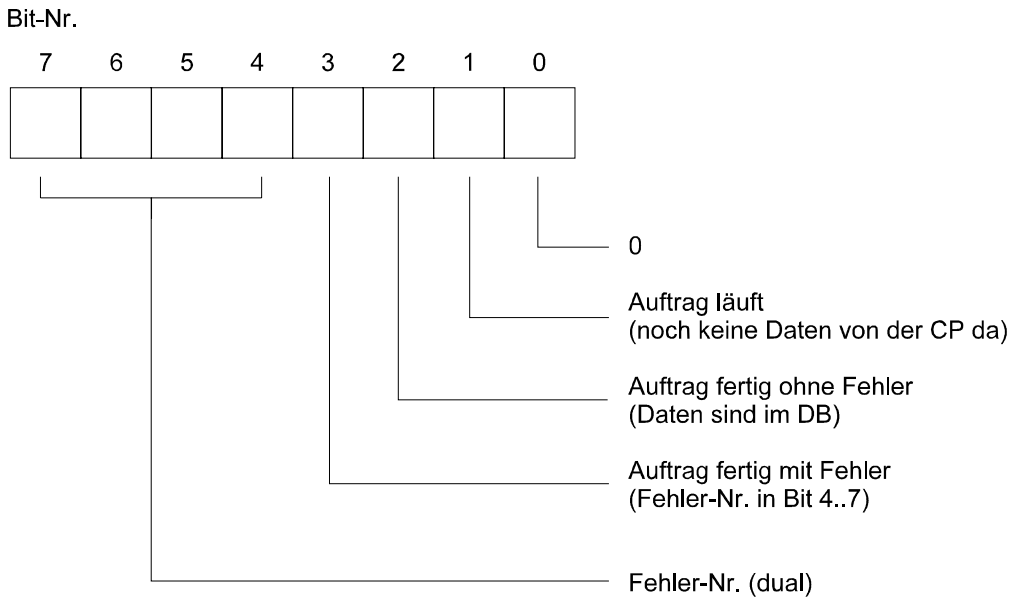
Bei einer Nummer 1...127 wird geprüft, ob die Daten, welche die CP bereitstellt, die gleiche Auftragsnummer haben. Die Daten werden nur bei gleicher Auftragsnummer übernommen. Bei Auftragsnummer 0 werden die Daten auf jeden Fall von der CP übernommen.

Bei indirekter Parametrierung enthält A-NR die DW-Nr, ab der die Parameter im offenen DB liegen. Dabei wird der Inhalt des Parameters als Wort ausgewertet.

ANZW linkes Byte Reserve

rechtes Byte Enthält die MW-Nr., in dem das Anzeigenwort abgelegt werden soll (zulässig sind MW0..MW198)

Das Anzeigenwort kann beim RECEIVE folgende Informationen enthalten:



Die Fehlernummern sind dual codiert.

Fehlernummer: 2 keine Daten vorhanden
3 keine Daten für diesen Auftrag vorhanden
4 undefinierter Auftrags-Status auf der CP
6 Schnittstelle belegt von CP

ZT/N linkes Byte: Reserve

rechtes Byte: Zielbausteinnummer (2...255), es wird die DB-Nr. des Bausteins angegeben, in dem die Daten abgelegt werden sollen.

ZANF Anfangsadresse im DB (0...32761), es wird die DW-Nr. angegeben, ab der die zu übertragenden Daten im DB abgelegt werden sollen.

ZLAE Anzahl der maximal zu übertragenden Datenworte (1...504), es werden nur so viele Datenworte von der CP übertragen, wie diese liefert.

PAFE Merkerbyte, in dem die PAFE-Meldung an das Programm übergeben wird (zulässig 0...255).
Fehlerrückmeldung des Hantierungsbausteins:

= 0 es ist kein Fehler aufgetreten

≠ 0 es ist ein Fehler aufgetreten, Fehlernummer im PAFE-Byte:

- 3 Die Basiskachelnummer ist nicht durch 8 teilbar
- 4 Kachel ist nicht vorhanden (Quittungsverzug bei Zugriff aufgetreten)
- 5 Die Kachel ist noch nicht von der CP synchronisiert
- 10 ungültige Auftragsnummer (außerhalb von 1...127)
- 12 kein DB für indirekte Parametrierung aufgeschlagen
- 13 Zielbaustein ist nicht vorhanden
- 14 Zielbaustein ist zu kurz
- 15 ZLAE ist ungültig (außerhalb von 1...504)
- 16 DB für indirekte Parametrierung zu kurz
- 18 ungültige Zielbaustein-Nr. (außerhalb von 2...255)
- 19 ungültige Zielanfangsadresse (außerhalb von 0...32761)
- 20 ungültige Merkerwort-Nr. für ANZW (außerhalb von 0...198)

Die Fehlernummer 4 im PAFE ist für zukünftige Verbesserungen gedacht. Momentan ist es in den CPUs für das AG115 nicht generell möglich QVZ per Software zu erkennen. In dieser Version geht die CPU mit QVZ in Stop, falls die Kachelnummer der CP nicht mit der in diesem Baustein parametrierten Nummer übereinstimmt.

Ablage der Parameter in einem DB bei indirekter Parametrierung:

	DL DR
A-NR zeigt auf den Beginn	INSS
	A-NR
	ANZW
	ZT/N
	ZANF
	ZLAE

Hinweis

PAFE ist nicht indirekt parametrierbar.

Bei allen anderen Formaloperanden (SS, ANZW, ZT/N, ZANF, ZLAE) kann 0 parametriert werden, da diese bei indirekter Parametrierung nicht ausgewertet werden.

Ablage der Daten in einem DB:

ZANF zeigt auf den Beginn

DL	DR
A-Nr	F-Nr
Anz. der gelesenen Worte	
ab hier liegen die von der CP gelesenen Daten	

1.2.5.5 Parametrierung der Dateizugriffe über Handles

Der COM-Treiber unterstützt den vollen Zugriff auf alle Laufwerke und Verzeichnisse des CPs. Wie von MS-DOS her bekannt können bei Dateizugriffen Laufwerks- und ein Verzeichnisname angegeben werden.

Bei Dateienzugriffen werden unter MS-DOS Nummern verwendet, die sogenannten "Handles". Die Anzahl der verfügbaren Handles und damit die maximale Anzahl an gleichzeitig offenen Dateien ist in der Datei CONFIG.SYS durch den Eintrag `FILES=n` festgelegt. `n` sollte mindestens den Wert 20 haben.

Mit Hilfe von Handles können Sie nicht nur auf Massenspeicher, wie Festplatte, RAM-Disk oder Diskette zugreifen, sondern Sie haben auch die Möglichkeit mit Handles auf Geräte (Devices), wie Drucker und serielle Schnittstellen zuzugreifen. Für eine Reihe von Devices sind bereits Standard-Handles vordefiniert. Vor einem Zugriff auf diese Geräte ist es nicht mehr erforderlich diese zu öffnen. Mit diesen Funktionen können Sie aus dem AG heraus unmittelbar auf Drucker, Bildschirm und Tastatur des CPs zugreifen.

Handle	Device	Zugriffsmodus
00	Standard-Input (Tastatur)	nur lesen
01	Standard-Output (Bildschirm)	nur schreiben
02	Standard-Error (auch Bildschirm)	nur schreiben
03	Standard-Auxiliary (V24-Schnittstelle)	lesen und schreiben
04	Standard-Printer (Auxiliary)	nur schreiben

Tab. 1-7: vordefinierte Standard-Handles

1.2.5.6 Dateinamen

Wie bereits erwähnt, können Dateinamen auch mit einer Laufwerks- und Pfadbezeichnung angegeben werden. Die CP-Software schränkt dies nicht ein. Verzeichnisse müssen wie üblich mit dem Backslash-Zeichen \ (ASCII-Code 92, 5Ch) versehen werden. Das Zeichen kann auch über das PG korrekt eingegeben werden.

Bitte beachten Sie folgendes:

- Die maximale Länge einer Pfad-Angabe darf maximal 64 Zeichen betragen.
- Die Bezeichnung einer Datei darf maximal aus drei Zeichen Laufwerksangabe, bis zu 64 Zeichen Pfadangabe, sowie 8 Zeichen Dateiname und 3 Zeichen Erweiterung (Extension) bestehen (insgesamt maximal 78 Zeichen).

Es ist ebenso möglich, für Dateinamen eine feste Maske aus 8 Zeichen, einem Punkt und 3 Zeichen Extension zu verwenden. Ist der Name kürzer als 8 Zeichen, oder hat die Erweiterung weniger als 3 Zeichen, so können die unbenutzten Stellen mit Leerzeichen (ASCII-Code 32, 20h) belegt werden.

Bei MS-DOS System-Funktionen, die einen Dateinamen als Parameter erfordern, muß der Name mit dem Zeichen ASCII-Null (ASCII-Code 0) abgeschlossen werden. Wird ein Dateiname ohne abschließende ASCII-Null angegeben, so ergänzt die CP-Software die fehlende ASCII-Null.

Die Hantierungsbausteine können nur einen Block von Worten in die Kachel eintragen, d.h. es können nur Dateinamen mit einer geraden Länge eingetragen werden.

Wenn Sie Dateinamen mit ungerader Länge haben, müssen Sie eine ASCII-Null anfügen.

1.2.6 Beschreibung der Funktionen

1.2.6.1 Alle Laufwerke zurücksetzen (Disk Reset)

Mit dieser Funktion werden alle geänderten und noch nicht gesicherten Datei-Puffer physikalisch auf die Laufwerke geschrieben.

Parametrierung des FB3: F-Nr 13 (0Dh)

1.2.6.2 Laufwerk anwählen (Select Disk)

Parametrierung des FB3: F-Nr 14 (0Eh)
DOSP Nummer des gewünschten Laufwerks

Laufwerksnummer	0	A:
	1	B:
	2	C:

Bei dieser Funktion entspricht Laufwerksnummer 0 dem Laufwerk A:, im Gegensatz zu anderen Funktionen wie "get current directory".

1.2.6.3 Laufwerk bestimmen (Get Disk)

Die Funktion liefert die Nummer des aktuellen (angemeldeten) Laufwerkes zurück. In Byte 6 wird der entsprechende Laufwerks-Buchstabe also "A", "C", usw. hinterlegt.

Parametrierung des FB5: F-Nr 25 (19h)

Parametrierung des FB6: ZT/N Nr. des DBs für die Laufwerksdaten
ZANF Lage des Datenwortes im DB
ZLAE Länge der Laufwerksdaten im DB in Worten (1)

Inhalt des DBs: DW1 A-NR F-Nr
DW2 Anzahl der gelesenen Worte
DW3 Laufwerks-Buchstabe Laufwerksnummer

A:	0
B:	1
C:	2

Bei dieser Funktion entspricht Laufwerksnummer 0 dem Laufwerk A:, im Gegensatz zu anderen Funktionen wie "get current directory".

1.2.6.4 Verzeichnis anlegen (Create Directory)

Parametrierung des FB3:	F-Nr	57 (39h)
	QT/N	Nr. des DBs mit dem Verzeichnisnamen
	QANF	Lage des Verzeichnisnamens im DB
	QLAE	Länge des Verzeichnisnamens im DB in Worten
Inhalt des DBs:	DW1	Verzeichnisname
	DW2	...
	DW3	...
	...	

Hinweis

Bitte beachten Sie, daß der Verzeichnisname bei ungerader Länge mit einem 0-Byte zu terminieren ist. Bei gerader Länge ist dies nicht erforderlich.

1.2.6.5 Verzeichnis löschen (Delete Directory)

Parametrierung des FB3:	F-Nr	58 (3Ah)
	QT/N	Nr. des DBs mit dem Verzeichnisnamen
	QANF	Lage des Verzeichnisnamens im DB
	QLAE	Länge des Verzeichnisnamens im DB in Worten
Inhalt des DBs:	DW1	Verzeichnisname
	DW2	...
	DW3	...
	...	

Hinweis

Bitte beachten Sie, daß der Verzeichnisname bei ungerader Länge mit einem 0-Byte zu terminieren ist. Bei gerader Länge ist dies nicht erforderlich.

Durch Angabe eines Laufwerkes im Verzeichnisnamen kann auch auf einem nicht-angemeldeten Laufwerk ein Verzeichnis gelöscht werden.

Diese Funktion wird mit Fehler beendet, wenn das angegebene Verzeichnis das aktuelle Verzeichnis ist, oder wenn das angegebene Verzeichnis Dateien enthält.

1.2.6.6 Verzeichnis wechseln (Set Current Directory)

Parametrierung des FB3:	F-Nr	59 (3Bh)
	QT/N	Nr. des DBs mit dem Verzeichnisnamen
	QANF	Lage des Verzeichnisnamens im DB
	QLAE	Länge des Verzeichnisnamens im DB in Worten
Inhalt des DBs:	DW1	Verzeichnisname
	DW2	...
	DW3	...
	...	

Hinweis

Bitte beachten Sie, daß der Verzeichnisname bei ungerader Länge mit einem 0-Byte zu terminieren ist. Bei gerader Länge ist dies nicht erforderlich.

Mit dieser Funktion können Sie das angemeldete (aktuelle) Laufwerk nicht wechseln. Das Verzeichnis kann zwar mit einer Laufwerks-Angabe versehen werden, das aktuelle Verzeichnis auf dem angemeldeten Laufwerk bleibt aber auf dem vorherigen Wert eingestellt. Erst wenn auf das angegebene Laufwerk zugegriffen wird, z.B. mit der Funktion "Select Disk" wird das gewünschte Laufwerk eingestellt.

1.2.6.7 Aktuelles Verzeichnis bestimmen (Get Current Directory)

Parametrierung des FB5:	F-Nr	71 (47h)
	DOSP	Laufwerksnummer
Parametrierung des FB6:	ZT/N	Nr. des DBs mit dem Verzeichnisnamen
	ZANF	Lage des Verzeichnisnamens im DB
	ZLAE	Länge des Verzeichnisnamens im DB in Worten
Inhalt des DBs:	DW1	A-NR F-Nr
	DW2	Anzahl der gelesenen Worte
	DW3	Verzeichnisname
	DW4	...
	...	

1.2.6.8 Datei erzeugen (Create File/Rewrite Existing File)

Parametrierung des FB3:	F-Nr	60 (3Ch)
	DOSP	Attribut der neuen Datei
	QT/N	Nr. des DBs mit dem Dateinamen
	QANF	Lage des Dateinamens im DB
	QLAE	Länge des Dateinames im DB in Worten
Inhalt des DBs:	DW1	Dateiname
	DW2	...
	DW3	...
	...	
Parameter:		
Attribut:	00	normal
	01	read only
	02	hidden
	04	system

Datei-Attribute können aufaddiert werden:
z.B. Attribut 03 → die Datei hat die Attribute "read-only" und "hidden".

Rückgabe des FB3:	DOSP	Handle der neuen Datei

Hinweis

Falls eine Datei mit dem angegebenen Namen bereits existiert, wird diese auf Länge Null abgeschnitten, d.h. alle vorhandenen Daten werden gelöscht.

1.2.6.9 Neue Datei erzeugen (Create New File)

Parametrierung des FB3:	F-Nr	90 (5Ah)
	DOSP	Attribut der neuen Datei
	QT/N	Nr. des DBs mit dem Dateinamen
	QANF	Lage des Dateinamens im DB
	QLAE	Länge des Dateinames im DB in Worten
Inhalt des DBs:	DW1	Dateiname
	DW2	...
	DW3	...
	...	
Parameter:		
Attribut:	00	normal
	01	read-only
	02	hidden
	04	system
	Datei-Attribute können aufaddiert werden:	
	z.B. Attribut 03 → die Datei ist read-only und hidden.	
Rückgabe des FB3:	DOSP	Handle der neuen Datei

Hinweis

Falls eine Datei mit dem angegebenen Namen bereits existiert, wird diese auf Länge Null abgeschnitten, d.h. alle vorhandenen Daten werden gelöscht.

1.2.6.10 Datei öffnen (Open File)

Parametrierung des FB3:	F-Nr	61 (3Dh)
	DOSP	Zugriffsmodus
	QT/N	Nr. des DBs mit dem Dateinamen
	QANF	Lage des Dateinamens im DB
	QLAE	Länge des Dateinames im DB in Worten
Inhalt des DBs:	DW1	Dateiname
	DW2	...
	DW3	...
	...	
Parameter:		
Zugriffsmodus:	00	Datei öffnen zum Lesen
	01	Datei öffnen zum Schreiben
	02	Datei öffnen zum Lesen und Schreiben

Rückgabe des FB3: DOSP Handle der neuen Datei

Hinweis

Nach dem Öffnen der Datei kann der Zugriffsmodus nicht mehr geändert werden, erst nach einem Schließen und erneuten Öffnen kann ein anderer Zugriffsmodus angewendet werden. Netz-Zugriffe sind bei dieser Funktion möglich, aber nicht berücksichtigt. (SHARE.EXE muß geladen sein)

1.2.6.11 Datei physikalisch auf Disk schreiben (Commit File)

Diese Funktion sorgt dafür, daß alle geänderten internen Daten-Puffer einer Datei auf der CP physikalisch auf das Laufwerk übertragen werden und im Inhalts-Verzeichnis Datum und Uhrzeit der letzten Änderung, sowie die Größe der Datei aktualisiert werden. Diese Funktion ist äquivalent dem Schließen und erneuten Öffnen der Datei.

Parametrierung des FB3: F-Nr 104 (68h)
DOSP Handle-Nummer der zu schreibenden Datei

Diese Funktion überträgt keine Daten aus dem AG zur CP.

1.2.6.12 Datei schließen (Close File)

Parametrierung des FB3:	F-Nr	62 (3Eh)
	DOSP	Handle-Nummer der zu schließenden Datei

Diese Funktion überträgt keine Daten aus dem AG zur CP.

1.2.6.13 Datei löschen (Delete File)

Diese Funktion löscht eine Datei auf einem Laufwerk der CP. Vor dem Löschen braucht die Datei nicht geöffnet zu sein. Es ist sogar möglich eine gerade geöffnete Datei ohne Fehlermeldung zu löschen. Der Anwender muß selbst dafür Sorge tragen, daß keine Dateien, auf die gerade zugegriffen wird, gelöscht werden. Bei Netzwerken wird diese Aufgabe von der Netzwerk-Management-Software durchgeführt.

Parametrierung des FB3:	F-Nr	65 (41h)
	QT/N	Nr. des DBs mit dem Dateinamen
	QANF	Lage des Dateinamens im DB
	QLAE	Länge des Dateinames im DB in Worten
Inhalt des DBs:	DW1	Dateiname
	DW2	...
	DW3	...
	...	

1.2.6.14 Datei umbenennen (Rename File)

Parametrierung des FB3:	F-Nr	86 (56h)
	QT/N	Nr. des DBs mit den Filenamen
	QANF	Lage der Filenamen im DB
	QLAE	Länge der Filenamen im DB in Worten
Inhalt des DBs:	DW1	ursprünglicher Dateiname, Nullzeichen, neuer
	DW2	Dateiname
	DW3	...
	...	

Vor dem Umbenennen braucht die Datei nicht geöffnet zu werden.

Als Daten sind beide Datei-Namen aneinandergehängt zu übergeben, zuerst der ursprüngliche Datei-Name, dann der neue Datei-Name. Die beiden Namen müssen durch mindestens ein ASCII-Null Zeichen getrennt sein. Als Datenlänge ist die Länge der beiden Datei-Namen einschließlich aller Null-Zeichen anzugeben.

Sie können diese Funktion auch zum Verschieben von Dateien in ein anderes Verzeichnis verwenden (move file). Geben Sie hierzu den Namen des gewünschten Ziel-Verzeichnisses bei dem neuen Datei-Namen an.

Hinweis

Bitte beachten Sie, daß Sie eine Datei nur innerhalb eines Laufwerkes verschieben können.

1.2.6.15 Datei-Zeiger setzen (Set File Pointer)

Parametrierung des FB3:	F-Nr	66 (42h)
	DOSP	POS (höherwertiges Byte), Handle (niederw. Byte)
	QT/N	Nr. des DBs mit dem Dateizeiger
	QANF	Lage des Dateizeigers im DB
	QLAE	Länge des Datensatzes (2 Worte)
Inhalt des DBs:	DW1	höherwertiges Wort des Dateizeigers
	DW2	niederwertiges Wort des Dateizeigers
Parameter:		
POS:	0	abs. Position von Datei-Anfang
	1	rel. Position ab der aktuellen Position (vorzeichenbehaftet)
	2	rel. Position vom Datei-Ende (vorzeichenbehaftet.)

Mit dieser Funktion kann die Länge einer Datei ermittelt werden, wenn als Funktions-Code 02 und als neue relative Position vom Datei-Ende 0 angegeben wird. Anschließend kann mit "get file pointer" die Position erhalten werden, die gleichzeitig die Datenanzahl ist.

Hinweis

Der Wert des Datei-Zeigers ist immer als die Angabe einer Byte-Position zu betrachten.

1.2.6.16 Datei-Zeiger lesen (Get File Pointer)

Parametrierung des FB5:	F-Nr	194 (C2h)
	DOSP	Handle
Parametrierung des FB6:	ZT/N	Nr. des DBs für die gewünschten Daten
	ZANF	Ziel-Position im DB
	ZLAE	2
Inhalt des DBs:	DW1	A-NR F-Nr
	DW2	Anzahl der gelesenen Worte
	DW3	höherwertiges Wort des Dateizeigers
	DW4	niederwertiges Wort des Dateizeigers

Der Datei-Zeiger wird als ein Doppelwort zurückgegeben, daher ist als Datenanzahl 2 anzugeben.

Hinweis

Der Wert des Datei-Zeigers ist immer als die Angabe einer Byte-Position zu betrachten.

1.2.6.17 Von Datei/Device lesen (Read File or Device)

Parametrierung des FB5:	F-Nr	63 (3Fh)
	DOSP	Handle der Datei
Parametrierung des FB6:	ZT/N	Nr. des DBs für die zu lesenden Daten
	ZANF	Ziel-Position im DB
	ZLAE	Anzahl der zu lesenden Datenworte (2)
Inhalt des DBs:	DW1	A-NR F-Nr
	DW2	Anzahl der gelesenen Worte
	DW3	Datenwort 1
	DW4	Datenwort 2
	...	

Die Anzahl der Worte, die von der Datei gelesen werden sollen, darf nicht größer als 504 sein, sonst wird die Funktion mit Fehler abgebrochen.

Bei dieser Funktion wird keine Vertauschung von Bytes in einem Datenwort oder Doppelwort vorgenommen. Alle Daten werden unverändert aus der CP in das AG übertragen. Da in den meisten Fällen mit ASCII-Dateien gearbeitet wird, ist eine Vertauschung nicht erforderlich ist. Bei Bedarf muß die Vertauschung je nach Anforderung auf der CP oder SPS-Seite durchgeführt werden.

1.2.6.18 Auf Datei/Device schreiben (Write File or Device)

Parametrierung des FB3:	F-Nr	64 (40h)
	DOSP	Handle der Datei
	QT/N	Nr. des DBs mit den zu schreibenden Daten
	QANF	Position der Daten im DB
	QLAE	Länge des zu schreibenden Datensatzes in Worten

Die Anzahl der Worte, die auf die Datei geschrieben werden sollen, darf nicht größer als 504 sein, sonst wird die Funktion mit Fehler abgebrochen.

Bei dieser Funktion wird keine Vertauschung von Bytes in einem Datenwort oder Doppelwort vorgenommen. Alle Daten werden unverändert aus dem AG in die CP übertragen. Da in den meisten Fällen mit ASCII-Dateien gearbeitet wird, ist eine Vertauschung nicht erforderlich ist. Bei Bedarf muß die Vertauschung je nach Anforderung auf der CP oder SPS-Seite durchgeführt werden.

Wenn die Funktion fehlerfrei beendet wurde, aber die geschriebene Anzahl kleiner als die gewünschte Anzahl ist, dann kann während der Ausführung ein partieller Schreib-Fehler aufgetreten sein, oder auf ein Character-Device (Standard-Ausgabe) wurde das Zeichen ^Z ASCII-Code 26, 1Ah geschrieben.

1.2.6.19 Datum lesen (Get Date)

Parametrierung des FB5:	F-Nr	42 (2Ah)	
Parametrierung des FB6:	ZT/N	Nr. des DBs für das zu lesende Datum	
	ZANF	Ziel-Position im DB	
	ZLAE	Anzahl der zu lesenden Datenworte (3)	
Inhalt des DBs:	DW1	A-NR	F-Nr
	DW2	Anzahl der gelesenen Worte	
	DW3	Jahr	
	DW4	Monat	Tag
	DW4	Wochentag	-----

Parameter:

Jahr	1980 ... 2099
Monat	1 ... 12
Tag	1 ... 31
Wochentag	0 ... 6, (0 = Sonntag, 1 = Montag, ...)

1.2.6.20 Uhrzeit lesen (Get Time)

Parametrierung des FB5:	F-Nr	44 (2Ch)	
Parametrierung des FB6:	ZT/N	Nr. des DBs für die zu lesende Uhrzeit	
	ZANF	Ziel-Position im DB	
	ZLAE	Anzahl der zu lesenden Datenworte (2)	
Inhalt des DBs:	DW1	A-NR	F-Nr
	DW2	Anzahl der gelesenen Worte	
	DW3	Stunden	Minuten
	DW4	Sekunden	Hundertstel-Sekunden

Parameter:

Stunde	0 ... 23
Minute	0 ... 59
Sekunde	0 ... 59
Hundertstel-Sekunde	0 ... 99

Die Funktion gibt nach einer fehlerfreien Beendigung als Datenanzahl 2 Worte zurück. Alle Werte sind als Bytes zu interpretieren.

1.2.6.21 Programm aufrufen (Program Execute)

Über diese Funktion können direkte Kommandos an die CP übergeben werden.

Parametrierung des FB3:	F-Nr	75 (4Bh)
	QT/N	Nr. des DBs mit der MS-DOS-Kommandozeile
	QANF	Position der Kommandozeile im DB
	QLAE	Länge der Kommandozeile im DB in Worten

Diese Funktion kann momentan nur aufgerufen werden, wenn neben dem COM-Treiber und anderen residenten Utilities kein Programm auf der CP läuft. Der COM-Treiber CP386COM.EXE muß dazu im nicht-residenten Betrieb (Option/NOTSR beim Aufruf von CP386COM.EXE) gestartet werden.

Mit dem Beenden des aufgerufenen Programms wird auch diese Funktion beendet.

Die Kachel bleibt während des Programmlaufes für andere Aufträge gesperrt. Beachten Sie dies beim Aufruf anderer Funktionen!

1.2.6.22 MS-DOS Version bestimmen (Get MS-DOS Version)

Parametrierung des FB5:	F-Nr	48 (30h)
Parametrierung des FB6:	ZT/N	Nr. des DBs für die zu lesenden Daten
	ZANF	Ziel-Position im DB
	ZLAE	Anzahl der zu lesenden Datenworte (3)

Inhalt des DBs:	DW1	A-NR	F-Nr
	DW2	Anzahl der gelesenen Worte	
	DW3	Hauptnummer	Unternummer
	DW4	OEM-Nummer	Anwendernummer
	DW5	Seriennummer	

Die Funktion gibt nach einer fehlerfreien Beendigung als Datenanzahl 3 Worte zurück. In Datenbyte 4 wird die Versions-Hauptnummer, in Datenbyte 5 ist die Versions-Unternummer, und in Datenbyte 6 ist die OEM-Kennung eingetragen. In den Bytes 7 bis 9 wird eine 24 Bit Anwender-Seriennummer zurückgegeben. Davon ist das höchstwertige Byte in Byte 7 und das niederwertigste Byte in Byte 9 abgelegt.

1.2.6.23 Ausführliche Fehler-Information besorgen

Parametrierung des FB5: F-Nr 89 (59h)

Parametrierung des FB6:

	ZT/N	Nr. des DBs für die zu lesenden Daten
	ZANF	Ziel-Position im DB
	ZLAE	Anzahl der zu lesenden Datenworte (3)

Inhalt des DBs:

	DW1	A-NR F-Nr
	DW2	Anzahl der gelesenen Worte
	DW3	Fehlercode
	DW4	Fehlerklasse Maßnahme
	DW5	Fehlerort ------

Die Funktion gibt nach einer fehlerfreien Beendigung MSDOS-Fehlercodes im Datensatz zurück. In Datenbyte 4 und 5 wird der Fehlercode, in Datenbyte 6 die Fehlerklasse, in Datenbyte 7 die empfohlene Maßnahme und in Datenbyte 8 der Fehlerort eingetragen.

MS-DOS-Fehlercode-Tabellen

Fehlernummer		Beschreibung
dez	hex	
01		ungültige Funktionsnummer
02		Datei nicht gefunden
03		Pfad (Verzeichnis) nicht gefunden
04		zu viele offene Files, Abhilfe: Die Anzahl der Files in CONFIG.SYS erhöhen
05		Zugriff wurde verweigert
		Versuch eine schreibgeschützte Datei zu verändern
06		ungültiges Handle, zu dem angegebenen Handle gibt es keine offene Datei
07		Speicher-Kontrollblöcke zerstört
		MS-DOS nicht mehr lauffähig, System muß neu gebootet werden
08		kein Speicher mehr vorhanden
09		ungültiger Speicherkontrollblock
10	(0Ah)	ungültiges Environment
11	(0Bh)	ungültiges Format eines Programmes. Das Programm ist nicht korrekt aufgebaut oder die Datei enthält kein Programm
12	(0Ch)	ungültiger Zugriffscode, falscher Zugriffsmodus beim Datei-öffnen angegeben
13	(0Dh)	ungültige Daten
14	(0Eh)	ungültige Einheit
15	(0Fh)	ungültiges Laufwerk, es wurde versucht, ein nicht existierendes Laufwerk anzusprechen
16	(10h)	ungültiges Kommando
17	(11h)	nicht das gleiche Device

Fehlernummer dez	Fehlernummer hex	Beschreibung
18	(12h)	keine Files können mehr angelegt werden, Inhalts-Verzeichnis ist voll
19	(13h)	Diskette ist schreibgeschützt
20	(14h)	unbekannte Einheit
21	(15h)	Laufwerk nicht bereit. Keine Diskette eingelegt
22	(16h)	unbekanntes Kommando
23	(17h)	Datenfehler (CRC-Error) Prüfsumme des Disketten-/Plattensektors falsch; Sektor vermutlich defekt
24	(18h)	falsche Länge der Request-Struktur
25	(19h)	Seek-Fehler, Positionier-Fehler, Datei-Zeiger wurde über das Ende der Datei hinaus positioniert.
26	(1Ah)	unbekannter Media-Typ, (Diskette ist nicht im MS-DOS Format aufgebaut)
27	(1Bh)	Sektor nicht gefunden.
28	(1Ch)	Drucker meldet Papier-Ende
29	(1Dh)	Schreib-Fehler.
30	(1Eh)	Lese-Fehler.
31	(1Fh)	allgemeiner Fehler
32	(20h)	Verletzung bei gemeinsamen Zugriff
33	(21h)	Verletzung bei File-Locking
34	(22h)	ungültiger Disketten-Wechsel
35	(23h)	FCB nicht verfügbar
36	(24h)	Buffer für gemeinsamen Datei-Zugriff überschritten
80	(50h)	Datei existiert bereits
82	(52h)	Verzeichnis kann nicht angelegt werden
83	(53h)	Fehler bei Int 24 (Behandlung von kritischen Fehlern)
112	(70h)	Längenfehler, falsche Datenanzahl (z.B. Versuch mehr als 504 Worte zu lesen oder zu schreiben)
113	(71h)	Zeit-Überschreitung bei Kommunikation Für ca. 10 sec konnte die CP keinen Zugriff auf die Kachel erhalten.

Tab. 1-8: MS-DOS-Fehlercodes mit Beschreibung

Nr.	Fehlerklassen
01h	keine Resourcen mehr verfügbar (Speicher oder Handles)
02h	kein Fehler, sondern momentaner Zustand (gesperrte Region in einem File), der voraussichtlich wieder verschwindet.
03h	Autorisierungsproblem
04h	interner Fehler in der System-Software
05h	Hardware-Fehler
06h	System-Software-Fehler, kein Fehler des aktiven Prozesses (wie fehlende Konfigurationsdateien)
07h	Anwendungsprogramm-Fehler
08h	Datei oder Element nicht gefunden
09h	Datei oder Element hat einen fehlerhaften Typ oder Format
0Ah	Datei oder Element vor Zugriff gesperrt
0Bh	falsche Diskette im Laufwerk, fehlerhafte Datensektoren oder Fehler des Speichermediums
0Ch	sonstiger Fehler

Tab. 1-9: MS-DOS-Codes für Fehlerklassen

Nr.	Maßnahme
01h	Funktion mehrmals wiederholen. Danach den Anwender fragen, ob abgebrochen oder der Fehler ignoriert werden soll.
02h	Funktion mehrmals wiederholen, mit Zeitverzögerung zwischen den einzelnen Versuchen. Anschließend den Anwender fragen, ob abgebrochen oder der Fehler ignoriert werden soll.
03h	die richtige Information von Benutzer eingeben lassen (i.d.R. durch einen falschen Datei-Namen oder Laufwerksangabe verursacht).
04h	Die Anwendung ordentlich abbrechen (offene Files schließen, File-Locking aufheben)
05h	Die Anwendung sofort abbrechen, ohne 'Aufräumen'.
06h	Fehler ignorieren.
07h	Wiederholen, nachdem Anwender aufgefordert wurde den Fehler zu beseitigen.

Tab. 1-10: MS-DOS-Codes für die empfohlenen Maßnahmen

Nr.	Fehler-Ort
01h	unbekannt
02h	Block-Device oder Laufwerks-Emulation (RAM-Disk)
03h	Netzwerk
04h	Serielles Device
05h	Speicher

Tab. 1-11: MS-DOS-Codes für die Fehler-Orte

1.2.6.24 Beliebiger Interrupt (General Interrupt)

Mit dieser Funktion haben Sie die Möglichkeit beliebige Interrupts des CPs aufzurufen, wie z.B. VGA-BIOS-Interrupts, Keyboard-Interrupt, Maus-Interrupt etc. Aufgrund der vielfältigen Parametrier-Möglichkeiten ist es nicht möglich, alle Register mit Parametern zu belegen.

Dieser Funktion werden 4 Datenworte übergeben, die entsprechend in die Register AX, BX, CX und DX geladen werden. Die Nummer des Interrupts ist in DOSP-Parameter abzulegen. Es sind alle Interrupt-Nummern zugelassen.

Nach der Ausführung der Funktion wird der im Register AX zurückgegebene Wert im DOSP-Parameter eingetragen.

Parametrierung des FB3:	F-Nr	255 (FFh)
	DOSP	Interrupt-Nummer
	QT/N	Nr. des DBs mit den zu schreibenden Daten
	QANF	Position der Daten im DB
	QLAE	Länge des zu schreibenden Datensatzes in Worten
Inhalt des DBs:	DW1	Datenwort für Register AX
	DW2	Datenwort für Register BX
	DW3	Datenwort für Register CX
	DW4	Datenwort für Register DX
Rückgabe des FB3:	DOSP	Datenwort aus dem Register AX

Dieser Aufruf sollte nur von erfahrenen DOS-Programmierern verwendet werden, da dieser Aufruf beliebige DOS-Zugriffe ermöglicht.

1.3 CP-Aufträge für die SPS (Funktionen für Kachel 2, 3 und 7)

1.3.1 Übersicht

Das Treiberprogramm CP386COM bedient die Aufträge von der SPS und vom CP. So stellt der Treiber für die Kacheln 2, 3 und 7 eine Reihe von Funktionen zur Verfügung. Mit diesen Funktionen können, aus einem auf der CP laufenden Anwenderprogramm heraus, Daten aus dem AG gelesen bzw. in das AG geschrieben werden. Alle Funktionen werden mit dem Software-Interrupt 78h aufgerufen.

Die Übergabe und die Rückgabe von Parametern erfolgt beim Aufruf eines Interrupts ausschließlich in den Prozessor-Registern. Die Belegung der Register ist in der Beschreibung der Funktionen enthalten. Für Turbo-Pascal, Turbo-C, C++ und Microsoft-C sind Schnittstellen implementiert. Es gibt Funktionen zum Lesen und Schreiben von Daten in das AG, zur Status-Abfrage und Abbruch-Funktionen. Die Funktionen für die Datenübertragung werden anhand ihrer Übertragungsart unterschieden. Man unterscheidet Einzelelementübertragung sowie Blockübertragung. Die einzelnen Funktionen werden als "Aufträge" abgewickelt. Bei dem Aufruf einer Funktion wird eine Auftragsnummer zurückgegeben, unter der der Status der Bearbeitung abgefragt werden kann. In Kachel 2 und 3 können pro Kachel bis zu 127 Aufträge gleichzeitig in Bearbeitung sein.

Folgende Funktionen stehen zur Verfügung:

verwendete Kachel	Funktions-Nr.	Funktion
keine	00h	Status-Abfrage
2	21h	Lesen eines Einzelelementes aus dem AG
2	21h	Lesen eines Blockes aus dem AG
2	20h	Statusabfrage zu Lese-Auftrag
2	28h	Abbrechen aller Lese-Aufträge
3	31h	Schreiben einer Einzelelementes in das AG
3	31h	Schreiben eines Blockes in das AG
3	30h	Statusabfrage zu Schreib-Auftrag
3	38h	Abbrechen aller Schreib-Aufträge
7	70h	Statusabfrage für Prozeßabbild
7	71h	Lesen eines Bereiches des Prozeßabbildes

Tab. 1-12: Funktionen

1.3.2 Installation der Software zur Kopplung von SPS und CP

Für die Kommunikation mit der CP müssen die Hantierungsbausteine FB1 und FB2 in der SPS geladen werden. Der Hantierungsbaustein FB1 wird im OB1 aufgerufen, der Hantierungsbaustein FB2 in den Neustartbausteinen (OB20, OB21 und OB22).

1.3.2.1 FB1 (CP-L/S), CP lesen und schreiben

Mit FB1 können Sie Daten zwischen CP und AG transferieren. Mit dem Aufruf von FB1 sind folgende Parameter zu übergeben:

Bez.	Format	Beschreibung
ANSS	KY	Anzahl der Aufträge
PAA	KF	Kennung Prozeßabbild
PAFE	MB	Merkerbyte für Fehlermeldungen

Tab. 1-13: Parameterliste für den Aufruf von FB1

Für die o.g. Parameter sind im einzelnen die folgenden Angaben erforderlich:

ANSS

AN Anzahl der Aufträge, die maximal bei einem Hantierungsbaustein-Aufruf auf der Kachel abgearbeitet werden sollen.

SS Nummer der Basiskachel

PAA

Kennung der Prozeßabbilder beim Aufruf des Hantierungsbausteins auf der Kachel aktualisieren

- ≠ 0 Prozeßabbilder werden aktualisiert
- = 0 Prozeßabbilder werden nicht aktualisiert

PAFE

Fehlerrückmeldung des Hantierungsbausteins

- = 0 es ist kein Fehler aufgetreten
- ≠ 0 es ist ein Fehler aufgetreten. Fehlernummer im PAFE-Byte:
  - 1 Anzahl der Aufträge, die maximal bei einem Hantierungsbaustein-Aufruf abgearbeitet werden sollen, ist 0.
  - 2 Anzahl der Aufträge, die maximal bei einem Hantierungsbaustein-Aufruf abgearbeitet werden sollen, ist größer als 127.
  - 3 Die Basiskachelnummer ist nicht durch 8 teilbar
  - 5 Die Kachel ist noch nicht von der CP synchronisiert.
  - 6 Bei einem Blockauftrag als ersten Aufruf darf kein weiterer Auftrag mehr in der Kachel stehen.
  - 7 Ein weiterer Blockauftrag darf nur als erster Auftrag in der Kachel stehen.

1.3.2.2 FB2 (SYNCHRON), Synchronisation CP und AG

FB2 schreibt ein Synchronisationsmuster auf die Kachel. Mit dem Aufruf von FB2 sind folgende Parameter zu übergeben:

Bez.	Format	Beschreibung
SSNR	KF	Nummer der Basiskachel
WART	KF	Art der Synchronisation
PAFE	MB	Merkerbyte für Fehlermeldungen

Tab. 1-14: Parameterliste für den Aufruf von FB2

Für die o.g. Parameter sind im einzelnen die folgenden Angaben erforderlich:

SSNR Nummer der Basiskachel

WART
- = 0 Der FB-SYNCHRON wartet nicht darauf bis die CP jede einzelne Kachel synchronisiert hat.
- ≠ 0 Der FB-SYNCHRON wartet bei jeder einzelnen Kachel bis die CP diese Kachel synchronisiert hat.

PAFE Fehlerrückmeldung des Hantierungsbausteins
- = 0 es ist kein Fehler aufgetreten
- ≠ 0 es ist ein Fehler aufgetreten:
  - 3 Basiskachelnummer ist nicht durch 8 teilbar.

1.3.3 Treiber-Funktionen über Software-Interrupt

1.3.3.1 CP-Status Abfrage

Diese Funktion gibt verschiedene allgemeine Status-Informationen über die CP zurück. Mit dieser Funktion können Sie ermitteln, ob der CP Software-Treiber geladen ist. Weitere Informationen, die zurückgegeben werden, sind der Ausgabestand von Hardware und Software, CPU-Kennung usw.

Register	**IN**	high	**OUT**	low
AX	$00	$C386		
BX		VGA-Bios	Bios	
CX		CP	CPU	
DX		CP-Status	AG-Status	

AX C386h Kennung, daß CP-Software geladen ist.
BH Ausgabestand des CP VGA-BIOS
BL Ausgabestand des CP-BIOS
CH Ausgabestand der Treiber-Software (CP)
CL Kennung der CPU im AG (erst gültig, wenn Kacheln synchronisiert sind)
DH CP-Statusregister (IO-Adresse 280h)
DL AG-Statusregister (IO-Adresse 281h)

Die Kennungen der Ausgabestände (Versionsnummern) für BIOS, VGA und Treiber sind in einem Byte BCD-kodiert. Das heißt der Wert 10h entspricht Version 1.0; 15h entspricht Versionsnummer 1.5 und 1Ah entspricht Version 1.10.

Diese Funktion führt keine Initialisierungen an den Kacheln aus.

1.3.3.2 Lesen eines Einzelelements aus dem AG

Mit dieser Funktion kann ein einzelnes Datum (Bit, Byte, Wort ...) aus dem AG gelesen werden. Die Funktion startet den Auftrag nur, wartet nicht bis das AG ihn bearbeitet, sondern kehrt sofort wieder zum Aufrufer zurück. Die Daten selbst können daher erst durch einen Aufruf der Funktion "Status-Abfrage" gelesen werden (siehe Kapitel 1.3.4.6).

Register	high	**In**	low	**Out**
AX	$21		typ	status
BX	size		bst	
CX	adr			
DX	-		bit	

Parameter

typ		Elementtyp der Daten bei Einzelelement im AG (DB, MB), siehe Kapitel 1.3.4.10.2.
size		Kennung der Elementgröße (Bit, Byte, ...), siehe Kapitel 1.3.4.10.1.
bst		Bausteinnummer nur bei Elementtyp DB oder DX. Bei Elementtyp "absolut" sind hier die höchstwertigen Bits von adr.
adr		Anfangsadresse im Bereich.
bit		Bit-Nummer wenn die Elementgröße (size) Bit oder Semaphor ist.
status	< 0	Fehlernummer, weil Fehler aufgetreten. Fehlernummern des AG werden mit FF00h aufaddiert
	1..127	Auftragsnummer unter der der Status abgefragt werden kann

Hinweis

Diese Funktion gibt noch keine Daten zurück! Wurde der Auftrag "Fertig ohne Fehler", so können die Daten durch einen Aufruf der Funktion Statusabfrage abgeholt werden.

Um sicherzustellen, daß ein fertiger Lese-Auftrag nicht durch einen neuen Auftrag überschrieben wird, bevor die Daten abgeholt werden, wird der Auftrag blockiert. Nach dem Starten eines Auftrages muß dessen Status solange abgefragt werden, bis der Auftrag fertig mit oder ohne Fehler ist. Ist der Auftrags-Status "Fertig ohne Fehler" so werden die Daten an die angegebene Adresse im CP kopiert. Erfolgt keine Statusabfrage bleibt der Auftrag blockiert, und es können unter Umständen keine weiteren Lese-Aufträge mehr gestartet werden, selbst wenn alle Aufträge in der Kachel beendet sind.

1.3.3.3 Lesen eines Blockes aus dem AG

Mit dieser Funktion kann ein ganzer Block von Daten aus dem AG gelesen werden. Die Funktion startet den Auftrag nur, wartet nicht bis das AG ihn bearbeitet, sondern kehrt sofort wieder zum Aufrufer zurück. Die Daten selbst können daher erst durch einen Aufruf der Funktion "Status-Abfrage" gelesen werden (siehe Kapitel 1.3.4.6).

Register	high	**In**	low	**Out**
AX	$21	typ		status
BX	size	bst		
CX	adr			
DX	len			

Parameter

typ	Elementtyp der Daten bei Einzelelement im AG (DB, MB), siehe Kapitel 1.3.4.10.2.	
size	Elementgröße der Blockdaten	
	(Kennung ob die Einzel-Bytes vertauscht werden)	
	07h	Datenblock von Bytes (kein Vertauschen)
	17h	Datenblock von Worten (Vertauscht High- und Low-Byte)
	27h	Datenblock von Doppelworten (Vertauscht aller 4 Bytes)
bst	Bausteinnummer, Elementtyp DB, DX , FX	
	Bei Elementtyp absolut stehen hier die höchstwertigen Bits von adr.	
adr	Anfangsadresse im Bereich	
len	Anzahl der Daten in Worten, auch wenn size Byte oder Doppelwort ist.	
status	< 0	Fehlernummer, weil Fehler aufgetreten ist
	> 0	Auftragsnummer unter der Status abgefragt werden kann

Hinweis

Für die automatische Anpassung der Daten bei der Übertragung, ist die Art der Daten (Bytes, Worte, Doppelworte) im Block anzugeben. Ein Block kann nur Daten der gleichen Art enthalten. Bei Worten und Doppelworten wird für jedes einzelne Datum die Vertauschung der Bytes entsprechend vorgenommen.

Diese Funktion gibt noch keine Daten zurück! Wurde der Auftrag "Fertig ohne Fehler", so können die Daten durch Aufruf der Funktion Statusabfrage abgeholt werden.

Ein Block-Lese-Auftrag kann nur dann gestartet werden kann, wenn die Kachel 2 leer ist, d.h. es dürfen keine Variablen-Lese-Aufträge oder kein anderer Block-Lese-Auftrag mehr in Bearbeitung sein.

Um sicherzustellen, daß ein fertiger Lese-Auftrag nicht durch einen neuen Auftrag überschrieben wird, bevor die Daten abgeholt werden, wird der Auftrag blockiert. Nach dem Starten eines Auftrages muß dessen Status solange abgefragt werden, bis der Auftrag fertig mit oder ohne Fehler ist. Ist der Auftrags-Status "Fertig ohne Fehler", so werden die Daten an die angegebene Adresse im CP kopiert. Erfolgt keine Statusabfrage, bleibt der Auftrag blockiert und Sie können keine weiteren Lese-Aufträge starten.

1.3.3.4 Schreiben einer Variable in das AG

Mit dieser Funktion kann ein einzelnes Datum (Bit, Byte, Wort ...) in den Speicher des AG geschrieben werden. Bei dem Aufruf ist die Adresse einer zu schreibenden Variable anzugeben. Die Funktion überträgt deren Wert in die Kachel und wartet nicht bis das AG die Daten abholt, sondern kehrt sofort wieder zum Aufrufer zurück.

Register	high	**In**	low	**Out**
AX	$31	typ		status
BX	size	bst		
CX	adr			
DX	-	bit		
SI	offset			
DS	segment			

typ	Elementtyp der Daten bei Einzelelement im AG (DB, MB), siehe Kapitel 1.3.4.10.2.	
size	Kennung der Elementgröße (Bit, Byte, ...), siehe Kapitel 1.3.4.10.1.	
bst	Bausteinnummer, nur bei Elementtyp DB oder DX. Bei Elementtyp absolut stehen hier die höchstwertigen Bits von adr.	
adr	Anfangsadresse im Bereich	
bit	Bit-Nummer wenn die Elementgröße size Bit oder Semaphor ist.	
offset	Offset der Variablen-Adresse (im CP)	
segment	Segment der Variablen-Adresse (im CP)	
status	< 0	Fehlernummer, weil Fehler aufgetreten
	129-255	Auftragsnummer unter der der Status abgefragt werden kann

Hinweis

Im Gegensatz zu den Lese-Aufträgen wird ein Schreibauftrag nicht blockiert. Dennoch sollte ebenfalls der Auftrags-Status solange abgefragt werden, bis der Auftrag mit oder ohne Fehler fertig ist.

Je nach der Elementgröße ist der Zeiger auf die Daten im CP unterschiedlich zu interpretieren:

- Bit oder Semaphore:
  Der Zeiger ist die Adresse eines Bytes, das Bit wird von Bit-Nummer 0 gelesen.
- Byte, linkes Byte, rechtes Byte: Der Zeiger ist die Adresse eines Bytes. Das Byte wird von der Speicherzelle gelesen.
- Wort:
  Der Zeiger ist die Adresse eines Wortes. High- und Low-Byte werden bei der Übertragung vertauscht.
- Doppelwort/Langwort:
  Der Zeiger ist die Adresse eines Langwortes, das Langwort gelesen und die Reihenfolge aller 4 Bytes wird umgedreht.

1.3.3.5 Schreiben eines Blockes in das AG

Mit dieser Funktion kann ein ganzer Datenblock in das AG übertragen werden. Beim Aufruf ist ein Zeiger auf einen Datenblock anzugeben. Die Funktion schreibt die Daten in die Kachel und kehrt sofort wieder zum Aufrufer zurück. Es wird nicht gewartet, bis das AG die Daten übernommen hat. Der Datenblock ist anschließend in der CP wieder frei verfügbar. Er kann auch überschrieben werden.

Register	high	**In**	low	**Out**
AX	$31	typ		status
BX	size	bst		
CX	adr			
DX	len			
SI	offset			
DS	segment			

typ	Elementtyp der Daten bei Blockelement im AG (DB, MB), siehe Kapitel 1.3.4.10.2.	
size	Elementgröße der Blockdaten:	
	07h	Datenblock von Bytes
	17h	Datenblock von Worten
	27h	Datenblock von Doppelworten
bst	Bausteinnummer, bei DB, DX, FB relevant	
	Bei Elementtyp absolut stehen hier die höchstwertigen Bits von adr.	
adr	Anfangsadresse im Bereich	
len	Anzahl der Daten in Worten	
offset	Offset der Block-Adresse (im CP)	
segment	Segment der Block-Adresse (im CP)	
status	< 0	Fehlernummer, weil Fehler aufgetreten
	128	Auftragsnummer unter der der Status abgefragt werden kann

Hinweis

Damit bei der Übertragung die Anpassung der Daten automatisch durchgeführt werden kann, muß die Art der Daten im Block (Bytes, Worte, Doppelworte) angegeben werden. Ein Block kann nur Daten der gleichen Art enthalten. Bei Worten und Doppelworten wird für jedes einzelne Datum die Vertauschung der Bytes entsprechend vorgenommen.

Ein Block-Schreib-Auftrag kann nur dann gestartet werden kann, wenn die Kachel 3 leer ist. D.h. es dürfen keine Variablen-Schreib-Aufträge oder kein anderer Block-Schreib-Auftrag mehr in Bearbeitung sein.

Ein Block-Schreib-Auftrag blockiert die Kachel. Nach dem Starten des Auftrages muß der Status solange abgefragt werden, bis der Auftrag fertig mit oder ohne Fehler ist. Ist der Auftrags-Status "Fertig ohne Fehler" so wurden die Daten an die angegebene Adresse im AG kopiert. Erfolgt keine Statusabfrage bleibt der Auftrag blockiert und keine weiteren Schreib-Aufträge können mehr gestartet werden.

1.3.3.6 Auftrags-Status lesen

Mit dieser Funktion kann der Status eines vorher gestarteten Auftrages abgefragt werden. Bei Lese-Aufträgen - Variablen- und Block-Lese-Aufträge- werden mit dieser Funktion auch die Daten an eine angegebene Adresse in der CP kopiert, wenn der Auftrags-Status "Fertig ohne Fehler" ist.

Register	high	**In**	low	**Out**
AX	$20/$30	a_nr		status
SI	offset			
DS	segment			

fn	Funktionsnummer für Statusabfrage	
	$20	bei Lese-Aufträgen
	$30	bei Schreib-Aufträgen
a_nr	Auftragsnummer	
offset	Offset der Daten-Adresse (im CP)	
	nur bei Lese-Aufträgen (Variablen und Block) anzugeben.	
segment	Segment der Daten-Adresse (im CP)	
	nur bei Lese-Aufträgen (Variablen und Block) anzugeben.	
status	< 0	Auftrag fertig mit Fehler
		Fehlermeldungen des AG werden mit FF00h aufaddiert.
	1	Auftrag noch in Bearbeitung
	2	Auftrags-Status undefiniert
	3	Auftrag fertig ohne Fehler

Vorgehensweise

- Ist der Auftrag noch "in Bearbeitung", so muß die Status-Funktion noch solange aufgerufen werden, bis sich der Status ändert.
- Bei Schreib-Aufträgen (Kachel 3): Ist der Auftrag "Fertig ohne Fehler", so wurden die Daten in das AG geschrieben. Bei Blockelement wurde die Kachel wieder freigegeben.
- Bei Lese-Aufträgen (Kachel 2): Ist der Auftrag "Fertig ohne Fehler" und wurde ein Zeiger für die Daten angegeben, so werden die Daten an die angegebene Adresse im CP kopiert. Dabei wird je nach angegebener Datengröße ggf. eine Vertauschung der Bytes vorgenommen. Der Auftragsblock ist wieder freigegeben, für neue Aufträge. Wird als Zeiger die Adresse NULL (0:0) angegeben, so werden keine Daten kopiert, aber der Auftrag ist trotzdem freigegeben.
- Ist der Auftrags-Status "undefiniert", so wurde der Auftrag bereits früher beendet aber noch nicht durch einen neuen Auftrag überschrieben. Handelte es sich bei dem Auftrag um einen Lese-Auftrag und wurde ein Zeiger ungleich NULL angegeben, so werden die Daten an die angegebene Ziel-Adresse kopiert.
- Ist der Auftrag "Fertig mit Fehler", so wurde der Auftragsblock freigegeben, falls es sich um einen Lese- oder Block-Auftrag handelte.

- Bei einem Lese-Auftrag für einzelne Variablen ist je nach Elementgröße der Zeiger unterschiedlich zu interpretieren:
  - Bit oder Semaphore:

    Der Zeiger ist die Adresse eines Bytes, das Bit wird auf Bit-Nummer 0 geschrieben, das ganze Byte wird hierbei überschrieben.
  - Byte, linkes Byte, rechtes Byte:

    Der Zeiger ist die Adresse eines Bytes. Das Byte wird in die Speicherzelle geschrieben.
  - Wort:
    Der Zeiger ist die Adresse eines Wortes. High- und Low-Byte werden bei der Übertragung vertauscht.
  - Doppelwort/Langwort:
    Der Zeiger ist die Adresse eines Langwortes, die Reihenfolge aller 4 Bytes wird umgedreht und das Langwort wird geschrieben.

- Bei einem Lese-Auftrag für einen Datenblock ist je nach Elementgröße der Zeiger unterschiedlich zu interpretieren:
  - Byte:
    Der Zeiger ist die Adresse eines Blocks von Bytes, die einzelnen Bytes werden unverändert aus dem AG übertragen.
  - Wort:
    Der Zeiger ist die Adresse eines Blocks von Worten, für jedes einzelne Wort werden bei der Übertragung High- und Low-Byte vertauscht.
  - Doppelwort/Langwort:
    Der Zeiger ist die Adresse eines Blocks von Langworten, für jedes einzelne Langwort werden bei der Übertragung alle 4 Bytes in ihrer Reihenfolge umgedreht.

1.3.3.7 Alle Aufträge einer Kachel abbrechen

Register	high	**In** low	**Out**
AX	`fn`		`status`

`fn`	Funktionsnummer für Statusabfrage	
	$28	Abbruch aller Lese-Aufträge
	$38	Abbruch aller Schreib-Aufträgen
`status`	< 0	Fehlernummer, weil Fehler aufgetreten
	0	alle Aufträge wurden abgebrochen.

Mit dieser Funktion können alle noch nicht beendeten Schreib- oder Lese-Aufträge abgebrochen werden. Es muß keine Unterscheidung zwischen Variablen- und Blockaufträgen vorgenommen werden. Auch wenn keine Aufträge in der Kachel mehr aktiv waren, kehrt die Funktion mit dem Returnwert 0 zurück.

1.3.3.8 Prozeßabbild-Status lesen

Register	high	**In** low	**Out**
AX	$70	-	`status`

`status`	0	kein Prozeßabbild verfügbar
	1..255	aktueller Prozeßabbild-Zähler

Mit Hilfe dieser Funktion kann der aktuelle Wert des Prozeßabbild-Zählers (Kachel 7 Adresse 3FEh) gelesen werden. Bei 0 ist kein Prozeßabbild verfügbar.

1.3.3.9 Bereich des Prozeßabbilds lesen

Register	high	In	low	Out
AX	$71	typr		status
BX	adr			
CX	len			
SI	offset			
DS	segment			

typ	Elementtyp der Daten des Prozeßabbildes (EB, MB)	
adr	Anfangsadresse im Bereich	
len	Anzahl der Daten in Bytes oder Worten (je nach TYP)	
offset	Offset der Daten-Adresse (im CP)	
segment	Segment der Daten-Adresse (im CP)	
status	< 0	Fehlernummer, weil Fehler aufgetreten
	= 0	Prozeßabbild nicht verfügbar
	> 0	Zähler für Prozeßabbild (wie bei Statusfunktion)

Mit dieser Funktion kann ein Bereich des aktuellen Prozeßabbilds gelesen werden. Bei dem Zugriff auf die einzelnen Bereiche erfolgt eine Überwachung der Länge des Bereiches, z.B. kann nicht ab EB126 mit der Länge von 4 Bytes gelesen werden, da nur 128 Byte EB vorhanden sind.

Bei Zugriff auf Timer und Zähler erfolgt die Längenangabe in Worten, bei allen anderen Typen in Bytes. Bei Timer und Zähler wird auch beim Transfer das High und Low-Byte jedes Wortes getauscht, so daß die Daten im CP korrekt als Worte verarbeitet werden können.

Durch Angabe des Typs "absolut" kann ein beliebiger Ausschnitt des Prozeßabbildes auch bereichsübergreifend gelesen werden. Die Längenangabe erfolgt in Bytes, auch wenn aus dem Timer oder Zählerbereich gelesen wird. Wird ein Bereich der Timer und Zähler gelesen, so werden High- und Low-Byte getauscht.

Nr.	Elementtyp
06h	Zähler (Länge in Worten anzugeben)
07h	Timer (Länge in Worten anzugeben)
08h	Merker (Länge in Bytes)
09h	EB (Länge in Byte)
0Ah	AB (Länge in Byte)
0Fh	Absoluter Zugriff auf Prozeßabbild (Länge in Byte)

Tab. 1-15: Elementtypen des Prozeßabbilds

1.3.3.10 Fehlermeldungen der CP für Kacheln 2, 3 und 7

hex	dez.	Beschreibung
FFFFh	-1	ungültiger Elementtyp
FFFEh	-2	Längenfehler (z.B. Adresse zu groß, Bit-Nummer zu groß)
FFFDh	-3	ungültige Elementgröße (falscher Wert bei Einzel- oder Blockauftrag)
FFFCh	-4	Elementtyp bei dieser CPU nicht möglich
FFFBh	-5	Kachel voll, 127 Einzel-Aufträge noch in Kachel, oder mind. 1 Einzel-Auftrag und ein Blockauftrag soll gestartet werden.
FFFAh	-6	Zugriff auf Kachel für 10 sec nicht möglich (AG ist vermutlich in Stop)
FFF9h	-7	Auftrag/Kachel ist noch blockiert (Auftrags-Status wurde nicht abgefragt, um Auftrag zu deblockieren).
FFF8h	-8	falsche Auftragsnummer bei Status-Funktion (z.B. Auftragsnr > 255)
FFF7h	-9	fehlerhafter Quelldatenzeiger (Adresse NULL wurde bei Schreib-Auftrag angegeben)
FFF6h	-10	Auftrag nicht in Bearbeitung (unbenutzt!)
FFh	255	Ungültiger Funktionsaufruf
EEh	238	Auftrag bei Initialisierung abgebrochen

Tab. 1-16: Fehlermeldungen der CP für Kacheln 2, 3 und 7

1.3.4 Schnittstelle für Turbo-Pascal (ab Version 4.0)

Für Funktionsaufrufe des COM-Treibers aus Pascal-Programmen heraus, wurde eine Turbo-Pascal-Unit erstellt, die alle Funktionen des Service-Interrupts INT 78 zur Verfügung stellt. Zu jeder Funktion des Treibers ist eine entsprechende Pascal-Prozedur definiert, die die Versorgung der Register, den Aufruf des Interrupts und die Rückgabe der Werte durchführt. So können auch Anwender, die nicht mit systemnaher Programmierung auf CP vertraut sind, alle Möglichkeiten des Treibers nutzen.

Alle erforderlichen Funktionen, Datentypen und Konstanten sind in der Unit CP386LIB enthalten. Bei Verwendung in einem Pascal-Programm muß diese mit "USES CP386LIB" in das Anwenderprogramm eingebunden werden. Die Unit liegt als Quellcode vor. Diese ist vor der Verwendung zu kompilieren. Es muß auch dafür Sorge getragen werden, daß sich die übersetzte Unit CP386LIB.TPU in dem Verzeichnis befindet, in dem Turbo-Pascal nach Units sucht. Die Einstellung erfolgt über die Menü-Punkte "Optionen| Directories| EXE & TPU-Verzeichnis" (vgl. Handbuch oder Hilfe-Funktionen zu Turbo-Pascal).

In den folgenden Abschnitten wird nur ein kurzer Überblick der einzelnen Funktionen gegeben. Eine genaue Beschreibung mit allen wichtigen Informationen ist den Funktionsbeschreibungen der vorhergehenden Abschnitte zu entnehmen.

1.3.4.1 Funktion CP-Status-Abfrage

```
FUNCTION CP_Info (VAR inforec : CP386InfoRec) : INTEGER;
```

Datenstrukturen:

```
TYPE CP386InfoRec =
      RECORD
         CP_id : WORD;              (* Kennung: CP Wert = $C386 *)
         VGA_ver, BIOS_ver : BYTE;  (* Versionsnummern: VGA-BIOS und BIOS *)
         DRV_ver: BYTE;             (* Kennung: Software-Version *)
         CPU_AG : BYTE;             (* Kennung CPU im AG *)
         CP_reg, S5_reg : BYTE;     (* CP- und SPS-Statusregister *)
      END;
```

Datenstruktur für die allgemeine Status-Info Funktion, die Komponenten werden entsprechend der Werte der CP besetzt.

Diese Funktion ruft die Treiber-Funktion "allgemeine Status-Information" auf, vorher wird jedoch festgestellt, ob der Treiber installiert ist. Ist er nicht installiert, gibt die Funktion den Wert -1 zurück. Ist der COM-Treiber installiert, so wird der Wert 0 zurückgegeben, und die Komponenten der Info-Struktur inforec sind entsprechend besetzt. Die Komponente CP_id enthält als Kennung immer den Wert $C386.

1.3.4.2 Lesen eines Einzelelements aus dem AG

```
FUNCTION CP_read_AG (size, typ, bst : BYTE; adr : longint;
                     bit : BYTE) : INTEGER;
```

Parameter

size	Elementgröße von Einzelementen (siehe Kapitel 1.3.4.10.1)
typ	Elementtyp für Einzelelemente (siehe Kapitel 1.3.4.10.2)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
bit	Bit-Nummer

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler

Diese Funktion ruft die Treiber-Funktion "Lesen eines Einzelelementes aus dem AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Empfohlene Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden. Andernfalls kann z.B. die Kachel blockiert werden.

```
VAR   a_nr,                                (* Auftragsnummer für Lese-Auftrag *)
      stat : INTEGER;                      (* momentaner Auftrags-Status      *)
      wert : BYTE;                         (* aus dem AG gelesener Wert       *)
BEGIN

(* Auftrag starten *)
   a_nr := CP_read_AG(LBYTE_ELM, DB_SNG, 10, 1, 0);

   IF a_nr < 0                             (* Fehler ist aufgetreten          *)
   THEN WriteLn('Auftrag wurde beendet mit Fehler: ', a_nr)

   ELSE  BEGIN                             (* a_nr enthält die Auftragsnummer *)
      REPEAT
            stat:= CP_stat_AG(a_nr,Addr(wert));(*Auftrags-Status/Daten abholen*)
      UNTIL stat <> REQ_WRKN;                  (*solange, bis Auftrag fertig *)
      CASE stat  OF
        REQ_NO_ERR:     WriteLn('Datum: ', wert, ' wurde gelesen');
        REQ_UNDEF:      WriteLn('Auftragstatus ist undefiniert,Datum: ',wert, '
                                gelesen');
        ELSE            WriteLn('Auftrag ist fertig mit Fehler: ', stat)
      END;
   END;
END.
```

1.3.4.3 Lesen eines Blockes aus dem AG

```
FUNCTION CP_readn_AG (size, typ, bst:BYTE; adr:longint;
                      len:WORD):integer;
```

Parameter

size	Datentyp der Blockelemente (siehe Kapitel 1.3.4.10.4)
typ	Elementtyp der Blockelemente (siehe Kapitel 1.3.4.10.3)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Datenanzahl in Worten

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Lesen eines Blockes aus dem AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer 0 als Funktionswert zurückgegeben.

Empfohlene Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden. Andernfalls kann z.B. die Kachel blockiert werden.

```
VAR   a_nr,                                (* Auftragsnummer für Lese-Auftrag *)
      stat : INTEGER;                      (* momentaner Auftrags-Status      *)
      buff : ARRAY[1..100 ] OF INTEGER     (* aus dem AG gelesene Daten       *)
BEGIN

(* Auftrag starten *)
   a_nr := CP_readn_AG(W_BLOCK, DB_BLK, 5, 10, 100);

   IF a_nr < 0                             (* Fehler ist aufgetreten          *)
   THEN WriteLn('Auftrag wurde beendet mit Fehler: ', a_nr)

   ELSE  BEGIN                             (* a_nr enthält die Auftragsnummer *)
      REPEAT
            stat:= CP_stat_AG(a_nr, Addr(buf));(*Auftrags-Status/Daten abholen*)
      UNTIL stat <> REQ_WRKN;                  (*solange, bis Auftrag fertig *)

      CASE stat  OF
          REQ_NO_ERR:   WriteLn('Puffer wurde gelesen');
          REQ_UNDEF:    WriteLn('Auftragstatus ist undefiniert, Puffer wurde
                                 gelesen');
          ELSE          WriteLn('Auftrag ist fertig mit Fehler: ', stat)
      END;

   END;
END.
```

1.3.4.4 Schreiben eines Einzelelementes in das AG

```
FUNCTION CP_write_AG (size, typ, bst : BYTE; adr: longint;
                      bit: BYTE; p: POINTER): integer;
```

Parameter

size	Elementgröße (siehe Kapitel 1.3.4.10.1)	
typ	Elementtyp Einzelelemente (siehe Kapitel 1.3.4.10.2)	
bst	Bausteinnummer	
adr	Adresse im Baustein oder absolute Adresse	
bit	Bit-Nummer	
p	Zeiger auf zu schreibendes Datum	
	bei Elementgröße Bit, Semaphore oder Byte:	Zeiger auf ein Byte
	bei Elementgröße Wort:	Zeiger auf ein Wort
	bei Elementgröße Doppelwort:	Zeiger auf ein Doppelwort

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler

Diese Funktion ruft die Treiber-Funktion "Schreiben eines Einzelelementes in das AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Empfohlene Vorgehensweise beim Aufruf

```
VAR   a_nr,                                  (* Auftragsnummer für Lese-Auftrag *)
      stat : INTEGER;                        (* momentaner Auftrags-Status      *)
      wert : BYTE;                           (* zu schreibender Wert            *)
BEGIN
 ...
(* Auftrag starten *)
   wert := $7E;
   a_nr := CP_write_AG(BYTE_ELM, DB_SNG, 10, 1, 0, Addr(wert));

   IF a_nr < 0             (* Fehler ist aufgetreten *)
   THEN WriteLn('Auftrag wurde beendet mit Fehler: ', a_nr)

   ELSE  BEGIN                               (* a_nr enthält die Auftragsnummer *)
      REPEAT
            stat := CP_stat_AG(a_nr, NIL);          (* Auftrags-Status lesen *)
      UNTIL stat <> REQ_WRKN;                 (* solange bis, Auftrag fertig *)
      CASE stat OF
          REQ_NO_ERR:   WriteLn('Datum: ', wert, ' wurde geschrieben');
          REQ_UNDEF:    WriteLn('Auftragstatus ist undefiniert.');
          ELSE          WriteLn('Auftrag ist fertig mit Fehler: ', stat)
      END;
   END;
END.
```

1.3.4.5 Schreiben eines Blockes in das AG

```
FUNCTION CP_writen_AG (size, typ, bst : BYTE; adr : longint;
                       len : word; p : POINTER) : integer;
```

Parameter

size	Datentyp der Blockelemente (siehe Kapitel 1.3.4.10.4)
typ	Elementtyp Blockelemente (siehe Kapitel 1.3.4.10.3)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Datenanzahl in Worten
p	Zeiger auf zu schreibenden Datenblock

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler

Diese Funktion ruft die Treiber-Funktion "Schreiben eines Blockes in das AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, so wird die Auftragsnummer $80h als Funktionswert zurückgegeben.

Empfohlene Vorgehensweise beim Aufruf

```
VAR   a_nr,                                (* Auftragsnummer für Lese-Auftrag *)
      stat : INTEGER;                      (* momentaner Auftrags-Status      *)
      i : INTEGER;
      buff : ARRAY[1..100 ] OF INTEGER;    (* zu schreibende Daten            *)
BEGIN
 ...
   FOR i := 1 TO 100 DO
            buff[i] := i;                  (* Datenpuffer vorbesetzen         *)
(* Auftrag starten *)
   a_nr := CP_writen_AG(B_BLOCK, DB_BLK, 5, 10, 100, Addr(buff));

   IF a_nr < 0                             (* Fehler ist aufgetreten          *)
   THEN WriteLn('Auftrag wurde beendet mit Fehler: ', a_nr)

   ELSE  BEGIN                             (* a_nr enthält die Auftragsnummer *)
      REPEAT
            stat := CP_stat_AG(a_nr, NIL);          (* Auftrags-Status lesen *)
      UNTIL stat <> REQ_WRKN;                        (* solange bis, Auftrag *)

      CASE stat OF
          REQ_NO_ERR:   WriteLn('Puffer wurde geschrieben');
          REQ_UNDEF:    WriteLn('Auftragstatus ist undefiniert.');
          ELSE          WriteLn('Auftrag ist fertig mit Fehler: ', stat)
      END;
   END;
END.
```

1.3.4.6 Status eines Auftrags lesen

```
FUNCTION CP_stat_AG (a_nr : INTEGER; p: POINTER): INTEGER;
```

Parameter

a_nr	Auftragsnummer des zu testenden Auftrags	
p	Zeiger auf Datum oder Datenblock im CP-Speicher	
	(nur bei Lese-Aufträgen mit Einzel- und Blockelement zu besetzen)	
	bei Elementgröße Bit, Semaphore oder Byte	Zeiger auf ein Byte
	bei Elementgröße Wort	Zeiger auf ein Wort
	bei Elementgröße Doppelwort	Zeiger auf ein Doppelwort
	bei Elementgröße Block	Zeiger auf Puffer für Datenblock

Rückgabe

Auftrags-Status oder negative Zahl bei Fehler

Diese Funktion ruft die Treiber-Funktion "Statusabfrage zu Auftrag" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Trat bei der Ausführung des Auftrages ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, so wird der Auftrags-Status zurückgegeben (siehe Kapitel 1.3.4.10.5).

1.3.4.7 Alle Aufträge einer Kachel abbrechen

```
FUNCTION CP_cncl_AG (a_nr : BYTE): INTEGER;
```

Parameter

a_nr	Kennung für Kachel 2 oder 3
	$00 alle noch aktiven Aufträge der Kachel 2 abbrechen
	$80 alle noch aktiven Aufträge der Kachel 3 abbrechen

Rückgabe

0 oder negative Zahl bei Fehler

Diese Funktion ruft die Treiber-Funktion "Abbruch aller Aufträge einer Kachel" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Trat bei der Ausführung des Auftrags ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, d.h. alle Aufträge wurden abgebrochen, so wird 0 zurückgegeben.

1.3.4.8 Prozeßabbild-Status lesen

```
FUNCTION CP_stat_PA : BYTE;
```

Rückgabe

Prozeßabbild-Zähler

Diese Funktion ruft die Treiber-Funktion "Statusabfrage Prozeßabbild" auf. Die Funktion gibt den Prozeßabbild-Zähler zurück.

1.3.4.9 Bereich des Prozeßabbilds lesen

```
FUNCTION CP_read_PA (typ : BYTE; adr, len : WORD;
                     p : POINTER) : INTEGER;
```

Parameter

typ	Elementtyp Prozeßabbild (siehe Kapitel 1.3.4.10.6)
adr	Adresse im Bereich oder absolute Adresse
len	Anzahl der Daten (Bytes oder Worte) je nach Typ
p	Zeiger auf Datenpuffer im Speicher

Rückgabe

Prozeßabbild-Zähler

Diese Funktion ruft die Treiber-Funktion "Lesen eines Bereichs des Prozeßabbilds" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt der Treiber-Funktion beschrieben. Trat bei der Ausführung des Auftrages ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird der aktuelle Wert des Prozeßabbild-Zählers zurückgegeben.

1.3.4.10 Konstanten von Turbo Pascal

Folgende Konstanten sind bereits vordefiniert. Aufgrund der besseren Lesbarkeit und Übersichtlichkeit wird empfohlen, diese Konstanten auch im Programmtext zu verwenden. Eine Anpassung an spätere Änderungen des COM-Treibers kann so leichter vorgenommen werden..

1.3.4.10.1 Konstanten für Elementgrößen

```
CONST
     BIT_ELM        = $00;    (* Bit *)
     SEMA_ELM       = $01;    (* Bit als Semaphor *)
     BYTE_ELM       = $02;    (* Byte *)
     LBYTE_ELM      = $02;    (* linkes Byte eines Wortes *)
     RBYTE_ELM      = $03;    (* rechtes Byte eines Wortes *)
     WORD_ELM       = $04;    (* Wort *)
     DWORD_ELM      = $05;    (* Doppelwort *)
     BLOCK_ELM      = $07;    (* Block *)
```

1.3.4.10.2 Konstanten für Elementtypen bei Einzelelementen

```
CONST
     DB_SNG    = $00;    (* DB *)
     DX_SNG    = $01;    (* DB im Externspeicher *)
     BA_SNG    = $02;    (* BA *)
     BB_SNG    = $03;    (* BB *)
     BS_SNG    = $04;    (* BS *)
     BT_SNG    = $05;    (* BT *)
     Z_SNG     = $06;    (* Zähler *)
     T_SNG     = $07;    (* Timer *)
     MB_SNG    = $08;    (* Merker *)
     EB_SNG    = $09;    (* Eingangsbereich *)
     AB_SNG    = $0A;    (* Ausgangsbereich *)
     PB_SNG    = $0B;    (* P-Peripherie *)
     QB_SNG    = $0C;    (* Q-Peripherie *)
     ABS_SNG   = $0F;    (* Absoluter Speicher *)
```

1.3.4.10.3 Konstanten für Elementtypen bei Blockelementen

```
CONST
     DB_BLK    = $00;    (* Datenbaustein *)
     DX_BLK    = $01;    (* DB im Externspeicher *)
     BA_BLK    = $02;    (* BA *)
     BB_BLK    = $03;    (* BB *)
     BS_BLK    = $04;    (* BS *)
     BT_BLK    = $05;    (* BT *)
     FB_BLK    = $06;    (* FB *)
     FX_BLK    = $07;    (* FB im Externspeicher *)
     OB_BLK    = $08;    (* OB *)
     PB_BLK    = $09;    (* PB *)
     SB_BLK    = $0A;    (* SB *)
     ABS_BLK   = $0F;    (* Absoluter Speicher *)
```

1.3.4.10.4 Konstanten für den Datentyp bei Blockelementen

```
CONST
     B_BLOCK   = $07;    (* Typ: Block von Bytes  *)
     W_BLOCK   = $17;    (* Typ: Block von Worten  *)
     D_BLOCK   = $27;    (* Typ: Block von Langworten *)
```

1.3.4.10.5 Kennungen für Auftrags-Status

```
CONST
     REQ_WRKN       = $01;    (* Auftrag in Bearbeitung *)
     REQ_UNDEF      = $02;    (* Auftrags-Status undefiniert *)
     REQ_NO_ERR     = $03;    (* Auftrag fertig ohne Fehler *)
```

1.3.4.10.6 Konstanten für Elementtypen bei Prozeßabbild

```
CONST
     Z_PA      = $06;    (* Zähler  *)
     T_PA      = $07;    (* Timer  *)
     MB_PA     = $08;    (* Merker  *)
     EB_PA     = $09;    (* Eingangsbereich  *)
     AB_PA     = $0A;    (* Ausgangsbereich  *)
     ABS_PA    = $0F;    (* absoluter Block im PA  *)
```

1.3.4.10.7 Konstanten für Fehlermeldungen: Kachel 2, 3 und 7

```
CONST
```

`ERR_S5_TYP = $01;` (* ungültiger Elementtyp *)

Bei Einzelelementzugriff mit Elementtyp DX_SNG, BA_SNG, BB_SNG, BT_SNG oder QB_SNG bzw. bei Blockelementzugriff mit Elementtyp DX_BLK, BA_BLK, BB_BLK, BT_BLK oder FX_BLK wurde versucht auf Daten in einem AG der 115U-Reihe zuzugreifen. Diese Elementtypen existieren in diesem AG-Typ jedoch nicht.

Abhilfe: Parameter "typ" im Funktionsaufruf der CP-Anwender-Software korrigieren.

`ERR_S5_BST = $02;` (* Baustein nicht vorhanden *)

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementtyp DB_BLK wurde versucht auf einen nicht existierenden Baustein zuzugreifen.

Abhilfe: Datenbaustein im AG anlegen oder Parameter "bst" im Funktionsaufruf der CP-Anwendersoftware korrigieren.

```
ERR_S5_ELM      = $03;
```
(* Element nicht vorhanden *)

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementzugriff mit Elementtyp DB_BLK wurde versucht auf Daten in einem DB zuzugreifen, die nicht vorhanden sind.

Abhilfe: Datenbaustein im AG entsprechend verlängern oder Parameter "adr" bzw. "len" im Funktionsaufruf der CP-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten bzw. Zählerstände mit einer Nummer > 127 zuzugreifen.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG wurde versucht auf Merker mit Nummer > 199 bei Elementgröße Byte, mit Nummer > 198 bei Elementgröße Wort oder mit Nummer > 196 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software auf Wertigkeit überprüfen.

Bei Einzelelementzugriff mit Elementtyp EB_ - oder AB_SNG wurde versucht auf das Prozeßabbild des E/A-Bereichs mit Nummer > 127 bei Elementgröße Byte, mit Nummer > 126 bei Elementgröße Wort oder mit Nummer > 124 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software auf Wertigkeit überprüfen.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht auf Elemente der P-Peripherie mit Nummer > 255 bei Elementgröße Byte, mit Nummer > 254 bei Elementgröße Wort oder mit Nummer > 252 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software auf Wertigkeit überprüfen.

```
ERR_S5_SIZE     = $04;
```
(* ungültige Elementgröße *)

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten oder Zählerstände zuzugreifen, wobei der Parameter Elementgröße nicht auf Wortzugriff (WORD_ELM) gesetzt war.

Abhilfe: Parameter "size" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG wurde versucht, mit dem Parameter Elementgröße RBYTE_ELM auf Merker oder absolute Adressen zuzugreifen.

Abhilfe: Parameter "size" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht, mit dem Parameter Elementgröße SEMA_ELM oder RBYTE_ELM auf den Ein- bzw. Ausgängen im Prozeßabbild zuzugreifen.

Abhilfe: Parameter "typ" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht mit dem Parameter Elementgröße BIT_ELM, SEMA_ELM oder RBYTE_ELM auf die P-Peripherie zuzugreifen.

Abhilfe: Parameter "typ" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei lesenden Einzelelementzugriff mit Elementtyp ABS_SNG wurde versucht von absoluten Adressen mit Elementgröße SEMA_ELM zu lesen. Diese Zugriffsart ist unter absoluter Adressierung nur schreibend möglich! Wenn einzelne Bits gelesen werden sollen, ist die Elementgröße BIT_ELM zu verwenden.

Abhilfe: Parameter "typ" im Funktionsaufruf der CP-Anwender-Software korrigieren.

```
ERR_S5_BIT      = $05;  (* Bit-Nummer zu groß  *)
```

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG und der Elementgröße BIT_ELM oder SEMA_ELM wurde versucht, auf ein Merker- oder Absolutadress-Bit mit einer Bitnummer > 7 (15) zuzugreifen.

Abhilfe: Parameter "bit" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht auf ein E/A-Bit mit einer Bitnummer > 7 zuzugreifen.

Abhilfe: Parameter "bit" im Funktionsaufruf der CP-Anwender-Software korrigieren.

```
ERR_S5_STRT     = $06;  (* ungültige Anfangsadresse  *)
```

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde ein Blocktransfer über Bausteine versucht, wobei die relative Anfangsadresse im Block > 32767 ist.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software korrigieren.

`ERR_S5_LEN = $07;` (* ungültige Blocklänge *)

Bei Blockelementzugriff unter allen Elementtypen wurde ein Blocktransfer mit einer Länge > 504 Worten versucht.

Abhilfe: Parameter "len" im Funktionsaufruf der CP-Anwender-Software korrigieren.

`ERR_S5_ADR = $08;` (* Adresse zu groß *)

Bei Einzel- oder Blockelementzugriff mit Elementtyp ABS_SNG wurde versucht auf eine Adresse > FFFFh in einem AG der 115U-Reihe zuzugreifen. Die CPUs (bis CPU 944) haben jedoch nur 64 KByte Adressraum.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software korrigieren.

`ERR_S5_QVZ = $09;` (* QVZ/ADF im AG bei lesen/schreiben *)

Es wurde versucht auf einen Adressbereich zuzugreifen, der physikalisch nicht vorhanden ist.

Die AGs der 135- und 155-Reihe stellen diese Fehlermeldungen zu Verfügung. Ein AG der 115U-Reihe würde hierbei in den Stop-Zustand versetzt werden.

Abhilfe: Parameter "typ" bzw. "adr" im Funktionsaufruf der CP-Anwendersoftware korrigieren.

`ERR_S5_944 = $0A;` (* CPU 944: Baustein im Prog.Bank *)

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde versucht auf einen Baustein zuzugreifen, der nicht in der DB-Bank liegt. (Betrifft nur CPU 944 vom AG-Typ 115U)

Abhilfe: Baustein im AG in DB-Bank anlegen (über BIB-Nr. 19285) oder Funktionsaufruf in der CP-Anwender-Software korrigieren.

1.3.5 Schnittstelle zu Turbo-C (2.0 und C++ ab 1.0), Microsoft-C 6.0

Für Funktionsaufrufe des COM-Treibers aus C-Programmen heraus wurde ein Bibliotheks-File erstellt, das alle Funktionen des Service-Interrupts INT 78 zur Verfügung stellt. Zu jeder Funktion des Treibers ist eine entsprechende C-Funktion definiert, die die Versorgung der Register, den Aufruf des Interrupts und die Rückgabe der Werte durchführt. So können auch Anwender, die nicht mit systemnaher Programmierung auf CP vertraut sind, alle Möglichkeiten des Treibers nutzen.

In dem Include-File "CP386DEF.H" sind Datentypen und Konstante für Elementgrößen, Elementtypen und Fehlernummern definiert, ebenso sind dort auch die Funktionsprototypen der nachfolgend beschriebenen Funktionen im ANSI-C Stil definiert. Das Include-File muß im Anwenderprogramm genannt werden.

Alle erforderlichen Funktionen sind in dem File CP386LIB.C implementiert. Bei Verwendung in einem Programm muß die Datei CP386LIB.OBJ eingebunden werden. Je nach Programmier-Umgebung und Version ist die Datei in die Project-Datei (Turbo-C) oder Dependencie List (Microsoft-C) aufzunehmen oder im Make-File aufzunehmen. Details hierzu sind den jeweiligen Handbüchern zu entnehmen.

Hinweis

In "CP386LIB.H" ist "byte" als unsigned char und "word" als unsigned short definiert.

1.3.5.1 Funktion CP-Status-Abfrage

Datenstrukturen

```
int CP_info (p)
```

```
typedef struct
{
  word CP_id;                /* Kennung: CP Wert = $C386 */
  byte VGA_ver, BIOS_ver;    /* Versionsnummern: VGA-BIOS und BIOS */
  byte DRV_ver;              /* Kennung: Software-Version */
  byte CPU_AG;               /* Kennung CPU im AG */
  byte CP_reg, S5_reg;       /* CP- und SPS-Statusregister */
} CP_InfoBlk;                /* Info-Block */
```

Datenstruktur für die allgemeine Status-Info Funktion, die Komponenten werden entsprechend der Werte der CP besetzt.

1.3.5.2 Lesen eines Einzelelements aus dem AG

```
int CP_read_AG (byte size, byte typ, byte bst, unsigned long
                adr, byte bit);
```

size	Elementgröße (siehe Kapitel 1.3.5.10.1)
typ	Elementtyp Einzelelemente (siehe Kapitel 1.3.5.10.2)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
bit	Bit-Nummer

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Lesen eines Einzelelementes aus dem AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Empfohlene Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden. Andernfalls kann z.B. die Kachel blockiert werden.

```
int a_nr;               /* Auftragsnummer für Lese-Auftrag */
int stat;               /* momentaner Auftrags-Status */
int time_count=4000;    /* Zeitzähler für Timeout (= 4 Sekunden)*/
byte wert;              /* aus dem AG gelesener Wert */

/* Auftrag starten */
a_nr = CP_read_AG(LBYTE_ELM, DB_SNG, 10, 1, 0);

if(a_nr < 0)       /* Fehler ist aufgetreten */
  printf("Auftrag wurde beendet mit Fehler: %d\n", a_nr);

else
{                  /* a_nr enthält die Auftragsnummer */
  do
  {
    delay(1);                          /* 1 ms warten */
    stat = CP_stat_AG(a_nr, &wert);   /* Auftrags-Status/Daten abholen*/
    /* solange bis Zeit abgelaufen bzw. Auftrag fertig mit oder ohne Fehler */
  } while((time_count-- > 0) && (stat == REQ_WRKN));

  switch(stat)
  {
    case REQ_NO_ERR:
      printf("Datum: %d wurde gelesen\n", wert);
    break;
    case REQ_UNDEF:
      printf("Auftragstatus undefiniert, Datum: %d gelesen\n",wert);
    break;
    default:
      printf("Auftrag ist fertig mit Fehler: %d\n", stat);
  }
}
```

1.3.5.3 Lesen eines Blocks aus dem AG

```
int CP_readn_AG (byte size, byte typ, byte bst, unsigned long adr,
                 word len);
```

size	Datentyp der Blockelemente (siehe Kapitel 1.3.4.10.4)
typ	Elementtyp Blockelemente (siehe Kapitel 1.3.5.10.3)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Datenanzahl in Worten

Rückgabe

Auftragsnummer 0 oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Lesen eines Blockes aus dem AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer 0 als Funktionswert zurückgegeben.

Empfohlene Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden. Andernfalls kann z.B. die Kachel blockiert werden.

```
int a_nr;               /* Auftragsnummer für Lese-Auftrag */
int stat;               /* momentaner Auftrags-Status */
int time_count=4000;    /* Zeitzähler für Timeout (= 4 Sekunden) */
int buff[100];          /* aus dem AG gelesener Wert */

/* Auftrag starten */
a_nr = CP_readn_AG(W_BLOCK, DB_BLK, 5, 10, 100);

if(a_nr < 0)       /* Fehler ist aufgetreten */
  printf("Auftrag wurde beendet mit Fehler: %d\n", a_nr);

else
{                  /* a_nr enthält die Auftragsnummer */
  do
  {
    delay(1);                          /* 1 ms warten */
    stat = CP_stat_AG(a_nr, &buff);    /* Auftrags-Status/Daten abholen */
    /* solange bis Zeit abgelaufen bzw. Auftrag fertig mit oder ohne Fehler */
  } while((time_count-- > 0) && (stat == REQ_WRKN));

  switch(stat)
  {
    case REQ_NO_ERR:
      printf("Daten wurden gelesen\n");
    break;
    case REQ_UNDEF:
      printf("Auftragstatus undefiniert\n");
    break;
    default:  printf("Auftrag ist fertig mit Fehler: %d\n", stat);
  }
}
```

1.3.5.4 Schreiben eines Einzelelements in das AG

```
int CP_write_AG (byte size, byte type, byte bst, unsigned long
                 adr, byte bit, void far *p);
```

size	Elementgröße (siehe Kapitel 1.3.5.10.1)	
typ	Elementtyp Einzelelemente (siehe Kapitel 1.3.5.10.2)	
bst	Bausteinnummer	
adr	Adresse im Baustein oder absolute Adresse	
bit	Bit-Nummer	
p	Zeiger auf zu schreibendes Datum im CP-Speicher	
	bei Elementgröße Bit, Semaphore oder Byte	Zeiger auf ein Byte
	bei Elementgröße Wort	Zeiger auf ein Wort
	bei Elementgröße Dopppelwort	Zeiger auf ein Doppelwort

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Schreiben eines Einzelelementes in das AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Empfohlene Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden. Andernfalls kann z.B. die Kachel blockiert werden.

```
int a_nr;              /* Auftragsnummer für Lese-Auftrag */
int stat;              /* momentaner Auftrags-Status */
int time_count=4000;   /* Zeitzähler für Timeout (= 4 Sekunden) */
byte wert = 0x5A;      /* aus dem AG gelesener Wert */
/* Auftrag starten */
a_nr = CP_write_AG(LBYTE_ELM, DB_SNG, 10, 1, 0, &wert);

if(a_nr < 0)               /* Fehler ist aufgetreten */
  printf("Auftrag wurde beendet mit Fehler: %d\n", a_nr);
else
{                          /* a_nr enthält die Auftragsnummer */
  do
  {
    delay(1);                          /* 1 ms warten *
    stat = CP_stat_AG(a_nr, NULL);     /* Auftrags-Status lesen */
    /* solange bis Zeit abgelaufen bzw. Auftrag fertig mit oder ohne Fehler */
  } while((time_count-- > 0) && (stat == REQ_WRKN));
  switch(stat)
  {
    case REQ_NO_ERR:
      printf("Datum: %x wurde geschrieben\n", wert);
    break;
    case REQ_UNDEF:    printf("Auftragstatus undefiniert\n");
      break;
    default:
      printf("Auftrag ist fertig mit Fehler: %d\n", stat);
  }
}
```

1.3.5.5 Schreiben eines Blocks in das AG

```
int CP_writen_AG (byte size, byte typ, byte bst, unsigned long
                  adr, word len, void far *p);
```

size	Datentyp der Blockelemente (siehe Kapitel 1.3.4.10.4)
typ	Elementtyp Blockelemente (siehe Kapitel 1.3.5.10.3)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Datenanzahl in Worten
p	Zeiger auf zu schreibenden Datenblock im CP-Speicher

Rückgabe

Auftragsnummer $80 oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Schreiben eines Blockes in das AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer 80h als Funktionswert zurückgegeben.

Empfohlene Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden. Andernfalls kann z.B. die Kachel blockiert werden.

```
int a_nr;                   /* Auftragsnummer für Lese-Auftrag */
int stat;                   /* momentaner Auftrags-Status */
int i;
int time_count=4000;        /* Zeitzähler für Timeout (= 4 Sekunden) */
int buff[100];              /* zu schreibende Daten */
for(i = 0; i< 100; i++)
buff[i] = (byte)i;          /* Datenpuffer vorbesetzen */
/* Auftrag starten */
a_nr = CP_writen_AG(B_BLOCK, DB_BLK, 5, 10, 100, &buff);

if(a_nr < 0)                /* Fehler ist aufgetreten */
  printf("Auftrag wurde beendet mit Fehler: %d\n", a_nr);
else
{                           /* a_nr enthält die Auftragsnummer */
  do
  {
    delay(1);                       /* 1 ms warten */
    stat = CP_stat_AG(a_nr, NULL);  /* Auftrags-Status lesen */
    /* solange,bis Auftrag fertig mit oder ohne Fehler */
  } while((time_count-- > 0) && (stat == REQ_WRKN));

  switch(stat)
  {
    case REQ_NO_ERR:
     printf("Daten wurden geschrieben\n");
    break;
    case REQ_UNDEF:
      printf("Auftragstatus undefiniert\n");
    break;
    default:
      printf("Auftrag ist fertig mit Fehler: %d\n", stat);
  }
}
```

1.3.5.6 Status eines Auftrags lesen

```
int CP_stat_AG (int r, void far *p);
```

a_nr	Auftragsnummer des zu testenden Auftrags	
p	Zeiger auf Datum oder Datenblock im CP-Speicher	
	(nur bei Lese-Aufträgen mit Einzel- und Blockelement)	
	bei Elementgröße Bit, Semaphore oder Byte	Zeiger auf ein Byte
	bei Elementgröße Wort	Zeiger auf ein Wort
	bei Elementgröße Dopppelwort	Zeiger auf ein Doppelwort
	bei Elementgröße Block	Zeiger auf Puffer für Datenblock

Rückgabe

Auftrags-Status oder negative Zahl bei Fehler

Diese Funktion ruft die Treiber-Funktion "Statusabfrage zu Auftrag" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Trat bei der Ausführung des Auftrages ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, so wird der Auftrags-Status zurückgegeben (siehe Kapitel 1.3.4.10.5).

1.3.5.7 Alle Aufträge einer Kachel abbrechen

```
int CP_cncl_AG (int a_nr);
```

a_nr	Kennung für Kachel 2 oder 3
	$00 alle noch aktiven Aufträge der Kachel 2 abbrechen
	$80 alle noch aktiven Aufträge der Kachel 3 abbrechen

Diese Funktion ruft die Treiber-Funktion "Abbruch aller Aufträge einer Kachel" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt über die Treiber-Funktion beschrieben. Trat bei der Ausführung des Auftrages ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, d.h. alle Aufträge wurden abgebrochen, so wird 0 zurückgegeben.

1.3.5.8 Prozeßabbild-Status lesen

```
byte CP_stat_PA();
```

Rückgabe

Prozeßabbild-Zähler

Diese Funktion ruft die Treiber-Funktion "Statusabfrage Prozeßabbild" auf. Die Funktion gibt den Prozeßabbild-Zähler zurück.

1.3.5.9 Bereich des Prozeßabbilds lesen

```
int CP_read_PA (byte typ, word adr, word len, void far *p);
```

typ	Elementtyp Einzelelemente (siehe Kapitel 1.3.5.10.2)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Anzahl der Daten (Bytes oder Worte) je nach Typ
p	Zeiger auf Datenpuffer im CP-Speicher

Rückgabe

Prozeßabbild-Zähler

Diese Funktion ruft die Treiber-Funktion "Lesen eines Bereiches des Prozeßabbildes" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist im Abschnitt der Treiber-Funktion beschrieben. Trat bei der Ausführung des Auftrages ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird der aktuelle Wert des Prozeßabbild-Zählers zurückgegeben.

1.3.5.10 Konstanten

Folgende Konstanten sind bereits vordefiniert. Es wird empfohlen, diese Konstanten auch im Programmtext zu verwenden, aus Gründen der Übersichtlichkeit und besseren Lesbarkeit. Außerdem können leichter Anpassungen an mögliche spätere Änderungen des COM-Treibers vorgenommen werden.

1.3.5.10.1 Konstanten für Elementgrößen

```
#define BIT_ELM         0x00  /* Bit */
#define SEMA_ELM        0x01  /* Bit als Semaphor */
#define BYTE_ELM        0x02  /* Byte */
#define LBYTE_ELM       0x02  /* linkes Byte eines Wortes */
#define RBYTE_ELM       0x03  /* rechtes Byte eines Wortes */
#define WORD_ELM        0x04  /* Wort */
#define DWORD_ELM       0x05  /* Doppelwort */
#define BLOCK_ELM       0x07  /* Block */
```

1.3.5.10.2 Konstanten für Elementtypen bei Einzelelementen

```
#define DB_SNG    0x00  /* DB */
#define DX_SNG    0x01  /* DB im Externspeicher */
#define BA_SNG    0x02  /* BA */
#define BB_SNG    0x03  /* BB */
#define BS_SNG    0x04  /* BS */
#define BT_SNG    0x05  /* BT */
#define Z_SNG     0x06  /* Zähler */
#define T_SNG     0x07  /* Timer */
#define MB_SNG    0x08  /* Merker */
#define EB_SNG    0x09  /* Eingangsbereich */
#define AB_SNG    0x0A  /* Ausgangsbereich */
#define PB_SNG    0x0B  /* P-Peripherie */
#define QB_SNG    0x0C  /* Q-Peripherie */
#define ABS_SNG   0x0F  /* Absoluter Speicher */
```

1.3.5.10.3 Konstanten für Elementtypen bei Blockelementen

```
#define DB_BLK    0x00  /* Datenbaustein */
#define DX_BLK    0x01  /* DB im Externspeicher */
#define BA_BLK    0x02  /* BA */
#define BB_BLK    0x03  /* BB */
#define BS_BLK    0x04  /* BS */
#define BT_BLK    0x05  /* BT */
#define FB_BLK    0x06  /* FB */
#define FX_BLK    0x07  /* FB im Externspeicher */
#define OB_BLK    0x08  /* OB */
#define PB_BLK    0x09  /* PB */
#define SB_BLK    0x0A  /* SB */
#define MB_BLK    0x0B  /* MB */
#define ABS_BLK   0x0F  /* Absoluter Speicher */
```

1.3.5.10.4 Konstanten für den Datentyp bei Blockelementen

```
#define B_BLOCK   0x07  /* Typ: Block von Bytes  */
#define W_BLOCK   0x17  /* Typ: Block von Worten  */
#define D_BLOCK   0x27  /* Typ: Block von Langworten */
```

1.3.5.10.5 Kennungen für Auftrags-Status

```
#define REQ_WRKN  0x01  /* Auftrag in Bearbeitung */
#define REQ_UNDEF       0x02  /* Auftrags-Status undefiniert */
#define REQ_NO_ERR      0x03  /* Auftrag fertig ohne Fehler */
```

1.3.5.10.6 Konstanten für Elementtypen bei Prozeßabbild

```
#define Z_PA      0x06  /* Zähler */
#define T_PA      0x07  /* Timer */
#define MB_PA     0x08  /* Merker */
#define EB_PA     0x09  /* Eingangsbereich */
#define AB_PA     0x0A  /* Ausgangsbereich */
#define ABS_PA    0x0F  /* absoluter Block im PA */
```

1.3.5.10.7 Konstanten für Fehlermeldungen: Kachel 2, 3 und 7

`ERR_S5_TYP = 0x01;` /* ungültiger Elementtyp */

Bei Einzelelementzugriff mit Elementtyp DX_SNG, BA_SNG, BB_SNG, BT_SNG oder QB_SNG bzw. bei Blockelementzugriff mit Elementtyp DX_BLK, BA_BLK, BB_BLK, BT_BLK oder FX_BLK wurde versucht auf Daten in einem AG der 115U-Reihe zuzugreifen. Diese Elementtypen existieren in diesem AG-Typ jedoch nicht.

Abhilfe: Parameter "typ" im Funktionsaufruf der CP-Anwender-Software korrigieren.

`ERR_S5_BST = 0x02;` /* Baustein nicht vorhanden */

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementtyp DB_BLK wurde versucht auf einen nicht existierenden Baustein zuzugreifen.

Abhilfe: Datenbaustein im AG anlegen oder Parameter "bst" im Funktionsaufruf der CP-Anwendersoftware korrigieren.

`ERR_S5_ELM = 0x03;` /* Element nicht vorhanden */

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementzugriff mit Elementtyp DB_BLK wurde versucht auf Daten in einem DB zuzugreifen, die nicht vorhanden sind.

Abhilfe: Datenbaustein im AG entsprechend verlängern oder Parameter "adr" bzw. "len" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten bzw. Zählerstände mit einer Nummer > 127 zuzugreifen.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG wurde versucht auf Merker mit Nummer > 199 bei Elementgröße Byte, mit Nummer > 198 bei Elementgröße Wort oder mit Nummer > 196 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software auf Wertigkeit überprüfen.

Bei Einzelelementzugriff mit Elementtyp EB_ - oder AB_SNG wurde versucht auf das Prozeßabbild des E/A-Bereichs mit Nummer > 127 bei Elementgröße Byte, mit Nummer > 126 bei Elementgröße Wort oder mit Nummer > 124 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software auf Wertigkeit überprüfen.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht auf Elemente der P-Peripherie mit Nummer > 255 bei Elementgröße Byte, mit Nummer > 254 bei Elementgröße Wort oder mit Nummer > 252 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software auf Wertigkeit überprüfen.

`ERR_S5_SIZE = 0x04;` /* ungültige Elementgröße */

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten oder Zählerstände zuzugreifen, wobei der Parameter Elementgröße nicht auf Wortzugriff (WORD_ELM) gesetzt war.

Abhilfe: Parameter "size" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG wurde versucht, mit dem Parameter Elementgröße RBYTE_ELM auf Merker oder absolute Adressen zuzugreifen.

Abhilfe: Parameter "size" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht, mit dem Parameter Elementgröße SEMA_ELM oder RBYTE_ELM auf den Ein- bzw. Ausgängen im Prozeßabbild zuzugreifen.

Abhilfe: Parameter "typ" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht mit dem Parameter Elementgröße BIT_ELM, SEMA_ELM oder RBYTE_ELM auf die P-Peripherie zuzugreifen.

Abhilfe: Parameter "typ" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei lesenden Einzelelementzugriff mit Elementtyp ABS_SNG wurde versucht von absoluten Adressen mit Elementgröße SEMA_ELM zu lesen. Diese Zugriffsart ist unter absoluter Adressierung nur schreibend möglich! Wenn einzelne Bits gelesen werden sollen, ist die Elementgröße BIT_ELM zu verwenden.

Abhilfe: Parameter "typ" im Funktionsaufruf der CP-Anwender-Software korrigieren.

`ERR_S5_BIT = 0x05;` /* Bit-Nummer zu groß */

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG und der Elementgröße BIT_ELM oder SEMA_ELM wurde versucht, auf ein Merker- oder Absolutadress-Bit mit einer Bitnummer > 7 (15) zuzugreifen.

Abhilfe: Parameter "bit" im Funktionsaufruf der CP-Anwender-Software korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht auf ein E/A-Bit mit einer Bitnummer > 7 zuzugreifen.

Abhilfe: Parameter "bit" im Funktionsaufruf der CP-Anwender-Software korrigieren.

`ERR_S5_STRT = 0x06;` /* ungültige Anfangsadresse */

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde ein Blocktransfer über Bausteine vesucht, wobei die relative Anfangsadresse im Block > 32767 ist.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software korrigieren.

`ERR_S5_LEN = 0x07;` /* ungültige Blocklänge */

Bei Blockelementzugriff unter allen Elementtypen wurde ein Blocktransfer mit einer Länge > 504 Worten versucht.

Abhilfe: Parameter "len" im Funktionsaufruf der CP-Anwender-Software korrigieren.

```
ERR_S5_ADR      = 0x08; /* Adresse zu groß */
```

Bei Einzel- oder Blockelementzugriff mit Elementtyp ABS_SNG wurde versucht auf eine Adresse > FFFFh in einem AG der 115U-Reihe zuzugreifen. Die CPUs (bis CPU 944) haben jedoch nur 64 KByte Adressraum.

Abhilfe: Parameter "adr" im Funktionsaufruf der CP-Anwender-Software korrigieren.

```
ERR_S5_QVZ      = 0x09; /* QVZ/ADF im AG bei lesen/schreiben */
```

Es wurde versucht auf einen Adressbereich zuzugreifen, der physikalisch nicht vorhanden ist. Die AGs der 135 und 155-Reihe stellen diese Fehlermeldungen zu Verfügung. Ein AG der 115U-Reihe würde hierbei in den Stop-Zustand versetzt werden.

Abhilfe: Parameter "typ" bzw. "adr" im Funktionsaufruf der CP-Anwendersoftware korrigieren.

```
ERR_S5_944      = 0x0A; /* CPU 944: Baustein im Prog.Bank */
```

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde versucht auf einen Baustein zuzugreifen, der nicht in der DB-Bank liegt. (Betrifft nur CPU 944 vom AG-Typ 115U)

Abhilfe: Baustein im AG in DB-Bank anlegen (über BIB-Nr. 19285) oder Funktionsaufruf in der CP-Anwender-Software korrigieren.

1.3.6 Ablage der Prozeßabbilder in der Kachel 7

Das Prozeßabbild kann vom Anwender auch direkt ausgelesen werden. Folgende Übersicht zeigt wie die Kachel 7 aufgeteilt ist. Der direkte Zugriff ist sehr schnell:

```
Adresse in der
Kachel (hex)

Byte   0 Prozessabbild EB 0          -+
       .                              ¦
       .                              ¦ 128 Byte PAE 0-127
       .                              ¦
Byte 127 Prozessabbild EB 127        -+
Byte 128 Prozessabbild AB 0          -+
       .                              ¦
       .                              ¦ 128 Byte PAA 0-127
       .                              ¦
Byte 255 Prozessabbild AB 127        -+
Byte 256 Merkerbyte 0                -+
       .                              ¦
       .                              ¦ 256 Byte Merker 0-255
       .                              ¦
Byte 511 Merkerbyte 255              -+
Byte 512/513 Timer 0   (high/low)    -+
       .                              ¦
       .                              ¦ 128 Worte Timer 0-127
       .                              ¦
Byte 766/767 Timer 127 (high/low)    -+
Byte 768/769 Zähler  0 (high/low)    -+
       .                              ¦
       .                              ¦ 127 Worte Zähler 0-126
       .                              ¦
Byte 1020/1021 Zähler 126 (high/low) -+
Byte 1022 Zählbyte 1)                 + Zählbyte
Byte 1023 Interrupt auf CP auslösen
```

Hinweis

Alle Werte in dieser Kachel werden beim Aufruf des Hantierungsbausteins CP L/S aufgefrischt ohne einen Interrupt auszulösen (wenn dies am Formaloperanden des Hantierungsbausteins freigegeben ist). Nach jeder Auffrischung der Daten in der Kachel 7 erhöht der Hantierungsbaustein das Zählbyte um 1. An dem Zählbyte kann die CP erkennen, ob die Daten gültig sind und wie oft die Daten seit dem letzten Lesen aufgefrischt wurden. Gültig sind die Daten dann, wenn der Inhalt des Zählbytes im Bereich dual 1...255 liegt. Bei Überlauf beginnt das Zählbyte wieder bei 1.

Der Hantierungsbaustein *Synchron* setzt den laufenden Zähler, Adresse 3FEh in der Kachel 7 auf 0. Hieran erkennt die CP daß die Daten in der Kachel 7 momentan nicht gültig sind.

Es ist nicht erforderlich, diese Kachel von der CP aus zu löschen muß von der CP nicht gelöscht werden, wenn im Zählbyte (Adresse 3FEh der Kachel) eine 0 steht.

Auf diese Kachel greift der Hantierungsbaustein nur schreibend zu.

1.4 Betrieb des CP386COM unter WINDOWS

Ab der Version 2.2 der Tool-Diskette liegt eine Programmlibrary für MS-WINDOWS 3.1 mit folgenden Dateien vor:

Die Header-Datei CP386WIN.H und die OBJ-Datei CP386WIN.OBJ.

Die Datei CP386WIN.H enthält die erforderlichen Definitionen für den Betrieb unter WINDOWS.

Die Datei CP386WIN.OBJ enthält die Kommunikationsfunktionen unter WINDOWS. Die Funktionen werden, wie in Kap. 6.4 für DOS beschrieben, aufgerufen. (**Ausnahme:** CP_stat_AG)

Änderungen im Funktionsaufruf:

CP_stat_AG: CP_stat_AG (byte r) mit r = Auftragsnummer.

Neue Funktionen:

CP_init (void): Legt einen Datenbereich für die Kommunikation auf der Kachel 4 an und gibt einen Zeiger auf diesen Bereich zurück.

CP_exit (void): Gibt den Datenbereich wieder frei. Dieser Befehl muß am Programmende unbedingt ausgeführt werden.

Hinweis

In der SYSTEM.INI muß unter der Sektion [386Enh], zusätzlich zum Eintrag in der CONFIG.SYS, mit dem Befehl *EMMExclude* = ... der Kachelbereich von der WINDOWS-Speicherverwaltung ausgeklammert werden!

(Die gilt in allen Fällen, in denen die CP unter WINDOWS 3.1 läuft, da WINDOWS den Kachelbereich nicht selbständig ausklammert!)

Auf der Tool-Diskette 2.2 ist ein Demobeispiel für den Betrieb unter Windows enthalten.

2 Kopplung der SPS mit CP486COM

2 Kopplung der SPS mit CP486COM

2.1 Allgemeine Beschreibung

Der Datentransfer zwischen CP486 und SPS wird durch Hantierungsbausteine auf SPS-Seite und durch Software-Interrupts auf CP-Seite unterstützt. Folgende Routinen stehen zur Verfügung:

		Bedienung auf SPS-Seite	Bedienung auf CP-Seite
Kachel 2	CP-Auftrag: Daten von SPS lesen/schreiben **(CP486 aktiv)**	zyklisch aufgerufener Hantierungsbaustein (FB1)	Softwareinterrupt bei DOS Treiberaufruf bei WIN
Kachel 7	Prozeßabbild an CP übertragen	zyklisch aufgerufener Hantierungsbaustein (FB1)	Softwareinterrupt oder direkter Zugriff auf die Kachel

Tab. 2-1: Routinen

Folgende Datenstrukturen in der SPS können auf CP-Seite angesprochen werden:

- einzelne Elemente im Format Bit, Byte, Wort und Doppelwort, DB, DX, BA, BB, BT, BS, Merker, Eingänge, Ausgänge, Timer, Zähler
- Datenblöcke DB, DX, MB, T, Z, BA, BB, BT, BS, FB, FX, OB, PB, SB

Die nachfolgend beschriebenen Funktionen stehen ab CPX86-Version 3.00
(Software CP4-SW593 Version 3.00) und Hantierungsbaustein-Version 3.00
(CP4-SW977 und CP4-SW978 Version 3.00) zur Verfügung.

Das Programm CPX86 wird in der folgenden Beschreibung als COM-Treiber bezeichnet.

Folgende CPUs wurden getestet: CPU943B, CPU944, CPU945, CPU928, CPU928B, CPU946/947.

Getestet wurden die nachstehenden Formate:

Format	read	write	Bemerkung
Datenbit	X	X	
Datenwort linkes Byte	X	X	
Datenwort rechtes Byte	X	X	
Datenwort	X	X	
Datendoppelwort	X	X	
Datenwort (Block)	X	X	Länge in Wort angeben
Datendoppelwort (Block)	X	X	Länge in Wort angeben
BS-Bereich Bit	X	X	
BS-Bereich linkes Byte	X	X	
BS-Bereich rechtes Byte	X	X	
BS-Bereich Wort	X	X	
BS-Bereich Doppelwort	X	X	
BS-Bereich Wort (Block)	X	X	Länge in Wort angeben
Zähler Wort (1 Zähler)	X	X	

Format	read	write	Bemerkung
Zähler Doppelwort (2 Zähler)	X	X	Die Zähler werden gedreht
Zähler Wort (Block n-Zähler)	X	X	
Timer Wort (1 Timer)	X	X	
Timer Doppelwort (2 Timer)	X	X	Die Timer werden gedreht
Timer Wort (Block n-Timer)	X	X	
Merkerbit	X	X	
Merkerbyte	X	X	
Merkerwort	X	X	
Merkerdoppelwort	X	X	
Merkerbyte (Block)	X	X	Länge in Byte angeben
Merkerwort (Block)	X	X	Länge in Byte angeben
Eingangsbit	X	X	
Eingangsbyte	X	X	
Eingangswort	X	X	
Eingangsdoppelwort	X	X	
Eingangsbyte (Block)	X	X	Länge in Byte angeben
Eingangswort (Block)	X	X	Länge in Byte angeben
Ausgangsbit	X	X	
Ausgangsbyte	X	X	
Ausgangswort	X	X	
Ausgangsdoppelwort	X	X	
Ausgangsbyte (Block)	X	X	Länge in Byte angeben
Ausgangswort (Block)	X	X	Länge in Byte angeben
Peripheriebit	X	X	
Peripheriebyte	X	X	
Peripheriewort	X	X	
Peripheriedoppelwort	X	X	
Peripheriebyte (Block)	X	X	Länge in Byte angeben
Peripheriewort (Block)	X	X	Länge in Byte angeben
Absolutadresse Byte	X	X	Bei CPUs mit Wortadresse wird rechtes Byte gelesen
Absolutadresse Wort	X	X	
Absolutadresse Doppelwort	X	X	
Absolutadresse Block	X	X	Die Länge der Daten in Byte angeben
FB - Block	X		Die Länge der Daten in Byte angeben
OB - Block	X		Die Länge der Daten in Byte angeben
PB - Block	X		Die Länge der Daten in Byte angeben
SB - Block	X		Die Länge der Daten in Byte angeben

Tab. 2-2: Übersicht der getesteten Formate

Laufzeit des FBs in den einzelnen CPUs in ms:

CPU Bezeichnung	Leerlauf	Übertragen von 100 DW	Übertragen von 200 DW
944	0,2	1,6	2,2
928	1,0	4,5	5,2
928B	0,2	1,8	3,1
945	0,02	0,55	1,0
946/947	0,125	1,0	1,6

Tab. 2-3: Laufzeit des FBs in den einzelnen CPUs

Änderungen zum Treiber V2.0

Aus der Konstanten PB_SNG wurde PY_SNG.

Aus der Konstanten QB_SNG wurde QY_SNG.

Die Konstante BLOCK_ELM wird nicht mehr unterstützt, es ist die Konstante B_BLOCK zu verwenden.

Die Konstanten mit der Kennung _SNG und _BLK sind gleichrangig, d.h. bei Blockaufträgen und Einzelaufträgen können sowohl SNG als auch BLK Konstanten angegeben werden.

Das Anwenderprogramm muß gemäß der neuen Konstantenliste abgeändert und neu kompiliert werden, wenn von der Peripherie, Q-Peripherie, Merkerblöcken und Absolutadresse gelesen und geschrieben wird.

Werden Blockaufträge mit mehr als 245 Worten bearbeitet, so kann der Treiber für einen SPS Zyklus "kachelblockiert" anzeigen. In diesem Fall ist der Auftrag erneut zu starten.

Bei Mehrprozessorbetrieb werden die Aufträge auf CP-Seite mit der Funktion CP_CALL definiert. Auf der SPS-Seite ist der FB 10 (Kachel lesen/schreiben) im OB 1 aufzurufen und in den Anlauf-OBs wird der FB 20 (Synchron) benötigt. Im Mehrprozessorbetrieb kann eine Kommunikation von einer CP mit bis zu 4 CPUs realisiert werden.

Im Singleprozessorbetrieb kann mit den Funktionen CP_read_AG, CP_write_AG, CP_readn_AG, CP_writen_AG oder mit der Funktion CP_CALL gearbeitet werden. Wird die Funktion CP_CALL verwendet ist als CPU-Nr. immer 0 anzugeben. Im SPS-Programm kann der FB 1 bzw. der FB 10 und zur Synchronisation der FB2 bzw. FB12 verwendet werden.

2.2 Installation der Kachelsoftware

2.2.1 SPS-Seite: Hantierungsbausteine

Für die Kommunikation mit der CP486 müssen die Hantierungsbausteine FB1/FB10 und FB2/FB12 in der SPS geladen werden. Der Hantierungsbaustein FB1 wird im OB1 aufgerufen, der FB2 in den Neustartbausteinen (OB20, OB21 und OB22).

2.2.1.1 FB1/FB10 (CP-L/S), CP lesen und schreiben

Bez.	Format	Beschreibung
ANSS	KY	Anzahl der Aufträge
PAA	KF	Kennung Prozeßabbild
PAFE	MB	Merkerbyte für Fehlermeldungen

Tab. 2-4: Parameterliste für den Aufruf von FB1/FB10

ANSS AN Anzahl der Aufträge, die maximal bei einem Hantierungsbaustein-Aufruf auf der Kachel abgearbeitet werden sollen

SS Nummer der Basiskachel

PAA Kennung der Prozeßabbilder beim Aufruf des Hantierungsbausteins auf der Kachel aktualisieren

= 0 Prozeßabbilder werden nicht aktualisiert

≠ 0 Prozeßabbilder werden aktualisiert

Der angegebene Wert, der auf das Prozeßabbild übertragen werden soll, ist die Kachelnummer +1. Zulässige Kachelnummern sind 4 - 7.

PAFE Fehlerrückmeldung des Hantierungsbausteins

= 0 es ist kein Fehler aufgetreten

≠ 0 es ist ein Fehler aufgetreten. Die Fehlernummer wird im PAFE-Byte übergeben

1 Anzahl der Aufträge, die maximal bei einem Hantierungsbaustein-Aufruf abgearbeitet werden sollen ist 0.

2 Anzahl der Aufträge, die maximal bei einem Hantierungbaustein-Aufruf abgearbeitet werden sollen ist größer als 127.

3 Die Basiskachelnummer ist nicht durch 8 teilbar

5 Die Kachel ist noch nicht von der CP synchronisiert.

9 Kachel für Prozeßabbild nicht im gültigem Bereich.

Benutzte Schmiermerker: MB200-MB255

2.2.1.2 FB2/FB12 (SYNCHRON), Synchronisation CP und AG

Bez.	Format	Beschreibung
SSNR	KF	Nummer der Basiskachel
WART	KF	Art der Synchronisation
PAA	KF	Kennung Prozeßabbild
PAFE	BY	Merkerbyte für Fehlermeldungen

Tab. 2-5: Parameterliste für den Aufruf von FB2/FB12

SSNR Nummer der Basiskachel

WART
- = 0 Der FB-SYNCHRON wartet nicht bis der CP jede einzelne Kachel synchronisiert hat
- ≠ 0 Der FB-SYNCHRON wartet bei jeder einzelnen Kachel bis der CP diese synchronisiert hat

PAA Kachelnummer auf der das Prozeßabbild abgelegt werden soll. Der gültige Bereich liegt zwischen 4..7.

PAFE Fehlerrückmeldung des Hantierungsbausteins
- = 0 es ist kein Fehler aufgetreten
- ≠ 0 es ist ein Fehler aufgetreten:
  - 3 Basiskachelnummer ist nicht durch 8 teilbar.

Benutzte Schmiermerker: MB200-MB255

2.2.2 CP486-Seite: MSDOS-Treiber-Programm

Zur Kommunikation zwischen AG und CP über die Kacheln muß in der CP486 ein spezieller Kommunikationstreiber geladen werden. Der Treiber ist speziell zur Kommunikation mit den VIPA-Hantierungsbausteinen ausgelegt. Er stellt einfach handzuhabende Funktionen bereit, d.h. der Anwender muß keine Details über Aufbau und Wirkungsweise der Kacheln besitzen. Die Software zur Bedienung aller Kacheln ist im Treiber enthalten. Momentan werden Kachel 2 (AG passiv, CP aktiv) und Kachel X (Prozeßabbild) unterstützt. Der Treiber stellt automatisch alle Funktionen für alle Kacheln bereit, eine weitere Konfiguration für die einzelnen Kacheln ist nicht notwendig.

2.2.2.1 Installation des Treibers

Der Treiber wird beim Start der CP486 geladen, d.h. bereits in der CONFIG.SYS. Beachten Sie bitte, daß im AG in der Regel während des Neustartes die Hantierungsbausteine zur Synchronisation aufgerufen werden. Diese warten, wenn der Parameter WART nicht gesetzt ist, nur eine bestimmte Zeit auf eine Reaktion der CP. Solange der Treiber auf CP-Seite nicht gestartet wurde, wird im AG "nicht synchronisiert" angezeigt. Der Aufruf sieht folgendermaßen aus:

```
DEVICE = CP486COM.EXE
```

Der Treiber belegt ca. 26KB des Arbeitsspeichers (Programm und Daten) und kann immer nur einmal geladen sein. Jeder weitere Aufruf bewirkt die Ausgabe, daß der Treiber bereits installiert ist. Der Treiber kann nur durch erneutes Booten der CP aus dem Arbeitsspeicher entfernt werden (Deinstallation).

Der COM-Treiber ist ausschließlich für CP486 Baugruppen der VIPA GmbH entwickelt und kann nur auf diesen Systemen installiert werden. Wird der Treiber auf anderen PC-Systemen geladen, auch mit i386 oder i486 Prozessor, so ist das Verhalten des Rechners undefiniert. In der Regel wird erkannt, daß der Rechner kein CP ist.

2.2.2.2 Belegte Interrupts bei DOS

Der Treiber verwendet mehrere Software-Interrupts zum Betrieb und zur Kommunikation mit der Anwendungssoftware in der CP-Baugruppe:

- INT 1Ch Timer-Interrupt

  Für regelmäßige, zyklische Überprüfungen der Kachel wird der sog. Ticker-Interrupt mit der Nummer 1Ch sowie der sog. DOS-Idle Interrupt verwendet. Damit kann z.B. regelmäßig überprüft werden, ob das AG die Kacheln neu synchronisieren will. Nach Ausführung der CP-spezifischen Funktionen, wird die ursprüngliche Interrupt-Behandlungsroutine (Interrupt Service Routine) aufgerufen.

- INT 74h (IRQ 12):

  Der CP486 verwendet den Hardware-IRQ 12, der den Software-Interrupt 74h belegt. Dieser Interrupt wird immer ausgelöst, wenn im AG BASP aktiv ist oder auf die höchste Speicherzelle jeder Kachel (Byte1023) vom AG geschrieben wurde. Die Verwendung des Interrupts ermöglicht eine schnelle Reaktion der CP auf Anforderungen durch die SPS. Nach Bearbeitung der CP-spezifischen Funktionen wird die ursprüngliche Interrupt-Behandlungsroutine aufgerufen. Dadurch können auch mehrere Geräte den IRQ 12 verwenden.

- INT 78h Service-Interrupt:

  Dieser Interrupt ist von der Anwendersoftware in der CP zum Aufruf der Treiberfunktionen zu verwenden, z.B. Datenaustausch mit dem AG über Kacheln 2. Durch entsprechende Belegung der Prozessor-Register können unterschiedliche Funktionen ausgelöst werden. Wird der Int 78 mit Register-Werten aufgerufen, die für die CP486-Kommunikation ungültig sind, so wird die ursprüngliche Interrupt-Behandlungsroutine aufgerufen.

2.2.3 Unterschiedliche Darstellung von Daten im Speicher

Bei der Übertragung von Daten zwischen CP und AG muß die unterschiedliche Darstellung von Worten und Doppelworten (Langworten) auf CP und AG berücksichtigt werden.

Im CP sind Datenworte in anderer Form im Speicher abgelegt als im AG. Das höherwertige (High-Byte) und das niederwertige Byte (Low-Byte) sind vertauscht abgespeichert. Bei Doppelworten sind alle 4 Bytes genau im umgekehrter Reihenfolge abgespeichert. Tauschen AG und CP Daten vom Typ Wort oder Doppelwort aus, so muß irgendwann eine Vertauschung vorgenommen werden, da sonst nach der Übertragung falsche Daten vorliegen. Soweit dies sinnvoll möglich ist, nimmt der COM-Treiber die Anpassung der Daten automatisch vor.

Bei Kachel 2 und Kachel 7 führt der Treiber in allen Fällen automatisch eine Vertauschung durch.

- Bei der Übertragung von Bytes findet keine Vertauschung statt.
- Bei der Übertragung von Worten werden High- und Low-Byte vertauscht.
- Bei der Übertragung von Langworten werden alle 4 Bytes in ihrer Reihenfolge umgedreht.

Darstellung von Daten im AG

Adressen		
	Byte	Darstellung Byte
Adresse n	High-Byte	
Adresse n+1	Low-Byte	Darstellung Wort
Adresse n	High-Byte High-Word	
Adresse n+1	Low-Byte High-Word	
Adresse n+2	High-Byte Low-Word	Darstellung Doppelwort
Adresse n+3	Low-Byte Low-Word	

Darstellung von Daten im CP

Adresse n		
	Byte	Darstellung Byte
Adresse n	Low-Byte	
Adresse n+1	High-Byte	Darstellung Wort
Adresse n	Low-Byte Low-Word	
Adresse n+1	High-Byte Low-Word	
Adresse n+2	Low-Byte High-Word	Darstellung Doppelwort
Adresse n+3	High-Byte High-Word	

2.3 CP486-Aufträge für die SPS (Funktionen für Kachel 2 und 7)

2.3.1 Übersicht

Das Treiberprogramm CP486 bedient die Aufträge, die von der CP ausgelöst werden. Für die Kacheln 2 und 7 stellt der Treiber eine Reihe von Funktionen zur Verfügung. Mit diesen Funktionen können aus einem auf der CP486 laufenden Anwenderprogramm heraus, Daten aus dem AG gelesen bzw. in das AG geschrieben werden. Alle Funktionen werden mit Software-Interrupts 78h aufgerufen. Ab Software Version 3.10 existiert ein erweiterter Aufruf, der eine Struktur für die Datenübergabe enthält.

Die Übergabe und die Rückgabe von Parametern bei Aufruf des Interrupts erfolgt in den Prozessor-Registern. Die Belegung der Register ist in der Beschreibung der Funktionen enthalten. Für Turbo-Pascal, Turbo-C und C++ sowie Microsoft-C sind Schnittstellen implementiert. Es gibt Funktionen zum Lesen und Schreiben von Daten in das AG, zur Status-Abfrage und Abbruchfunktionen. Die Funktionen für die Datenübertragung werden anhand ihrer Übertragungsart unterschieden. Man unterschiedet Einzelelementübertragung und Blockübertragung. Die einzelnen Funktionen werden als "Aufträge" abgewickelt. Bei Aufruf einer Funktion wird eine Auftragsnummer zurückgegeben, unter der der Status der Bearbeitung abgefragt werden kann. In Kachel 2 können bis zu 90 Aufträge gleichzeitig in Bearbeitung sein.

Folgende Funktionen stehen zur Verfügung

Kachelnr.	Funktionsnr.	Funktion
keine	$00	Status-Abfrage
2	$21	Lesen/Schreiben eines Einzelelements in/aus dem AG
2	$21	Lesen/Schreiben eines Blocks aus dem AG
2	$20	Statusabfrage zu Lese/Schreib-Auftrag
2	$28	Abbrechen aller Lese/Schreib-Aufträge
2	$7f	Strukturaufruf
7	$70	Statusabfrage für Prozeßabbild
7	$71	Lesen des Prozeßabbild-Bereichs

Tab. 2-6: Funktionsbeschreibung

2.3.2 Treiber-Funktionen über Software-Interrupt

2.3.2.1 CP-Status Abfrage

Diese Funktion kann von der Anwendersoftware, zur Feststellung welcher CPx86 Softwaretreiber geladen ist, verwendet werden.

Register	IN	high	OUT	low
AX	$00	$C386		
BX				
CX				
DX				

AX C386h Kennung, daß CP-Software geladen ist.

Ist der Treiber geladen, gibt die Funktion den Wert 0 zurück, andernfalls den Wert -1.

Diese Funktion führt keine Initialisierungen an den Kacheln aus.

2.3.2.2 Lesen eines Einzelelements aus dem AG

Mit dieser Funktion können Sie ein einzelnes Datum (Bit, Byte, Wort, ...) aus dem AG lesen. Die Funktion stößt den Auftrag an und kehrt sofort wieder zum Aufrufer zurück. Die Daten selbst können Sie durch Aufruf der Funktion "Status-Abfrage" lesen (siehe Kapitel 2.3.2.6).

Register	high	In	low	Out
AX	$21		typ	status
BX	size		bst	
CX	adr			
DX	-		bit	

Parameter

typ	Elementtyp der Daten bei Einzelelement im AG (DB, MB...) s.Kap. 2.3.3.11.2	
size	Kennung der Elementgröße (Bit, Byte ...) siehe Kapitel 2.3.3.11.1	
bst	Bausteinnummer, nur bei Elementtyp DB oder DX. Bei Elementtyp "absolut" stehen hier die höchstwertigen Bits von adr.	
adr	Anfangsadresse im Bereich	
bit	Bit-Nummer wenn die Elementgröße size Bit ist.	
status	< 0	Fehlernummer, weil Fehler aufgetreten. Fehlernummern des AG werden mit FF00h aufaddiert
	1..255	Auftragsnummer unter der der Status abgefragt werden kann

Hinweis

Diese Funktion gibt noch keine Daten zurück! Wurde der Auftrag "Fertig ohne Fehler", so können die Daten durch einen Aufruf der Funktion Status-Abfrage abgeholt werden.

Um sicherzustellen, daß ein fertiger Lese-Auftrag nicht durch einen neuen Auftrag überschrieben wird bevor die Daten abgeholt werden, wird der Auftrag blockiert. Nach Starten eines Auftrags muß dessen Status solange abgefragt werden, bis der Auftrag "Fertig mit Fehler" oder "Fertig ohne Fehler" ist. Ist der Auftrags-Status "Fertig ohne Fehler" so werden die Daten an die angegebene Adresse im CP kopiert. Erfolgt keine Statusabfrage bleibt der Auftrag blockiert und es können keine weiteren Lese-Aufträge mehr gestartet werden, selbst wenn alle Aufträge in der Kachel beendet sind.

2.3.2.3 Lesen eines Blocks aus dem AG

Mit dieser Funktion können Sie einen ganzer Datenblock aus dem AG lesen. Die Funktion stößt den Auftrag an und kehrt sofort wieder zum Aufrufer zurück. Mit der Funktion "Status-Abfrage" haben Sie Zugriff auf die Daten (siehe Kapitel 2.3.2.6).

Register	high	In	low	Out
AX	$21	typ		status
BX	size	bst		
CX	adr			
DX	len			

Parameter

typ	Elementtyp der Daten bei Einzelelement im AG (DB, MB...) (siehe Kapitel 2.3.3.11.2)	
size	Elementgröße der Blockdaten. Kennung, ob die Einzelbytes vertauscht werden:	
	07h	Datenblock von Bytes (kein Vertauschen)
	17h	Datenblock von Worten (Vertauschen von High- und Low-Byte)
	27h	Datenblock von Doppelworten (Vertauschen aller 4 Bytes)
bst	Bausteinnummer, Elementtyp DB, DX , FX. Bei Elementtyp "absolut" stehen hier die höchstwertigen Bits von adr.	
adr	Anfangsadresse im Bereich	
len	Anzahl der Daten in Worten bzw. Bytes. Die Wertigkeit in der SPS bestimmt die Bytes oder Wortlänge, z.B. Timer, Zähler, DW Angabe der Länge in Worten oder bei Merker und Ausgängen in Bytes.	
status	< 0	Fehlernummer, weil Fehler aufgetreten
	1..255	Auftragsnummer unter der Status abgefragt werden kann

Hinweis

Damit bei der Übertragung die Anpassung der Daten automatisch durchgeführt werden kann, muß die Art der Daten im Block (Bytes, Worte, Doppelworte) angegeben werden. Ein Block kann nur Daten der gleichen Art enthalten. Bei Worten und Doppelworten wird für jedes einzelne Datum die Vertauschung der Bytes entsprechend vorgenommen.

Diese Funktion gibt noch keine Daten zurück! Wurde der Auftrag "Fertig ohne Fehler", so können die Daten durch einen Aufruf der Funktion Status-Abfrage abgeholt werden.

Um sicherzustellen, daß ein fertiger Lese-Auftrag nicht durch einen neuen Auftrag überschrieben wird, bevor die Daten abgeholt werden, wird der Auftrag blockiert. Nach dem Starten eines Auftrags muß dessen Status solange abgefragt werden, bis der Auftrag "Fertig mit Fehler" oder "Fertig ohne Fehler" ist. Ist der Auftrags-Status "Fertig ohne Fehler" so werden die Daten an die angegebene Adresse im CP kopiert. Erfolgt keine Status-Abfrage bleibt der Auftrag blockiert und weiteren Lese-Aufträge können nicht mehr gestartet werden.

2.3.2.4 Schreiben einer Variablen in das AG

Mit dieser Funktion kann ein einzelnes Datum (Bit, Byte, Wort,...) in den Speicher des AG geschrieben werden. Beim Aufruf dieser Funktion ist die Adresse einer zu schreibenden Variable anzugeben. Die Funktion überträgt deren Wert in die Kachel und wartet nicht bis das AG die Daten abholt, sondern kehrt sofort wieder zum Aufrufer zurück.

Register	high	In	low	Out
AX	$21		typ	status
BX	size		bst	
CX	adr			
DX	-		bit	
SI	offset			
DS	segment			

Parameter

typ	Elementtyp der Daten bei Einzelelement im AG (DB, MB ...) (siehe Kapitel 2.3.3.11.2)	
size	Kennung der Elementgröße (Bit, Byte, ...), siehe Kapitel 2.3.3.11.1	
bst	Bausteinnummer, nur bei Elementtyp DB oder DX Bei Elementtyp "absolut" stehen hier die höchstwertigen Bits von adr.	
adr	Anfangsadresse im Bereich	
bit	Bit-Nummer wenn die Elementgröße size Bit ist.	
offset	Offset der Variablen-Adresse (im CP)	
segment	Segment der Variablen-Adresse (im CP)	
status	< 0	Fehlernummer, weil Fehler aufgetreten
	1-255	Auftragsnummer unter der der Status abgefragt werden kann

Hinweis

Bei einem Schreib-Auftrag wird die Kachel, wie bei Lese-Aufträgen, blockiert. Deshalb sollte ebenfalls der Auftrags-Status solange abgefragt werden, bis der Auftrag "Fertig mit Fehler" oder "Fertig ohne Fehler" ist.

Je nach Elementgröße ist der Zeiger auf die Daten im CP unterschiedlich zu interpretieren.

- Bit:
  Der Zeiger ist die Adresse eines Bytes. Das Bit wird von Bit-Nummer 0 gelesen.
- Byte, linkes Byte, rechtes Byte:
  Der Zeiger ist die Adresse eines Bytes. Das Byte wird von der Speicherzelle gelesen.
- Wort:
  Der Zeiger ist die Adresse eines Wortes. High- und Low-Byte werden bei der Übertragung vertauscht.
- Doppelwort/Langwort:
  Der Zeiger ist die Adresse eines Langwortes. Das Langwort wird gelesen und die Reihenfolge aller 4 Bytes wird umgedreht.

2.3.2.5 Schreiben eines Blocks in das AG

Mit dieser Funktion können Sie einen ganzen Datenblock in das AG übertragen. Beim Aufruf der Funktion ist ein Zeiger auf einen Datenblock anzugeben. Die Funktion schreibt die Daten in die Kachel und kehrt sofort wieder zum Aufrufer zurück. Sie wartet nicht, bis das AG die Daten übernommen hat. Der Datenblock ist anschließend in der CP wieder frei verfügbar und könnte z.B. überschrieben werden.

Register	high	In	low
AX	$21	typ	
BX	size	bst	
CX	adr		
DX	len		
SI	offset		
DS	segment		

Parameter

typ	Elementtyp der Daten bei Blockelement im AG (DB, MB) (siehe Kapitel 2.3.3.11.3)	
size	Elementgröße der Blockdaten. Kennung ob Einzelbytes vertauscht werden.	
	07h	Datenblock von Bytes
	17h	Datenblock von Worten
	27h	Datenblock von Doppelworten
bst	Bausteinnummer, nur bei Elementtyp DB, DX, FB. Bei Elementtyp "absolut" stehen hier die höchstwertigen Bits von adr.	
adr	Anfangsadresse im Bereich	
len	Anzahl der Daten in Worten bzw. Byte (siehe Lese-Block)	
offset	Offset der Block-Adresse (im CP)	
segment	Segment der Block-Adresse (im CP)	
status	< 0	Fehlernummer, weil Fehler aufgetreten
	1..255	Auftragsnummer unter der der Status abgefragt werden kann

Hinweis

Damit bei der die Anpassung der Daten automatisch durchgeführt werden kann, muß die Art der Daten im Block (Bytes, Worte, Doppelworte) angegeben werden. Ein Block kann nur Daten der gleichen Art enthalten. Bei Worten und Doppelworten wird für jedes einzelne Datum die Vertauschung der Bytes entsprechend vorgenommen.

Ein Block-Schreib-Auftrag blockiert die Kachel. Nach Auftragsstart muß der Status solange abgefragt werden, bis der Auftrag "Fertig mit Fehler" oder "Fertig ohne Fehler" ist. Ist der Auftrags-Status "Fertig ohne Fehler" so wurden die Daten an die angegebene Adresse im AG kopiert. Erfolgt keine Status-Abfrage bleibt der Auftrag blockiert und weiteren Schreib-Aufträge können nicht mehr gestartet werden.

2.3.2.6 Auftrags-Status lesen

Mit dieser Funktion können Sie den Status eines vorher gestarteten Auftrags abfragen. Bei Lese-Auftrags-Variablen und Block-Lese-Aufträgen werden mit dieser Funktion auch die Daten an eine angegebene Adresse in der CP kopiert, wenn der Auftrags-Status "Fertig ohne Fehler" ist.

Register	high	**In**	low	**Out**
AX	$20	a_nr		status
SI	offset			
DS	segment			

Parameter

fn	Funktionsnummer für Status-Abfrage	
a_nr	Auftragsnummer	
offset	Offset der Daten-Adresse im PC. offset ist nur bei Lese-Aufträgen (Variablen und Block) anzugeben.	
segment	Segment der Daten-Adresse im PC. segment ist nur bei Lese-Aufträgen (Variablen und Block) anzugeben.	
status	< 0	Auftrag "Fertig mit Fehler" Fehlermeldungen des AG werden mit FF00h aufaddiert.
	1	Auftrag noch "in Bearbeitung"
	2	Auftrags-Status "undefiniert"
	3	Auftrag "Fertig ohne Fehler"

Vorgehensweise bei Status-Abfrage

Ist der Auftrag noch "in Bearbeitung", so muß die Status-Funktion noch solange aufgerufen werden, bis sich der Status ändert.

- Bei Schreib-Aufträgen: Ist der Auftrag "Fertig ohne Fehler", so wurden die Daten in das AG geschrieben.
- Bei Lese-Aufträgen: Ist der Auftrag "Fertig ohne Fehler" und wurde ein Zeiger für die Daten angegeben, so werden die Daten in die angegebene Adresse im CP kopiert. Dabei wird je nach angegebener Datengröße ggf. eine Vertauschung der Bytes vorgenommen.
- Ist der Auftrags-Status "undefiniert" so wurde der Auftrag bereits früher beendet, aber noch nicht durch einen neuen Auftrag überschrieben. Handelt es sich bei dem Auftrag um einen Lese-Auftrag und wurde ein Zeiger ungleich NULL angegeben, so werden die Daten in die angegebene Zieladresse kopiert.
- Ist der Auftrag "Fertig mit Fehler" so wurde der Auftragsblock freigegeben, falls es sich um einen Lese-Auftrag oder Block-Auftrag handelt.

- Bei einem Lese-Auftrag für einzelne Variable ist je nach Elementgröße der Zeiger unterschiedlich zu interpretieren:
    - Bit :
      Der Zeiger ist die Adresse eines Bytes. Das Bit wird auf Bit-Nummer 0 geschrieben, das ganze Byte wird dabei überschrieben.
    - Byte, linkes Byte, rechtes Byte:
      Der Zeiger ist die Adresse eines Bytes. Das Byte wird in die Speicherzelle geschrieben.
    - Wort:
      Der Zeiger ist die Adresse eines Worts. High- und Low-Byte werden bei der Übertragung vertauscht.
    - Doppelwort/Langwort:
      Der Zeiger ist die Adresse eines Langworts. Die Reihenfolge aller 4 Bytes wird umgedreht und das Langwort wird geschrieben.

- Bei einem Lese-Auftrag für einen Datenblock ist je nach Elementgröße der Zeiger unterschiedlich zu interpretieren:
    - Byte:
      Der Zeiger ist die Adresse eines Blocks von Bytes. Die einzelnen Bytes werden unverändert aus dem AG übertragen.
    - Wort:
      Der Zeiger ist die Adresse eines Blocks von Worten. Für jedes einzelne Wort werden bei der Übertragung High- und Low-Byte vertauscht.
    - Doppelwort/Langwort:
      Der Zeiger ist die Adresse eines Blocks von Langworten. Für jedes einzelne Langwort werden bei der Übertragung alle 4 Bytes in ihrer Reihenfolge umgedreht.

2.3.2.7 Alle Aufträge einer Kachel abbrechen

Register	high	**In** low	**Out**
AX	`fn`		`status`

Parameter

`fn`	Funktionsnummer für Status-Abfrage	
	$28	Abbruch aller Lese-Aufträge
`status`	< 0	Fehlernummer, weil Fehler aufgetreten
	0	alle Aufträge wurden abgebrochen.

Hinweis

Mit dieser Funktion können alle noch nicht beendeten Schreib- und Lese-Aufträge abgebrochen werden.

Es muß keine Unterscheidung zwischen Variablen- und Blockaufträgen vorgenommen werden. Auch wenn keine Aufträge in der Kachel mehr aktiv waren, kehrt die Funktion mit dem Returnwert 0 zurück.

2.3.2.8 Prozeßabbild-Status lesen

Register	high	**In** low	**Out**
AX	$70	-	`status`

Parameter

`status`	0	kein Prozeßabbild verfügbar
	1..255	aktueller Prozeßabbild-Zähler

Hinweis

Mit Hilfe dieser Funktion kann der aktuelle Wert des Prozeßabbild-Zählers (Kachel 7 Adresse 3FEh) gelesen werden. Wenn der Wert 0 ist, ist kein Prozeßabbild verfügbar.

2.3.2.9 Prozeßabbild-Bereich lesen

Mit dieser Funktion können Sie einen Bereich des aktuellen Prozeßabbildes lesen. Bei einem Zugriff auf die einzelnen Bereiche wird die Länge des Bereichs überwacht.

Register	high	In	low	Out
AX	$71	typ		status
BX	adr			
CX	len			
SI	offset			
DS	segment			

Parameter

typ	Elementtyp der Daten des Prozeßabbildes (EB, MB)	
adr	Anfangsadresse im Bereich	
len	Anzahl der Daten in Bytes oder Worten (je nach typ)	
offset	Offset der Daten-Adresse im PC	
segment	Segment der Daten-Adresse im PC	
status	< 0	Fehlernummer, weil Fehler aufgetreten
	= 0	Prozeßabbild nicht verfügbar
	> 0	Zähler für Prozeßabbild (wie bei Statusfunktion)

Hinweis

Die Längenangabe erfolgt bei Timer- und Zähler-Zugriffen in Worten, bei allen anderen Typen in Bytes. Beim Transfer von Timer- und Zähler-Worten werden High- und Low-Byte getauscht, sodaß die Daten korrekt im PC verarbeitet werden können.

Wird der Typ "absolut" angegeben, so kann ein beliebiger Ausschnitt des Prozeßabbildes bereichsübergreifend gelesen werden. Die Längenangabe erfolgt in Bytes. Wird ein Bereich der Timer und Zähler gelesen, so werden High- und Low-Byte getauscht.

Nr.	Typ
06	Zähler (Länge in Worten anzugeben)
07	Timer (Länge in Worten anzugeben)
08	Merker (Länge in Bytes)
09	EB (Länge in Byte)
0A	AB (Länge in Byte)
0F	Absoluter Zugriff auf Prozeßabbild (Länge in Byte)

Tab. 2-7: Elementtypen bei Prozeßabbild

2.3.2.10 Fehlernummern der CP für Kacheln 2, 3 und 7

hex	dez.	Beschreibung
FFFFh	-1	ungültiger Elementtyp
FFFEh	-2	Längenfehler (z.B. Adresse zu groß, Bit-Nummer zu groß)
FFFDh	-3	ungültige Elementgröße (falscher Wert bei Einzel- oder Blockauftrag)
FFFCh	-4	Elementtyp bei dieser CPU nicht möglich
FFFBh	-5	Kachel voll, es kann kein neuer Auftrag eingetragen werden Auftrag und ein Blockauftrag soll gestartet werden.
FFFAh	-6	Zugriff auf Kachel für 10 sec nicht möglich (AG ist vermutlich im Stop)
FFF9h	-7	Auftrag/Kachel ist noch blockiert (Auftrags-Status wurde nicht abgefragt, um Auftrag zu deblockieren).
FFF8h	-8	falsche Auftragsnummer bei Status-Funktion (z.B. Auftragsnr. > 255)
FFF7h	-9	fehlerhafter Quelldatenzeiger (Adresse NULL wurde bei Schreib-Auftrag angegeben)
FFF6h	-10	Auftrag nicht in Bearbeitung (unbenutzt!)
FFh	255	Ungültiger Funktionsaufruf
EEh	238	Auftrag bei Initialisierung abgebrochen

Tab. 2-8: Fehlernummern der CP für Kacheln 2, 3 und 7

2.3.3 Schnittstelle für Turbo-Pascal (ab Version 4.0)

Für Funktionsaufrufe des COM-Treibers aus Pascal-Programmen heraus, wurde eine Turbo-Pascal-Unit erstellt, die alle Funktionen des Service-Interrupts INT 78 zur Verfügung stellt. Zu jeder Funktion des Treibers ist eine entsprechende Pascal-Prozedur definiert, die die Versorgung der Register, den Aufruf des Interrupts und die Rückgabe der Werte durchführt. So können auch Anwender, die nicht mit systemnaher Programmierung auf PC vertraut sind, alle Möglichkeiten des Treibers nutzen.

Alle erforderlichen Funktionen, Datentypen und Konstanten sind in der Unit CP486LIB enthalten. Bei Verwendung in einem Pascal-Programm muß diese mit "USES CP486Lib" in das Anwenderprogramm eingebunden werden. Es muß auch dafür Sorge getragen werden, daß sich die übersetzte Unit CP486LIB.TPU in dem Verzeichnis befindet, in dem Turbo-Pascal nach Units sucht. Die Einstellung erfolgt über die Menüpunkte "Optionen | Directories | EXE & TPU-Verzeichnis" (vgl. Handbuch oder Hilfe-Funktionen zu Turbo-Pascal).

Die folgenden Abschnitten geben nur einen kurzen Überblick der einzelnen Funktionen. Eine genaue Beschreibung mit allen wichtigen Informationen entnehmen Sie bitte den Funktionsbeschreibungen der vorhergehenden Abschnitte.

2.3.3.1 Funktion CP-Status-Abfrage

```
FUNCTION CP_Info (VAR inforec : CP486InfoRec) : INTEGER;
```

Datenstrukturen

```
TYPE CP486InfoRec =
       RECORD
          CP_id : WORD;               (* Kennung: CP486 Wert = $C386 *)
          VGA_ver, BIOS_ver : BYTE;   (*                             *)
          DRV_ver: BYTE;              (*                             *)
          CPU_AG : BYTE;              (*                             *)
          CP_reg, S5_reg : BYTE;      (*                             *)
       END;
```

Datenstruktur für die allgemeine Status-Info Funktion. Die Komponenten werden entsprechend den Werten der CP486 besetzt.

Diese Funktion ruft die Treiber-Funktion "allgemeine Status-Information" auf. Dieser Funktion geht eine Installationsprüfung des Treibers voraus. Ist der Treiber nicht installiert, gibt die Funktion den Wert -1 zurück. Bei installiertem COM-Treiber, wird der Wert 0 zurückgegeben. Die Komponente CP_id enthält als Kennung immer den Wert $C386, die anderen Komponenten werden nicht aktualisiert.

2.3.3.2 Lesen eines Einzelelements aus dem AG

```
FUNCTION CP_read_AG (size, typ, bst : BYTE; adr : longint;
                     bit : BYTE):INTEGER;
```

Parameter

size	Elementgröße von Einzelelementen (siehe Kapitel 2.3.3.11.1)
typ	Elementtyp für Einzelelemente (siehe Kapitel 2.3.3.11.2)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
bit	Bit-Nummer

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Lesen eines Einzelelementes aus dem AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.3.11 beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden, andernfalls kann z.B. die Kachel blockiert werden:

```
VAR   a_nr,                    (* Auftragsnummer für Lese-Auftrag *)
      stat : INTEGER;          (* momentaner Auftrags-Status      *)
      wert : BYTE;             (* aus dem AG gelesener Wert       *)
BEGIN

(* Auftrag starten *)
   a_nr := CP_read_AG(LBYTE_ELM, DB_SNG, 10, 1, 0);

   IF a_nr < 0                 (* Fehler ist aufgetreten          *)
   THEN WriteLn('Auftrag wurde beendet mit Fehler: ', a_nr)

   ELSE  BEGIN                 (* a_nr enthält die Auftragsnummer *)
      REPEAT
            stat := CP_stat_AG(a_nr, Addr(wert)); (* Auftrags-
                                                Status/ Daten abholen *)
      UNTIL stat <> REQ_WRKN;  (* solange, bis Auftrag fertig
                                  mit oder ohne Fehler            *)

      CASE stat OF
        REQ_NO_ERR:WriteLn('Datum: ', wert, ' wurde gelesen');
        REQ_UNDEF: WriteLn('Auftrags-Status ist undefiniert, Datum:
                           ',wert,' gelesen');
        ELSE WriteLn('Auftrag ist fertig mit Fehler: ', stat)
      END;
   END;
END.
```

2.3.3.3 Lesen eines Blockes aus dem AG

```
FUNCTION CP_readn_AG (size, typ, bst:BYTE; adr : longint;
                      len :  WORD) : integer;
```

Parameter

size	Datentyp der Blockelemente (siehe Kapitel 2.3.3.11.4)
typ	Elementtyp der Blockelemente (siehe Kapitel 2.3.3.11.3)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Datenanzahl in Worten oder Byte je nach Elementtyp

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion Lesen eines Blockes aus dem AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.3.11 beschrieben. Erkennt der Treiber einer Fehler, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden, andernfalls kann z.B. die Kachel blockiert werden:

```
VAR   a_nr,                     (* Auftragsnummer für Lese-Auftrag *)
      stat : INTEGER;           (* momentaner Auftrags-Status
*)
      buff : ARRAY[1..100 ] OF INTEGER(* aus dem AG gelesene Daten*)
BEGIN
 (* Auftrag starten *)
   a_nr := CP_readn_AG(W_BLOCK, DB_BLK, 5, 10, 100);

   IF a_nr < 0                   (* Fehler ist aufgetreten        *)
   THEN WriteLn('Auftrag wurde beendet mit Fehler: ', a_nr)

   ELSE  BEGIN                   (* a_nr enthält die Auftragsnummer *)
      REPEAT
            stat := CP_stat_AG(a_nr, Addr(buf)); (* Auftrags-
                                                 status/Daten abholen *)
      UNTIL stat <> REQ_WRKN;  (* solange, bis Auftrag fertig mit
                                  oder ohne Fehler                *)
      CASE stat  OF
          REQ_NO_ERR:       WriteLn('Puffer wurde gelesen');
          REQ_UNDEF:        WriteLn('Auftrags-Status ist
                                undefiniert, Puffer wurde gelesen');
          ELSE              WriteLn('Auftrag ist fertig mit Fehler:
                                  ', stat);
      END;
   END;
END.
```

2.3.3.4 Schreiben eines Einzelelementes in das AG

```
FUNCTION CP_write_AG (size, typ, bst : BYTE; adr: longint;
                      bit: BYTE; p: POINTER): integer;
```

Parameter

size	Elementgröße (siehe Kapitel 2.3.3.11.1)	
typ	Elementtyp Einzelelemente (siehe Kapitel 2.3.3.11.2)	
bst	Bausteinnummer	
adr	Adresse im Baustein oder absolute Adresse	
bit	Bit-Nummer	
p	Zeiger auf zu schreibendes Datum	
	bei Elementgröße Bit oder Byte:	Zeiger auf ein Byte
	bei Elementgröße Wort:	Zeiger auf ein Wort
	bei Elementgröße Doppelwort:	Zeiger auf ein Doppelwort

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Schreiben eines Einzelelementes in das AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.3.11 beschrieben. Wird vom Treiber bei der Ausführung ein Fehler erkannt, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Vorgehensweise beim Aufruf

```
VAR   a_nr,                     (* Auftragsnummer für Lese-Auftrag *)
      stat : INTEGER;           (* momentaner Auftrags-Status
*)
      wert : BYTE;              (* zu schreibender Wert           *)
BEGIN
 ...
(* Auftrag starten *)
   wert := $7E;
   a_nr := CP_write_AG(LBYTE_ELM, DB_SNG, 10, 1, 0, Addr(wert));

   IF a_nr < 0                  (* Fehler ist aufgetreten         *)
   THEN WriteLn('Auftrag wurde beendet mit Fehler: ', a_nr)

   ELSE  BEGIN                  (* a_nr enthält die Auftragsnummer *)
      REPEAT
            stat := CP_stat_AG(a_nr, NIL);     (* Auftrags-Status
                                                  lesen          *)
      UNTIL stat <> REQ_WRKN;  (* solange bis, Auftrag fertig mit
                                  oder ohne Fehler               *)

      CASE stat OF
          REQ_NO_ERR: WriteLn('Datum: ', wert,' wurde geschrieben');
          REQ_UNDEF:  WriteLn('Auftrags-Status ist undefiniert.');
          ELSE        WriteLn('Auftrag ist fertig mit Fehler: ',
                              stat)
      END;
   END;
End.
```

2.3.3.5 Schreiben eines Blockes in das AG

```
FUNCTION CP_writen_AG (size, typ, bst : BYTE; adr : longint;
                       len : word; p : POINTER) : integer;
```

Parameter

size	Datentyp der Blockelemente (siehe Kapitel 2.3.3.11.4)
typ	Elementtyp Blockelemente (siehe Kapitel 2.3.3.11.3)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Datenanzahl in Worten oder Byte je nach Elementtyp
p	Zeiger auf zu schreibenden Datenblock

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Schreiben eines Blockes in das AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.3.11 beschrieben.

Erkennt der Treiber bei der Ausführung einen Fehler, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Vorgehensweise beim Aufruf

```
VAR   a_nr,                     (* Auftragsnummer für Lese-Auftrag *)
      stat : INTEGER;           (* momentaner Auftrags-Status      *)
      i : INTEGER;
      buff : ARRAY[1..100 ] OF INTEGER;(* zu schreibende Daten    *)
BEGIN
 ...
   FOR i := 1 TO 100 DO
            buff[i] := i;       (* Datenpuffer vorbesetzen         *)
(* Auftrag starten *)
   a_nr := CP_writen_AG(B_BLOCK, DB_BLK, 5, 10, 100, Addr(buff));

   IF a_nr < 0                 (* Fehler ist aufgetreten          *)
   THEN WriteLn('Auftrag wurde beendet mit Fehler: ', a_nr)

   ELSE  BEGIN                 (* a_nr enthält die Auftragsnummer  *)
      REPEAT
            stat := CP_stat_AG(a_nr, NIL);      (* Auftrags-Status
                                                   lesen       *)
      UNTIL stat <> REQ_WRKN;       (* solange bis, Auftrag fertig
                                       mit oder ohne Fehler       *)

      CASE stat OF
          REQ_NO_ERR: WriteLn('Puffer wurde geschrieben');
          REQ_UNDEF: WriteLn('Auftrags-Status ist undefiniert.');
          ELSE      WriteLn('Auftrag ist fertig mit Fehler:', stat)
      END;
   END;
END.
```

2.3.3.6 Status eines Auftrags lesen

```
FUNCTION CP_stat_AG (a_nr : INTEGER; p: POINTER): INTEGER;
```

Parameter

a_nr	Auftragsnummer des zu testenden Auftrags	
p	Zeiger auf Datum oder Datenblock im PC-Speicher (nur bei Lese-Aufträgen mit Einzel- und Blockelement zu besetzen)	
	bei Elementgröße Bit oder Byte	Zeiger auf ein Byte
	bei Elementgröße Wort	Zeiger auf ein Wort
	bei Elementgröße Doppelwort	Zeiger auf ein Doppelwort
	bei Elementgröße Block	Zeiger auf Puffer für Datenblock

Rückgabe

Auftrags-Status oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Status-Abfrage zu Auftrag" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.3.11 beschrieben. Tritt bei der Ausführung des Auftrags ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird der Auftrags-Status (siehe Kapitel 2.3.3.11.4) zurückgegeben.

2.3.3.7 Alle Aufträge einer Kachel abbrechen

```
FUNCTION CP_cncl_AG (a_nr : BYTE): INTEGER;
```

Parameter

a_nr	Kennung für Kachel 2 $28 alle noch aktiven Aufträge der Kachel 2 abbrechen

Rückgabe

0 oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Abbruch aller Aufträge einer Kachel" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.3.11 beschrieben. Tritt bei der Ausführung des Auftrags ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, d.h. alle Aufträge wurden abgebrochen, so wird 0 zurückgegeben.

2.3.3.8 Prozeßabbild-Status lesen

```
FUNCTION CP_stat_PA : BYTE;
```

Rückgabe

Prozeßabbild-Zähler

Diese Funktion ruft die Treiber-Funktion "Status-Abfrage Prozeßabbild" auf. Die Funktion gibt den Prozeßabbild-Zähler zurück.

2.3.3.9 Bereich des Prozeßabbilds lesen

```
FUNCTION CP_read_PA(typ : BYTE; adr, len : WORD; p : POINTER) :
                    INTEGER;
```

Parameter

typ	Elementtyp Prozeßabbild (siehe Kapitel 2.3.3.11.6)
adr	Adresse im Bereich oder absolute Adresse
len	Anzahl der Daten (Bytes oder Worte) je nach Typ
p	Zeiger auf Datenpuffer im Speicher

Rückgabe

Prozeßabbild-Zähler

Diese Funktion ruft die Treiber-Funktion "Lesen eines Bereiches des Prozeßabbildes" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.3.11 beschrieben. Tritt bei der Ausführung des Auftrags ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird der aktuelle Wert des Prozeßabbild-Zählers zurückgegeben.

2.3.3.10 Standardfunktion

```
FUNCTION CP_CALL (VAR setwert : CPX86_PARAMETER_REC) : INTEGER;
```

Datenstruktur

```
TYPE CPX86_PARAMETER_REC =
     RECORD
       Elementtyp       : byte;     (* Elementtyp siehe Kapitel 2.3.3.11.2 bzw.
                                       2.3.3.11.3 *)
       Auftrag          : byte;     (* Auftragsart siehe Kapitel 2.3.3.11.7*)
       Bausteinnummer   : byte;     (* Nummer des Datenbaustein oder 0 bei
                                       Merkern, Ausgängen usw. *)
       Kennung          : byte;     (* Elementgroesse oder Datentyp siehe
                                       Kapitel 2.3.3.11.1 bzw. 2.3.3.11.4 *)
       Adresse          : integer;  (* Nummer des ersten Datenelements *)
       Len              : integer;  (* Laenge der Daten *)
       ptr              : pointer;  (* Pointer auf DOS-Daten immer "nil" *)
       ptr_win          : pointer;  (* Pointer auf WIN-Daten immer "nil" *)
       upro_zeiger      : longint;  (* Zeiger auf Unterprogramm unbenutzt *)
       CPU              : byte;     (* Nummer der CPU 0-3, bei einer CPU immer 0*)
       Reserved         : byte;     (* Reservebyte*)
       Fehler           : integer;  (* Rueckgabebyte fuer Fehler*)
       case Integer of
         0:      (DatenByte   : ARRAY[1..1000] of byte);
                   (* Datenuebergabe von oder zur Prozedur als Byte (8 Bit) *)
         1:      (DatenWort   : ARRAY[1..500] of integer);
                   (* Datenuebergabe von oder zur Prozedur als Integer (16 Bit)*)
         2:      (DatenDoppel : ARRAY[1..250] of longint);
                   (* Datenuebergabe von oder zur Prozedur als Longint (32 Bit)*)
     END;
```

Datenstruktur für die Standardfunktion. Die Komponenten werden entsprechend der auszuführenden Funktion vorbesetzt.

Durch diese Funktion können alle vorher beschriebenen Funktionen ersetzt werden.

Beim Ersetzen der Funktionen write_AG bzw. writen_AG werden die zu schreibenden Daten in das entsprechende ARRAY eingetragen und dann die Funktion CP_CALL aufgerufen.

Beim Ersetzen der Funktionen read_AG bzw. readn_AG werden die Daten nach Aufruf der Funktion CP_CALL (als Funktion CP_stat_AG) im entsprechenden ARRAY bereitgestellt.

Vorgehensweise beim Aufruf

```
(* Beispiel in Turbo-Pascal zur Kommunikation ueber Kachel 2 mit 2 CPUs *)

        program Test;

        USES DOS, crt, CP486LIB;

        VAR
            ret1,ret2     : integer;
            a_nr          : BYTE;
            i,zaehl       : WORD;
            setwert       : cpx86_parameter_rec;

        begin

         clrscr;
         writeln;
         writeln('  Fuer diese DEMO wird auf der SPS-Seite der FB1
                    benoetigt,');
         writeln('  der zyklisch vom OB1 aufgerufen wird');
         delay(2000);

        repeat
        begin
                                    (* Struktur fuer Auftrag vorbelegen *)
            setwert.elementtyp := DB_SNG;(* Elementtyp laut Tabelle       *)
            setwert.auftrag := READ_AG;     (* Auftragsart                *)
            setwert.bausteinnummer := 10;   (* Nummer des Datenbausteins  *)
            setwert.kennung := W_BLOCK;     (* Elemetgroesse laut Tabelle *)
            setwert.adresse := 0;           (* Startadreese im AG         *)
            setwert.len     := 50;          (* Uebertragungslaenge        *)
            setwert.cpu     := 0;           (* CPU-Nummer 0 fuer erste CPU*)

            ret1 := CP_CALL(setwert);       (* Daten an Treiber uebergeben*)

                    (* Struktur fuer Auftrag vorbelegen*)
            setwert.elementtyp := DB_SNG;   (* Elementtyp laut Tabelle    *)
            setwert.auftrag := READ_AG;     (* Auftragsart                *)
            setwert.bausteinnummer := 10;   (* Nummer des Datenbausteins  *)
            setwert.kennung := W_BLOCK;  (* Elementgroesse laut Tabelle *)
            setwert.adresse := 0;                (* Startadreese im AG   *)
            setwert.len     := 50;               (* Uebertragungslaenge  *)
            setwert.cpu     := 1;        (* CPU-Nummer 1 fuer zweite CPU  *)

            ret2 := CP_CALL(setwert);    (* Daten an Treiber uebergeben   *)

           IF ret1 < 0                       (* Fehler ist aufgetreten     *)
            THEN write(' Auftrag wurde beendet mit Fehler: ',ret1,'   ')
            ELSE
             begin
             a_nr := ret1;              (* Auftragsnummer uebergeben      *)
             repeat
               setwert.elementtyp := a_nr;  (* Auftragsnummer an Struktur *)
               setwert.auftrag := statr_AG; (* Auftragsart                *)
               setwert.cpu     := 0;             (* CPU-Nummer eintragen *)
               setwert.ptr     := nil;           (* Zeiger löschen       *)

               ret1 := cp_call(setwert);    (* Daten an Treiber übergeben *)
             until (ret1 <> 1);

             if ret1 = 3 then    (* Ausführung der Funktion war fehlerfrei*)
               begin
                gotoxy(1,6);             (* Courser positionieren         *)
                write('Daten :   ');
                for i := 1 to 50 do
```

```
                    begin
                     write(setwert.datenwort[i]:5,'   '); (* Daten am
                                                          Bildschirm ausgeben*)
                   end;
                 end;
                end;


   IF ret2< 0                              (* Fehler ist aufgetreten          *)
    THEN write(' Auftrag wurde beendet mit Fehler: ',ret2,'   ')
    ELSE
     begin
     a_nr := ret2;                         (* Auftragsnummer uebergeben       *)
     repeat
       setwert.elementtyp := a_nr;         (* Auftragsnummer an Struktur      *)
       setwert.auftrag := statr_AG;        (* Auftragsart                     *)
       setwert.cpu     := 1;               (* CPU-Nummer eintragen            *)
       setwert.ptr     := nil;             (* Zeiger löschen                  *)

       ret2 := cp_call(setwert);           (* Daten an Treiber übergeben      *)
     until (ret1 <> 1);

     if ret2 = 3 then               (* Ausführung der Funktion war fehlerfrei*)
       begin
        gotoxy(1,6);                (* Courser positionieren                 *)
        write('Daten :   ');
        for i := 1 to 50 do
         begin
          write(setwert.datenwort[i]:5,'   ');(* Daten am Bildschirm ausgeben *)
        end;
     end;
    end;


end;

until keypressed;

end.
```

2.3.3.11 Konstanten

Die nachfolgend genannten Konstanten sind bereits vordefiniert. Aufgrund der besseren Lesbarkeit und Übersichtlichkeit wird empfohlen, diese Konstanten auch im Programmtext zu verwenden. Eine Anpassung an spätere Änderungen des COM-Treibers kann so leichter vorgenommen werden.

2.3.3.11.1 Konstanten für Elementgrößen

```
CONST
      BIT_ELM     = $00;      (* Bit                          *)
      SEMA_ELM    = $01;      (* Bit als Semaphor             *)
      BYTE_ELM    = $02;      (* Byte                         *)
      LBYTE_ELM   = $02;      (* linkes Byte eines Wortes     *)
      RBYTE_ELM   = $03;      (* rechtes Byte eines Wortes    *)
      WORD_ELM    = $04;      (* Wort                         *)
      DWORD_ELM   = $05;      (* Doppelwort                   *)
      BLOCK_ELM   = $07;      (* Block                        *)
```

2.3.3.11.2 Konstanten für Elementtypen bei Einzelelementen

```
CONST
      DB_SNG      = $00;      (* DB                           *)
      DX_SNG      = $01;      (* DB im Externspeicher         *)
      BA_SNG      = $02;      (* BA                           *)
      BB_SNG      = $03;      (* BB                           *)
      BS_SNG      = $04;      (* BS                           *)
      BT_SNG      = $05;      (* BT                           *)
      Z_SNG       = $06;      (* Zähler                       *)
      T_SNG       = $07;      (* Timer                        *)
      MB_SNG      = $08;      (* Merker                       *)
      EB_SNG      = $09;      (* Eingangsbereich              *)
      AB_SNG      = $0A;      (* Ausgangsbereich              *)
      PY_SNG      = $0B;      (* P-Peripherie                 *)
      QY_SNG      = $0C;      (* Q-Peripherie                 *)
      ABS_SNG     = $0F;      (* Absoluter Speicher           *)
      FB_SNG      = $10;      (* FB                           *)
      FX_SNG      = $11;      (* FB im Externspeicher         *)
      OB_SNG      = $12;      (* OB                           *)
      PB_SNG      = $13;      (* PB                           *)
      SB_SNG      = $14;      (* SB                           *)
      RB_FREE     = $0E;      (* Code für Auftragsblock frei *)
```

2.3.3.11.3 Konstanten für Elementtypen bei Blockelementen

```
CONST
      DB_BLK      = $00;          (* DB                    *)
      DX_BLK      = $01;          (* DB im Externspeicher  *)
      BA_BLK      = $02;          (* BA                    *)
      BB_BLK      = $03;          (* BB                    *)
      BS_BLK      = $04;          (* BS                    *)
      BT_BLK      = $05;          (* BT                    *)
      Z_BLK       = $06;          (* Zähler                *)
      T_BLK       = $07;          (* Timer                 *)
      MB_BLK      = $08;          (* Merker                *)
      EB_BLK      = $09;          (* Eingangsbereich       *)
      AB_BLK      = $0A;          (* Ausgangsbereich       *)
      PY_BLK      = $0B;          (* P-Peripherie          *)
      QY_BLK      = $0C;          (* Q-Peripherie          *)
      ABS_BLK     = $0F;          (* Absoluter Speicher    *)
      FB_BLK      = $10;          (* FB                    *)
      FX_BLK      = $11;          (* FB im Externspeicher  *)
      OB_BLK      = $12;          (* OB                    *)
      PB_BLK      = $13;          (* PB                    *)
      SB_BLK      = $14;          (* SB                    *)
```

2.3.3.11.4 Konstanten für den Datentyp bei Blockelementen

```
CONST
      B_BLOCK     = $07;      (* Typ: Block von Bytes       *)
      W_BLOCK     = $17;      (* Typ: Block von Worten      *)
      D_BLOCK     = $27;      (* Typ: Block von Langworten  *)
```

2.3.3.11.5 Kennung für Auftrags-Status

```
CONST
      REQ_WRKN    = $01;      (* Auftrag in Bearbeitung     *)
      REQ_UNDEF   = $02;      (* Auftrags-Status undefiniert*)
      REQ_NO_ERR  = $03;      (* Auftrag fertig ohne Fehler *)
```

2.3.3.11.6 Konstanten für Elementtypen bei Prozeßabbild

```
CONST
      Z_PA        = $06;      (* Zähler                     *)
      T_PA        = $07;      (* Timer                      *)
      MB_PA       = $08;      (* Merker                     *)
      EB_PA       = $09;      (* Eingangsbereich            *)
      AB_PA       = $0A;      (* Ausgangsbereich            *)
      ABS_PA      = $0F;      (* absoluter Block im PA      *)
```

2.3.3.11.7 Auftragsart für CP__Call

```
CONST
      GET_INFO          = $00
      STATR_AG          = $20
      READ_AG           = $21
      WRITE_AG          = $21
      CNCLR_AG          = $28
      STAT_PA           = $70
      READ_PA           = $71
      HANDLE_Struct     = $7F       (* dx:bx = Zeiger auf CP486_
                                       Parameter_Rec *)
```

2.3.3.11.8 Konstanten für Fehlermeldungen: Kachel 2 und 7

```
CONST
```

`ERR_S5_TYP = $01;` (* ungültiger Elementtyp *)

Bei Einzelelementzugriff mit Elementtyp DX_SNG, BA_SNG, BB_SNG, BT_SNG oder QB_SNG bzw. bei Blockelementzugriff mit Elementtyp DX_BLK, BA_BLK, BB_BLK, BT_BLK oder FX_BLK wurde versucht auf Daten in einem AG der 115U-Reihe zuzugreifen. Diese Elementtypen existieren in diesem AG-Typ jedoch nicht.

Abhilfe: Parameter "`typ`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`ERR_S5_BST = $02;` (* Baustein nicht vorhanden *)

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementtyp DB_BLK wurde versucht auf einen nicht existierenden Baustein zuzugreifen.

Abhilfe: Datenbaustein im AG anlegen oder Parameter "`bst`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`ERR_S5_ELM = $03;` (* Element nicht vorhanden *)

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementzugriff mit Elementtyp DB_BLK wurde versucht auf Daten in einem DB zuzugreifen, die nicht vorhanden sind.

Abhilfe: Datenbaustein im AG entsprechend verlängern oder Parameter "`adr`" bzw. "`len`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten bzw. Zählerstände mit einer Nummer > 127 zuzugreifen.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG wurde versucht auf Merker mit Nummer > 199 bei Elementgröße Byte, mit Nummer > 198 bei Elementgröße Wort oder mit Nummer > 196 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware auf Wertigkeit überprüfen.

Bei Einzelelementzugriff mit Elementtyp EB_ - oder AB_SNG wurde versucht auf das Prozeßabbild des E/A-Bereichs mit Nummer > 127 bei Elementgröße Byte, mit Nummer > 126 bei Elementgröße Wort oder mit Nummer > 124 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware auf Wertigkeit überprüfen.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht, auf Elemente der P-Peripherie mit Nummer > 255 bei Elementgröße Byte, mit Nummer > 254 bei Elementgröße Wort oder mit Nummer > 252 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware auf Wertigkeit überprüfen.

`ERR_S5_SIZE = $04;` (* ungültige Elementgröße *)

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht, auf Zeiten oder Zählerstände zuzugreifen, wobei der Parameter Elementgröße nicht auf Wortzugriff (WORD_ELM) gesetzt war.

Abhilfe: Parameter "`size`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG wurde versucht, mit dem Parameter Elementgröße RBYTE_ELM auf Merker oder absolute Adressen zuzugreifen.

Abhilfe: Parameter "`size`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht, mit dem Parameter Elementgröße SEMA_ELM oder RBYTE_ELM auf die Ein- bzw. Ausgängen im Prozeßabbild zuzugreifen.

Abhilfe: Parameter "`typ`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht, mit dem Parameter Elementgröße BIT_ELM, SEMA_ELM oder RBYTE_ELM auf die P-Peripherie zuzugreifen.

Abhilfe: Parameter "`typ`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei lesendem Einzelelementzugriff mit Elementtyp ABS_SNG wurde versucht, von absoluten Adressen mit Elementgröße SEMA_ELM zu lesen. Diese Zugriffsart ist unter absoluter Adressierung nur schreibend möglich! Wenn einzelne Bits gelesen werden sollen, ist die Elementgröße BIT_ELM zu verwenden.

Abhilfe: Parameter "`typ`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`ERR_S5_BIT = $05;` (* Bit-Nummer zu groß *)

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG und der Elementgröße BIT_ELM oder SEMA_ELM wurde versucht, auf ein Merker- oder Absolutadreß-Bit mit einer Bitnummer > 7 (15) zuzugreifen.

Abhilfe: Parameter "`bit`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht auf ein E/A-BIT mit einer Bitnummer > 7 zuzugreifen.

Abhilfe: Parameter "`bit`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`ERR_S5_STRT = $06;` (* ungültige Anfangsadresse *)

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde ein Blocktransfer über Bausteine versucht, wobei die relative Anfangsadresse im Block > 32767 ist.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`ERR_S5_LEN = $07;` (* ungültige Blocklänge *)

Bei Blockelementzugriff unter allen Elementtypen wurde ein Blocktransfer mit einer Länge > 504 Worten versucht.

Abhilfe: Parameter "`len`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

```
ERR_S5_ADR      = $08;
```
(* Adresse zu groß *)

Bei einem Einzel- oder Blockelementzugriff mit Elementtyp ABS_SNG wurde versucht auf eine Adresse > FFFFh in einem AG der 115U-Reihe zuzugreifen. Die CPUs (bis CPU 944) haben jedoch nur 64 KB Adreßraum.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

```
ERR_S5_QVZ      = $09;
```
(* QVZ/ADF im AG bei lesen/schreiben *)

Es wurde versucht auf einem Adreßbereich zuzugreifen, der physikalisch nicht vorhanden ist.
Die AGs der 135 und 155-Reihe stellen diese Fehlermeldungen zu Verfügung. Ein AG der 115U-Reihe würde hierbei in den Stop-Zustand versetzt werden.

Abhilfe: Parameter "`typ`" bzw. "`adr`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

```
ERR_S5_944      = $0A;
```
(* CPU 944: Baustein im Prog.Bank *)
Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde versucht, auf einen Baustein zuzugreifen, der nicht in der DB-Bank liegt.
(Betrifft nur CPU 944 vom AG-Typ 115U)

Abhilfe: Baustein im AG in DB-Bank anlegen (über BIB-Nr. 19285) oder Funktionsaufruf in der PC-Anwendersoftware korrigieren.

2.3.4 Schnittstelle zu Turbo-C (2.0 und C++ ab 1.0), Microsoft-C 6.0

Für Funktionsaufrufe des COM-Treibers einfach aus C-Programmen heraus, wurde ein Bibliotheks-File erstellt, das alle Funktionen des Service-Interrupts INT zur Verfügung stellt. Zu jeder Funktion des Treibers ist eine entsprechende C-Funktion definiert, die die Versorgung der Register, den Aufruf des Interrupts und die Rückgabe der Werte durchführt. So können auch Anwender, die nicht mit systemnaher Programmierung auf CP vertraut sind, alle Möglichkeiten des Treibers nutzen.

Im Include-File "CP486DEF.H" sind Datentypen und Konstanten für Elementgröße, Elementtypen und Fehlernummern definiert, ebenso sind die Funktionsprototypen der nachfolgend beschriebenen Funktionen im ANSI-C Stil definiert. Das Include-File muß im Anwenderprogramm genannt werden.

Alle erforderlichen Funktionen sind im File CP486LIB.C implementiert. Bei Verwendung in einem Programm muß die Datei CP486LIB.OBJ mit eingebunden werden. Je nach Programmierumgebung und Version ist die Datei in die Project-Datei (Turbo-C), Dependencie List (Microsoft-C) oder im Make-File aufzunehmen. Details hierzu sind den jeweiligen Handbüchern zu entnehmen.

Hinweis

In "CP486LIB.H" ist byte als unsigned char definiert, word als unsigned short.

2.3.4.1 Funktion CP-Status-Abfrage

Datenstrukturen

```
int CP_info (p)
```

```
typedef struct
{
  word CP_id;                  /* Kennung: CP486 Wert = $C386  */
  byte VGA_ver, BIOS_ver;      /*  */
  byte DRV_ver;                /*  */
  byte CPU_AG;                 /*  */
  byte CP_reg, S5_reg;         /*  */
} CP486_InfoBlk;               /*  */
```

Datenstruktur für die allgemeine Status-Info Funktion, bei der die Komponenten entsprechend der Werte der CP486 besetzt werden.

Diese Funktion ruft die Treiber-Funktion "allgemeine Status-Information" auf. Dieser Funktion geht eine Installationsprüfung des Treibers voraus. Ist der Treiber nicht installiert, gibt die Funktion den Wert -1 zurück. Bei installiertem COM-Treiber, wird der Wert 0 zurückgegeben. Die Komponente CP_id enthält als Kennung immer den Wert $C386, die anderen Komponenten werden nicht aktualisiert.

2.3.4.2 Lesen eines Einzelelements aus dem AG

```
int CP_read_AG (byte size, byte typ, byte bst, unsigned long adr,
                byte bit);
```

Parameter

size	Elementgröße (siehe Kapitel 2.3.4.11.1)
typ	Elementtyp Einzelelemente (siehe Kapitel 2.3.4.11.2)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
bit	Bit-Nummer

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Lesen eines Einzelelementes aus dem AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.4.11 beschrieben.

Erkennt der Treiber bei der Ausführung einen Fehler, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden, andernfalls kann z.B. die Kachel blockiert werden (siehe Kapitel 2.3.4.11):

```
int a_nr;           /* Auftragsnummer für Lese-Auftrag */
int stat;               /* momentaner Auftrags-Status */
int time_count=4000;/* Zeitzähler für Timeout (= 4 Sekunden)*/
byte wert ;         /* aus dem AG gelesener Wert */

/* Auftrag starten */
a_nr = CP_read_AG(LBYTE_ELM, DB_SNG, 10, 1, 0);

if(a_nr < 0)        /* Fehler ist aufgetreten */
  printf("Auftrag wurde beendet mit Fehler: %d\n", a_nr);

else
{                         /* a_nr enthält die Auftragsnummer */
  do
  {
    delay(1);                           /* 1 ms warten */
    stat = CP_stat_AG(a_nr, &wert);     /* Auftrags-Status/Daten abholen */
    /* solange bis Zeit abgelaufen bzw. Auftrag fertig mit oder ohne Fehler */
  } while((time_count-- > 0) && (stat == REQ_WRKN));

  switch(stat)
  {
    case REQ_NO_ERR:
      printf("Datum: %d wurde gelesen\n", wert);
    break;
    case REQ_UNDEF:
      printf("Auftrags-Status undefiniert, Datum: %d gelesen\n", wert);
    break;
    default:
      printf("Auftrag ist fertig mit Fehler: %d\n", stat);
  }
}
```

2.3.4.3 Lesen eines Blockes aus dem AG

```
int CP_readn_AG (byte size, byte typ, byte bst, unsigned long adr,
                 word len);
```

Parameter

size	Datentyp der Blockelemente (siehe Kapitel 2.3.4.11.4)
typ	Elementtyp Blockelemente (siehe Kapitel 2.3.4.11.3)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Datenanzahl in Worten oder Byte je nach Elementtyp

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler

Diese Funktion ruft die Treiber-Funktion "Lesen eines Blockes aus dem AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.4.11 beschrieben. Erkennt der Treiber bei der Ausführung ein Fehler, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der TreiberFunktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden, andernfalls kann z.B. die Kachel blockiert werden:

```
int a_nr;                  /* Auftragsnummer für Lese-Auftrag */
int stat;                  /* momentaner Auftrags-Status */
int time_count=4000;/* Zeitzähler für Timeout (= 4 Sekunden) */
int buff[100];                    /* aus dem AG gelesener Wert */

/* Auftrag starten */
a_nr = CP_readn_AG(W_BLOCK, DB_BLK, 5, 10, 100);

if(a_nr < 0)        /* Fehler ist aufgetreten */
  printf("Auftrag wurde beendet mit Fehler: %d\n", a_nr);

else
{                                 /* a_nr enthält die Auftragsnummer */
  do
  {
    delay(1);                            /* 1 ms warten */
    stat = CP_stat_AG(a_nr, &buff);      /* Auftrags-Status/Daten abholen */
    /* solange bis Zeit abgelaufen bzw. Auftrag fertig mit oder ohne Fehler */
  } while((time_count-- > 0) && (stat == REQ_WRKN));

  switch(stat)
  {
    case REQ_NO_ERR:
      printf("Daten wurden gelesen\n");
    break;
    case REQ_UNDEF:
      printf("Auftrags-Status undefiniert\n");
    break;
    default:
      printf("Auftrag ist fertig mit Fehler: %d\n", stat);
  }
}
```

2.3.4.4 Schreiben eines Einzelelementes in das AG

```
int CP_write_AG (byte size, byte type, byte bst, unsigned long adr,
                 byte bit, void far *p);
```

Parameter

size	Elementgröße (siehe Kapitel 2.3.4.11.1)	
typ	Elementtyp Einzelelemente (siehe Kapitel 2.3.4.11.2)	
bst	Bausteinnummer	
adr	Adresse im Baustein oder absolute Adresse	
bit	Bit-Nummer	
p	Zeiger auf zu schreibendes Datum im PC-Speicher	
	bei Elementgröße Bit oder Byte	Zeiger auf ein Byte
	bei Elementgröße Wort	Zeiger auf ein Wort
	bei Elementgröße Doppelwort	Zeiger auf ein Doppelwort

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Schreiben eines Einzelelementes in das AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.4.11 beschrieben. Erkennt der Treiber bei der Ausführung einen Fehler, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden, andernfalls kann z.B. die Kachel blockiert werden (siehe Kapitel 2.3.4.11):

```
int a_nr;              /* Auftragsnummer für Lese-Auftrag */
int stat;              /* momentaner Auftrags-Status */
int time_count=4000;          /* Zeitzähler für Timeout (= 4 Sekunden) */
byte wert = 0x5A;             /* aus dem AG gelesener Wert */
/* Auftrag starten */
a_nr = CP_write_AG(LBYTE_ELM, DB_SNG, 10, 1, 0, &wert);
if(a_nr < 0)                          /* Fehler ist aufgetreten */
  printf("Auftrag wurde beendet mit Fehler: %d\n", a_nr);
else
{                                      /* a_nr enthält die Auftragsnummer */
  do
  {
    delay(1);                         /* 1 ms warten *
    stat = CP_stat_AG(a_nr, NULL);    /* Auftrags-Status lesen */
    /* solange bis Zeit abgelaufen bzw. Auftrag fertig mit oder ohne
       Fehler */
  } while((time_count-- > 0) && (stat == REQ_WRKN));
  switch(stat)
    {
    case REQ_NO_ERR:
      printf("Datum: %x wurde geschrieben\n",       wert);
    break;
    case REQ_UNDEF:
      printf("Auftrags-Status undefiniert\n");
    break;
    default:
      printf("Auftrag ist fertig mit Fehler: %d\n", stat);
    }
}
```

2.3.4.5 Schreiben eines Blockes in das AG

```
int CP_writen_AG (byte size, byte typ, byte bst, unsigned long adr,
                  word len, void far *p);
```

Parameter

size	Datentyp der Blockelemente (siehe Kapitel 2.3.4.11.4)
typ	Elementtyp Blockelemente (siehe 2.3.4.11.3)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Datenanzahl in Worten oder Byte je nach Elementtyp
p	Zeiger auf zu schreibenden Datenblock im PC-Speicher

Rückgabe

Auftragsnummer oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Schreiben eines Blockes in das AG" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.4.11 beschrieben.

Erkennt der Treiber bei der Ausführung einen Fehler, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird die Auftragsnummer als Funktionswert zurückgegeben.

Vorgehensweise beim Aufruf

Zur korrekten Bearbeitung der Treiber-Funktionen sollte folgendes Schema bei der Ausführung von Funktionen eingehalten werden, andernfalls kann z.B. die Kachel blockiert werden:

```
int a_nr;            /* Auftragsnummer für Lese-Auftrag  */
int stat;            /* momentaner Auftrags-Status        */
int i;
int time_count=4000;        /* Zeitzähler für Timeout (= 4 Sekunden) */
int buff[100];              /* zu schreibende Daten */
for(i = 0; i< 100; i++)
buff[i] = (byte)i;          /* Datenpuffer vorbesetzen */

/* Auftrag starten */
a_nr = CP_writen_AG(B_BLOCK, DB_BLK, 5, 10, 100, &buff);

if(a_nr < 0)         /* Fehler ist aufgetreten */
  printf("Auftrag wurde beendet mit Fehler: %d\n", a_nr);
else
{                                /* a_nr enthält die Auftragsnummer */
  do
  {
    delay(1);                    /* 1 ms warten */
    stat = CP_stat_AG(a_nr, NULL);        /* Auftrags-Status lesen */
    /* solange, bis Auftrag fertig mit oder ohne Fehler */
  } while((time_count-- > 0) && (stat == REQ_WRKN));

  switch(stat)  {
  case REQ_NO_ERR:
    printf("Daten wurden geschrieben\n");
  break;
  case REQ_UNDEF:
    printf("Auftrags-Status undefiniert\n");
  break;
  default:
    printf("Auftrag ist fertig mit Fehler: %d\n", stat);
  }
}
```

2.3.4.6 Status eines Auftrags lesen

```
int CP_stat_AG (int r, void far *p);
```

Parameter

a_nr	Auftragsnummer des zu testenden Auftrags
p	Zeiger auf Datum oder Datenblock im PC-Speicher nur bei Lese-Aufträgen mit Einzel- und Blockelement

bei Elementgröße Bit oder Byte	Zeiger auf ein Byte
bei Elementgröße Wort	Zeiger auf ein Wort
bei Elementgröße Doppelwort	Zeiger auf ein Doppelwort
bei Elementgröße Block	Zeiger auf Puffer für Datenblock

Rückgabe

Auftrags-Status oder negative Zahl bei Fehler.

Diese Funktion ruft die Treiber-Funktion "Status-Abfrage zu Auftrag" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.4.11 beschrieben. Tritt bei der Ausführung des Auftrags ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird der Auftrags-Status (siehe Kapitel 2.3.4.11.5) zurückgegeben.

2.3.4.7 Alle Aufträge einer Kachel abbrechen

```
int CP_cncl_AG (int a_nr);
```

Parameter

a_nr	Kennung für Kachel 2 $28 alle noch aktiven Aufträge der Kachel 2 abbrechen

Diese Funktion ruft die Treiber-Funktion "Abbruch aller Aufträge einer Kachel" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.4.11 beschrieben. Tritt bei der Ausführung des Auftrags ein Fehler auf, so wird die entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, d.h. alle Aufträge wurden abgebrochen, so wird 0 zurückgegeben.

2.3.4.8 Prozeßabbild-Status lesen

```
byte CP_stat_PA ();
```

Rückgabe

Prozeßabbild-Zähler

Diese Funktion ruft die Treiber-Funktion "Status-Abfrage Prozeßabbild" auf. Die Funktion gibt den Prozeßabbild-Zähler zurück.

2.3.4.9 Prozeßabbild-Bereich lesen

```
int CP_read_PA (byte typ, word adr, word len, void far *p);
```

Parameter

typ	Elementtyp Einzelelemente (siehe Kapitel 2.3.4.11.2)
bst	Bausteinnummer
adr	Adresse im Baustein oder absolute Adresse
len	Anzahl der Daten (Bytes oder Worte) je nach Typ
p	Zeiger auf Datenpuffer im CP-Speicher

Rückgabe

Prozeßabbild-Zähler.

Diese Funktion ruft die Treiber-Funktion "Lesen eines Bereiches des Prozeßabbilds" auf. Bei Aufruf werden die Register gemäß den übergebenen Parametern vorbesetzt. Die Bedeutung der Parameter ist in Kapitel 2.3.4.11 beschrieben. Bei Fehlerauftritt während der Ausführung des Auftrags wird eine entsprechende Fehlermeldung (negative Zahl) als Funktionswert zurückgegeben. Kann die Funktion fehlerfrei ausgeführt werden, wird der aktuelle Wert des Prozeßabbild-Zählers zurückgegeben.

2.3.4.10 Standardfunktion

```
unsigned far pascal CP_Call(CP486_PARAMETER far *cp);
```

```
typedef struct
{
unsigned char Elementtyp;     // Elementtyp siehe Kapitel 2.3.4.11.2 bzw.
                                 2.3.4.11.3.
unsigned char Auftrag;        // Auftragsart siehe Kapitel 2.3.4.11.7
unsigned char Bausteinnummer; // Nummer des Datenbaustein oder 0 bei Merkern,
                                 Ausgängen usw.
unsigned char Kennung;        // Elementgröße oder Datentyp siehe Kapitel
                                 2.3.4.11.1 bzw. 2.3.4.11.4.
unsigned short Adresse;       // Startadresse im Element
unsigned short Len;           // Anzahl Übergebener Elemente
unsigned char far *ptr;       // 16:16 Zeiger auf Daten DOS. Bei Verwendung der
                              // Daten in der Struktur, ptr auf 0 setzen.
unsigned char far *ptr_win;   // 16:16 Zeiger auf Daten Windows. Bei 32Bit Flat
                              // Model 32Bit Daten auf 0 setzen.
unsigned short (far pascal *ConfirmFunction)(unsigned char Auftragsnummer);
                              // Rückruffunktion wenn Auftrag fertig ist.
unsigned char Cpu;            // CPU Nummer in der SPS 0 - 3, bei einer CPU
immer 0
unsigned char reserved;       // Reserve
unsigned short Fehler;        // Fehlercode. 0 = kein Fehler
unsigned char Daten[1];       // Platz für die Daten. Wird verwendet wenn
                              // ptr==0 und ptr_win==0.
}CP486_PARAMETER;
```

Datenstruktur für die Standardfunktion, bei der die Komponenten entsprechend der auszuführenden Funktion vorbesetzt werden.

Der Zeiger ptr_win muß, wenn vorhanden, ein Segment 'Offset Zeiger' sein.

Das Element `Len` zeigt bei stat_AG die Länge des Datenbereichs an (Aufruf).

Beim Ende stat_AG ohne Fehler steht im Element `Len` die Anzahl gelesener Byte.

Durch diese Funktion können alle vorher beschrieben Funktionen ersetzt werden.

Beim Ersetzen der Funktionen write_AG bzw. writen_AG werden die zu schreibenden Daten in das entsprechende ARRAY eingetragen und dann die Funktion CP_Call aufgerufen.

Beim Ersetzen der Funktionen read_AG bzw. readn_AG werden die Daten nach Aufruf der Funktion CP_Call (als Funktion CP_stat_AG) im entsprechenden ARRAY bereitgestellt.

2.3.4.11 Konstanten

Die nachfolgend genannten Konstanten sind bereits vordefiniert. Aufgrund der besseren Lesbarkeit und Übersichtlichkeit wird empfohlen, diese Konstanten auch im Programmtext zu verwenden. Eine Anpassung an spätere Änderungen des COM-Treibers kann so leichter vorgenommen werden.

2.3.4.11.1 Konstanten für Elementgröße

```
#define BIT_ELM     0x00      /* Bit                          */
#define SEMA_ELM    0x01      /* Bit als Semaphor             */
#define BYTE_ELM    0x02      /* Byte                         */
#define LBYTE_ELM   0x02      /* linkes Byte eines Wortes     */
#define RBYTE_ELM   0x03      /* rechtes Byte eines Wortes    */
#define WORD_ELM    0x04      /* Wort                         */
#define DWORD_ELM   0x05      /* Doppelwort                   */
#define BLOCK_ELM   0x07      /* Block                        */
```

2.3.4.11.2 Konstanten für Elementtypen bei Einzelelementen

```
#define DB_BLK      0x00      /* DB                           */
#define DX_BLK      0x01      /* DB im Externspeicher         */
#define BA_BLK      0x02      /* BA                           */
#define BB_BLK      0x03      /* BB                           */
#define BS_BLK      0x04      /* BS                           */
#define BT_BLK      0x05      /* BT                           */
#define Z_BLK       0x06      /* Zähler                       */
#define T_BLK       0x07      /* Timer                        */
#define MB_BLK      0x08      /* Merker                       */
#define EB_BLK      0x09      /* Eingangsbereich              */
#define AB_BLK      0x0A      /* Ausgangsbereich              */
#define PY_BLK      0x0B      /* P-Peripherie                 */
#define QY_BLK      0x0C      /* Q-Peripherie                 */
#define ABS_BLK     0x0F      /* Absoluter Speicher           */
#define FB_BLK      0x10      /* FB                           */
#define FX_BLK      0x11      /* FB im Externspeicher         */
#define OB_BLK      0x12      /* OB                           */
#define PB_BLK      0x13      /* PB                           */
#define SB_BLK      0x14      /* SB                           */
#define RB_FREE     0x0E      /* Code für Auftragsblock frei*/
```

2.3.4.11.3 Konstanten für Elementtypen bei Blockelementen

```
#define DB_SNG      DB_BLK          /* DB                      */
#define DX_SNG      DX_BLK          /* DB im Externspeicher    */
#define BA_SNG      BA_BLK          /* BA                      */
#define BB_SNG      BB_BLK          /* BB                      */
#define BS_SNG      BS_BLK          /* BS                      */
#define BT_SNG      BT_BLK          /* BT                      */
#define Z_SNG       Z_BLK           /* Zähler                  */
#define T_SNG       T_BLK           /* Timer                   */
#define MB_SNG      MB_BLK          /* Merker                  */
#define EB_SNG      EB_BLK          /* Eingangsbereich         */
#define AB_SNG      AB_BLK          /* Ausgangsbereich         */
#define PY_SNG      PY_BLK          /* P-Peripherie            */
#define QY_SNG      QY_BLK          /* Q-Peripherie            */
#define ABS_SNG     ABS_BLK         /* Absoluter Speicher      */
#define FB_SNG      FB_BLK          /* FB                      */
#define FX_SNG      FX_BLK          /* FB im Externspeicher    */
#define OB_SNG      OB_BLK          /* OB                      */
#define PB_SNG      PB_BLK          /* PB                      */
#define SB_SNG      SB_BLK          /* SB                      */
```

2.3.4.11.4 Konstanten für den Datentyp bei Blockelementen

```
#define B_BLOCK     0x07      /* Typ: Block von Bytes       */
#define W_BLOCK     0x17      /* Typ: Block von Worten      */
#define D_BLOCK     0x27      /* Typ: Block von Langworten  */
```

2.3.4.11.5 Kennung für Auftrags-Status

```
#define REQ_WRKN    0x01      /* Auftrag in Bearbeitung     */
#define REQ_UNDEF   0x02      /* Auftrags-Status undefiniert*/
#define REQ_NO_ERR  0x03      /* Auftrag fertig ohne Fehler */
```

2.3.4.11.6 Konstanten für Elementtypen bei Prozeßabbild

```
#define Z_PA        0x06          /* Zähler                 */
#define T_PA        0x07          /* Timer                  */
#define MB_PA       0x08          /* Merker                 */
#define EB_PA       0x09          /* Eingangsbereich        */
#define AB_PA       0x0A          /* Ausgangsbereich        */
#define ABS_PA      0x0F          /* absoluter Block im PA */
```

2.3.4.11.7 Auftragsart für CP__Call

```
#define GET_INFO        = 0x00
#define STATR_AG        = 0x20
#define READ_AG         = 0x21
#define WRITE_AG        = 0x21
#define CNCLR_AG        = 0x28
#define STAT_PA         = 0x70
#define READ_PA         = 0x71
#define HANDLE_Struct   = 0x7F      /* dx:bx = Zeiger auf CP486_
                                       Parameter_Rec */
```

2.3.4.11.8 Konstanten für Fehlermeldungen: Kachel 2, 3 und 7

```
#define ERR_S5_TYP = 0x01;
```
/* ungültiger Elementtyp */

Bei Einzelelementzugriff mit Elementtyp DX_SNG, BA_SNG, BB_SNG, BT_SNG oder QB_SNG bzw. bei Blockelementzugriff mit Elementtyp DX_BLK, BA_BLK, BB_BLK, BT_BLK oder FX_BLK wurde versucht auf Daten in einem AG der 115U-Reihe zuzugreifen. Diese Elementtypen existieren in diesem AG-Typ jedoch nicht.

Abhilfe: Parameter "`typ`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

```
#define ERR_S5_BST = 0x02;
```
/* Baustein nicht vorhanden */

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementtyp DB_BLK wurde versucht auf einen nicht existierenden Baustein zuzugreifen.

Abhilfe: Datenbaustein im AG anlegen oder Parameter "bst" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

```
#define ERR_S5_ELM = 0x03;
```
/* Element nicht vorhanden */

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementzugriff mit Elementtyp DB_BLK wurde versucht auf Daten in einem DB zuzugreifen, die nicht vorhanden sind.

Abhilfe: Datenbaustein im AG entsprechend verlängern oder Parameter "`adr`" bzw. "`len`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten bzw. Zählerstände mit einer Nummer > 127 zuzugreifen.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG wurde versucht auf Merker mit Nummer > 199 bei Elementgröße Byte, mit Nummer > 198 bei Elementgröße Wort oder mit Nummer > 196 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware auf Wertigkeit überprüfen.

Bei Einzelelementzugriff mit Elementtyp EB_ - oder AB_SNG wurde versucht auf das Prozeßabbild des E/A-Bereichs mit Nummer > 127 bei Elementgröße Byte, mit Nummer > 126 bei Elementgröße Wort oder mit Nummer > 124 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware auf Wertigkeit überprüfen.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht auf Elemente der P-Peripherie mit Nummer > 255 bei Elementgröße Byte, mit Nummer > 254 bei Elementgröße Wort oder mit Nummer > 252 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware auf Wertigkeit überprüfen.

`#define ERR_S5_SIZE = 0x04;` /* ungültige Elementgröße */

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten oder Zählerstände zuzugreifen, wobei der Parameter Elementgröße nicht auf Wortzugriff (WORD_ELM) gesetzt war.

Abhilfe: Parameter "`size`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG wurde versucht mit dem Parameter Elementgröße RBYTE_ELM auf Merker oder absolute Adressen zuzugreifen.

Abhilfe: Parameter "`size`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht mit dem Parameter Elementgröße SEMA_ELM oder RBYTE_ELM auf die Ein- bzw. Ausgängen im Prozeßabbild zuzugreifen.

Abhilfe: Parameter "`typ`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht mit dem Parameter Elementgröße BIT_ELM, SEMA_ELM oder RBYTE_ELM auf die P-Peripherie zuzugreifen.

Abhilfe: Parameter "`typ`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei lesendem Einzelelementzugriff mit Elementtyp ABS_SNG wurde versucht von absoluten Adressen mit Elementgröße SEMA_ELM zu lesen. Diese Zugriffsart ist unter absoluter Adressierung nur schreibend möglich. Wenn einzelne Bits gelesen werden sollen, ist die Elementgröße BIT_ELM zu verwenden.

Abhilfe: Parameter "`typ`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`#define ERR_S5_BIT = 0x05;` /* Bit-Nummer zu groß */

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG und der Elementgröße BIT_ELM oder SEMA_ELM wurde versucht, auf ein Merker- oder Absolutadreß-Bit mit einer Bitnummer > 7 (15) zuzugreifen.

Abhilfe: Parameter "`bit`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht auf ein E/A-BIT mit einer Bitnummer > 7 zuzugreifen.

Abhilfe: Parameter "`bit`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`#define ERR_S5_STRT = 0x06;` /* ungültige Anfangsadresse */

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde ein Blocktransfer über Bausteine versucht, wobei die relative Anfangsadresse im Block > 32767 ist.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`#define ERR_S5_LEN = 0x07;` /* ungültige Blocklänge */

Bei Blockelementzugriff unter allen Elementtypen wurde ein Blocktransfer mit einer Länge > 504 Worten versucht.

Abhilfe: Parameter "`len`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

`#define ERR_S5_ADR = 0x08;` /* Adresse zu groß */

Bei Einzel- oder Blockelementzugriff mit Elementtyp ABS_SNG wurde versucht auf eine Adresse > FFFFh in einem AG der 115U-Reihe zuzugreifen. Die CPUs (bis CPU 944) haben jedoch nur 64 KB Adreßraum.

Abhilfe: Parameter "`adr`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

```
#define ERR_S5_QVZ = 0x09;
```
/* QVZ/ADF im AG bei lesen/schreiben */

Es wurde versucht auf einem Adreßbereich zuzugreifen, der physikalisch nicht vorhanden ist. Die AGs der 135 und 155-Reihe stellen diese Fehlermeldungen zur Verfügung. Ein AG der 115U-Reihe würde hierbei in den Stop-Zustand versetzt werden.

Abhilfe: Parameter "`typ`" bzw. "`adr`" im Funktionsaufruf der PC-Anwendersoftware korrigieren.

```
#define ERR_S5_944 = 0x0A;
```
/* CPU 944: Baustein im Prog.Bank */

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde versucht auf einen Baustein zuzugreifen, der nicht in der DB-Bank liegt.
(Betrifft nur CPU 944 vom AG-Typ 115U)

Abhilfe: Baustein im AG in DB-Bank anlegen (über BIB Nr. 19285) oder Funktionsaufruf in der PC-Anwendersoftware korrigieren.

2.3.4.12 Ablage der Prozeßabbilder in der Kachel 7

Das Prozeßabbild kann vom Anwender auch direkt ausgelesen werden. Folgende Übersicht zeigt wie die Kachel 7 aufgeteilt ist. Der direkte Zugriff ist sehr schnell möglich:

```
Adresse in der
Kachel (hex)

Byte   0 Prozeßabbild EB 0                        +
       .                                         |
       .                                         | 128 Byte PAE 0-127
       .                                         |
Byte 127 Prozeßabbild EB 127                     +
Byte 128 Prozeßabbild AB 0                       +
       .                                         |
       .                                         | 128 Byte PAA 0-127
       .                                         |
Byte 255 Prozeßabbild AB 127                     +
Byte 256 Merkerbyte 0                            +
       .                                         |
       .                                         | 256 Byte Merker 0-255
       .                                         |
Byte 511 Merkerbyte 255       +                  |
Byte 512/513 Timer 0   (high/low)                +
       .                                         |
       .                                         | 128 Worte Timer 0-127
       .                                         |
Byte 766/767 Timer 127 (high/low)                +
Byte 768/769 Zähler  0 (high/low)                +
       .                                         |
       .                                         | 127 Worte Zähler 0-126
       .                                         |
Byte 1020/1021 Zähler 126 (high/low)             +
Byte 1022 Zählbyte 1)                            + Zählbyte
Byte 1023 Interrupt auf CP auslösen
```

Hinweis

Alle Werte in dieser Kachel werden beim Aufruf des Hantierungsbausteins FB1/FB10 aufgefrischt (wenn dies am Formaloperanden des Hantierungsbausteins freigegeben ist). Nach jeder Auffrischung der Daten in der Kachel 7 erhöht der Hantierungsbaustein das Zählbyte um 1. An diesem Zählbyte kann der PC erkennen ob die Daten gültig sind und wie oft die Daten seit dem letzten Lesen aufgefrischt wurden. Gültig sind die Daten dann, wenn der Inhalt des Zählbytes im Bereich dual 1...255 liegt. Bei Überlauf beginnt das Zählbyte wieder bei 1.

Der Hantierungsbaustein Synchron setzt den laufenden Zähler, Adresse 3FEh, in der Kachel 7 auf 0. Hieran erkennt der PC daß die Daten in der Kachel 7 momentan nicht gültig sind.

Diese Kachel muß von der PC nicht gelöscht werden, wenn im Zählbyte (Adresse 3FEh der Kachel) eine 0 steht.

Auf diese Kachel greift der Hantierungsbaustein nur schreibend zu.

Der Hantierungsbaustein löst zur Auffrischung der Daten in Kachel 7 keinen Interrupt aus.

2.4 Betrieb des CP486COM unter WINDOWS

Ab der Version 2.2 der Tooldiskette liegt eine Programmlibrary für MS-WINDOWS 3.1 mit folgenden Dateien vor:

- Die Header-Datei CP486WIN.H und die OBJ-Datei CP486WIN.OBJ.
- Die Datei CP486WIN.H enthält die erforderlichen Definitionen für den Betrieb unter WINDOWS.
- Die Datei CP486WIN.OBJ enthält die Kommunikationsfunktionen unter WINDOWS. Die Funktionen werden, wie in Kapitel 2.3.4 für DOS beschrieben, aufgerufen. (Ausnahme: CP_stat_AG)

Änderungen im Funktionsaufruf

CP_stat_AG: CP_stat_AG (byte r) mit r = Auftragsnummer.

Neue Funktionen

CP_init (void): Legt einen Datenbereich für die Kommunikation auf der Kachel 4 an und gibt einen Zeiger auf diesen Bereich zurück.

CP_exit (void): Gibt den Datenbereich wieder frei. Dieser Befehl muß am Programm ende unbedingt ausgeführt werden.

Hinweis

In der SYSTEM.INI muß unter der Sektion [386Enh] zusätzlich zum Eintrag in der CONFIG.SYS mit dem Befehl *EMMExclude* = ... der Kachelbereich von der WINDOWS-Speicherverwaltung ausgeklammert werden!

Dies gilt in allen Fällen, in denen der CP486 unter WINDOWS 3.1 läuft, da WINDOWS den Kachelbereich nicht selbständig ausklammert.

Auf der Tooldiskette 2.2 ist ein Demobeispiel für den Betrieb unter WINDOWS enthalten.

3 Kopplung der SPS mit CP486NT

3 Kopplung der SPS mit CP486NT

3.1 Allgemeine Beschreibung

Der Datentransfer zwischen CP486 und SPS wird durch Hantierungsbausteine auf SPS-Seite und durch Software-Interrupts auf CP-Seite unterstützt. Folgende Routinen stehen zur Verfügung:

		Bedienung auf SPS-Seite	Bedienung auf CP-Seite
Kachel 2	CP-Auftrag: Daten von SPS lesen/schreiben **(CP3/4 aktiv)**	zyklisch aufgerufener Hantierungsbaustein (FB1)	Softwareinterrupt bei DOS Treiberaufruf WinNT
Kachel 7	Prozeßabbild an CP übertragen	zyklisch aufgerufener Hantierungsbaustein (FB1)	direkter Zugriff

Tab. 3-1: Routinen

Folgende Datenstrukturen in der SPS können auf CP-Seite angesprochen werden:

- einzelne Elemente im Format Bit, Byte, Wort und Doppelwort, DB, DX, BA, BB, BT, BS, Merker, Eingänge, Ausgänge, Timer, Zähler
- Datenblöcke DB, DX, MB, T, Z, BA, BB, BT, BS, FB, FX, OB, PB, SB

Die im folgenden beschriebenen Funktionen stehen ab CP486-Version 3.00 (Software CP4-SW593 Version 3.00) und Hantierungsbaustein-Version 3.00 (CP4-SW973 Version 3.00) zur Verfügung.

Das Programm CP486 wird in der folgenden Beschreibung als COM-Treiber bezeichnet.

3.2 Installation der Kachel-Software

3.2.1 SPS-Seite: Hantierungsbausteine

Für die Kommunikation mit der CP486 müssen die Hantierungsbausteine FB1 und FB2 in der SPS geladen werden. Der Hantierungsbaustein FB1 wird im OB1 aufgerufen, der Hantierungsbaustein FB2 in den Neustartbausteinen (OB20, OB21 und OB22).

Beispiel für den Aufruf des FB1 im OB1

```
         Baustein#OB1
         BIB
0000             ;SPA FB 1
         NAME    #CP-L/S
         ANSS    =KY 2,32
         PAA     =KF +0
         PAFE    =MB 99
0005             ;BE
```

Bez.	Format	Beschreibung
ANNS	KY	Anzahl der Aufträge
PAA	KF	Kennung Prozeßabbild
PAFE	MB	Merkerbyte für Fehlermeldungen

Tab. 3-2: Parameterliste für den Aufruf von FB1

ANSS AN Anzahl der Aufträge, die maximal bei einem Hantierungsbaustein-Aufruf auf der Kachel abgearbeitet werden sollen

SS Nummer der Basiskachel

PAA Kennung der Prozeßabbilder beim Aufruf des Hantierungsbausteins auf der Kachel aktualisieren

= 0 Prozeßabbilder werden nicht aktualisiert

≠ 0 Prozeßabbilder werden aktualisiert

Der angegebene Wert, der auf das Prozeßabbild übertragen werden soll, ist die Kachelnummer +1. Zulässige Kachelnummern sind 4 - 7.

PAFE Fehlerrückmeldung des Hantierungsbausteins

= 0 es ist kein Fehler aufgetreten

≠ 0 es ist ein Fehler aufgetreten. Die Fehlernummer wird im PAFE-Byte übergeben

1 Anzahl der Aufträge, die maximal bei einem Hantierungsbaustein-Aufruf abgearbeitet werden sollen ist 0.

2 Anzahl der Aufträge, die maximal bei einem Hantierungsbaustein-Aufruf abgearbeitet werden sollen ist größer als 127.

3 Die Basiskachelnummer ist nicht durch 8 teilbar

5 Die Kachel ist noch nicht von der CP synchronisiert.

9 Kachel für Prozeßabbild nicht im gültigem Bereich.

Benutzte Schmiermerker: MB200-MB255

Beispiel für den Aufruf des FB2 im OB21:

```
         Baustein#OB21
         BIB
0000             ;SPA FB 2
         NAME    #SYNCHRON
         SSNR    =KF +32
         WART    =KF +0
         PAA     =KF +7
         PAFE    =MB 98
0005             ;BE
```

Bez.	Format	Beschreibung
SSNR	KF	Nummer der Basiskachel
WART	KF	Art der Synchronisation
PAA	KF	Kennung Prozeßabbild
PAFE	BY	Merkerbyte für Fehlermeldungen

Tab. 3-3: Parameterliste für Aufruf von FB2

SSNR Nummer der Basiskachel

WART
- = 0 Der FB-SYNCHRON wartet nicht darauf bis die CP jede einzelne Kachel synchronisiert hat
- ≠ 0 Der FB-SYNCHRON wartet bei jeder einzelnen Kachel bis die CP diese Kachel synchronisiert hat

PAA Kachelnummer auf der das Prozeßabbild abgelegt werden soll.
Gültiger Bereich 4..7.

PAFE Fehlerrückmeldung des Hantierungsbausteines
- = 0 es ist kein Fehler aufgetreten
- ≠ 0 es ist ein Fehler aufgetreten:
  - 3 Basiskachelnummer ist nicht durch 8 teilbar.

Benutzte Schmiermerker: MB200-MB255

3.2.2 Unterschiedliche Darstellung von Daten im Speicher

Bei der Übertragung von Daten zwischen CP und AG muß die unterschiedliche Darstellung von Worten und Doppelworten (Langworten) auf CP und AG berücksichtigt werden.

Im CP sind Datenworte in anderer Form im Speicher abgelegt als im AG. Das höherwertige (High-Byte) und das niederwertige Byte (Low-Byte) sind vertauscht abgespeichert. Bei Doppelworten sind alle 4 Bytes genau im umgekehrter Reihenfolge abgespeichert. Werden zwischen AG und CP Daten vom Typ Wort, Doppelwort ausgetauscht, so muß irgendwann eine Vertauschung vorgenommen werden, da sonst nach der Übertragung falsche Daten vorliegen. Soweit dies sinnvoll möglich ist, nimmt der COM-Treiber die Anpassung der Daten automatisch vor.

Bei Kacheln 2 und Kachel 7 wird die Vertauschung in allen Fällen automatisch vom Treiber durchgeführt.

- Bei der Übertragung von Bytes findet keine Vertauschung statt.
- Bei der Übertragung von Worten werden High- und Low-Byte vertauscht.
- Bei der Übertragung von Langworten werden alle 4 Bytes in ihrer Reihenfolge umgedreht.

Darstellung von Daten im AG

Adresse n	Byte	Darstellung Byte
Adresse n	High-Byte	Darstellung Wort
Adresse n+1	Low-Byte	
Adresse n	High-Byte High-Word	Darstellung Doppelwort
Adresse n+1	Low-Byte High-Word	
Adresse n+2	High-Byte Low-Word	
Adresse n+3	Low-Byte Low-Word	

Darstellung von Daten im CP

Adresse n	Byte	Darstellung Byte
Adresse n	Low-Byte	Darstellung Wort
Adresse n+1	High-Byte	
Adresse n	Low-Byte Low-Word	Darstellung Doppelwort
Adresse n+1	High-Byte Low-Word	
Adresse n+2	Low-Byte High-Word	
Adresse n+3	High-Byte High-Word	

3.3 Betrieb des CP486COM unter Windows-NT

Ab der Version 2.xx der Tooldiskette liegen für WINDOWS-NT 3.51 folgende Dateien vor:

Include-Dateien

CP486DEF.H	Allgemeine Definitionen und Prototypen
CPHWDEF.H	Definitionen zu den CP-Baugruppen
DRVFCALL.H	Definition der Treiberaufrufe für DevIOCtrl
WMKTYPES.H	Definition der Aufruftypen der verschiedenen Compiler und Betriebssysteme

C-Dateien

CPTEST.C	Demoprogramm zum lesen und beschreiben der Kachel

Bibliotheksdateien

CPWK.LIB	Importbibliothek mit den exportierenden Funktionen der DLL
CPWKNT.DLL	dynamische Bibliotheksdatei für NT

Kacheltreiber

CPWK.SYS	Kacheltreiber für Windows-NT 3.51

Betriebssystem

CPWK.INI	Script-Datei für regini.exe
REGINI.EXE	nimmt Einträge in das Registrierungsscript des NT-Systems vor und ist zusammen mit "CPWK.INI" zu benutzen

3.3.1 Einbindung des Kacheltreibers in Windows-NT

Die Installation des Kacheltreibers setzt eine installierte Version von Windows-NT 3.51 auf einem VIPA 486-DX voraus.

Zur Registrierung und Aktivierung des Kacheltreibers unter Windows-NT 3.51 sind folgende Schritte erforderlich:

- Kopieren Sie die Datei "CPWK.SYS" in das Verzeichnis:

  ```
  \WINNT35\SYSTEM32\DRIVERS
  ```

- Kopieren Sie die Dateien "REGINI.EXE" und "CPWK.INI" in ein von Ihnen erstelltes Verzeichnis.
- Zur Registrierung in Ihrem NT-System verwenden Sie das Programm REGINI. Gehen Sie hierzu auf Ausführen in Ihrem START-Menü und starten Sie die Datei "REGINI.EXE" mit der Script-Datei "CPWK.INI" als Kommandozeilenparameter. REGINI bindet nun den Treiber in Ihr System ein.
- Führen Sie zur Treiberinitialisierung einen Systemneustart durch.
- Starten Sie den Kacheltreiber, wechseln Sie hierzu in *Hauptgruppe*, *Systemsteuerung*, *Geräte* und suchen Sie in der Liste der eingetragenen Geräte nach dem Eintrag:

  "VIPA Dual Port S5 Device Driver".

  Geben Sie für den Kacheltreiber die gewünschte "Startart" vor und starten Sie den Treiber. Unter der Spalte "Status" im Gerätefenster erscheint die Meldung "Gestartet".

Der Kacheltreiber ist jetzt bereit und kann über die unten beschriebenen Funktionsaufrufe von Ihrer Applikation angesprochen werden.

Für einen ersten Test können Sie die mitgelieferte Konsolenapplikation "CPTEST.EXE" starten. Wechseln Sie hierzu in die *Hauptgruppe* und starten Sie die "MS-DOS-Eingabeaufforderung". Beachten Sie, daß die Bibliotheksdatei "CPWK.DLL" im selben Verzeichnis vorhanden sein muß.

3.3.2 Schnittstelle zu Microsoft-Visual C V2.0, V4.0

Für Funktionsaufrufe des Kacheltreibers aus C-Programmen heraus, wurde ein Bibliotheks-File erstellt, das alle Funktionen zur Verfügung stellt. Zu jeder Funktion des Treibers ist eine entsprechende C-Funktion definiert.

Mit diesen Funktionen können aus einer auf der CP486 laufenden Applikation heraus, Daten aus dem AG gelesen bzw. in das AG geschrieben werde. Die Übergabe und Rückgabe von Parametern erfolgt über eine Struktur, die von Ihnen entsprechend vorzubelegen ist.

In dem Include-File "CP486DEF.H" sind Datentypen und Konstanten für Elementgrößen, Elementtypen und Fehlernummern definiert. Die Funktionsprototypen der nachfolgend beschriebenen Funktionen sind dort im ANSI-C Stil definiert. Das Include-File muß im Anwenderprogramm genannt werden.

Alle benötigten Funktionen sind in der DLL (CPWKNT.DLL) implementiert. Die DLL muß sich zusammem mit Ihrer Applikation in einem Verzeichnis befinden. Bei der Übersetzung Ihrer Applikation muß die Importbibliothek "CPWK.LIB" mit eingebunden werden. Je nach Programmier-Umgebung und Version ist die Datei in der Dependence-List (Projektliste) oder im Make-File aufzunehmen. Details hierzu sind den jeweiligen Handbüchern zu entnehmen.

Alle folgenden Funktionen verwenden die in "WMKTYPES" vereinbarten Typdefinitionen

```
#define WENTRY_C        pascal
#define W_POINTER       *
```

3.3.2.1 CP-Funktionen Initialisieren

```
unsigned  short  WENTRY_C CP_init (void)
```

Aufrufparameter

keine

Rückgabe

`0`	kein Fehler
`CP_NO_DRIVER`	Fehler, siehe Kapitel 3.3.4

Diese Funktion muß vor Benutzung weiterer CP-Funktionen aufgerufen werden. Sie überprüft, ob der Windows-NT Treiber installiert ist, öffnet und initialisiert die Device Schnittstelle.

3.3.2.2 CP-Funktionen beenden

```
void  WENTRY_C CP_exit (void)
```

Aufrufparameter

keine

Rückgabe

keine

Diese Funktion schließt die Device-Schnittstelle und ist bei Programmende aufzurufen.

3.3.2.3 CP-Funktion lesen

```
unsigned short WENTRY_C CP_startread (CP386_PARAMETER
         W_POINTER cp)
```

Aufrufparameter

`CP386_PARAMETER W_POINTER cp`	Zeiger auf Struktur, mit den Einträgen über die zu lesenden Daten aus der SPS.

Rückgabe

`0`	kein Fehler
<> `0`	Systemfehler
`cp → Fehler`	Fehler, siehe Kapitel 3.3.4

Diese Funktion startet einen Leseauftrag. Die entsprechenden Einträge in den Strukturvariablen der zu übergebenden Struktur müssen vorher vorgenommen werden (siehe auch Kapitel 3.3.3).

Strukturvariablen

Übergabe

Elementtyp	`cp → Elementtyp`	Einzel- oder Blockelemente
Bausteinnummer	`cp → Bausteinnummer`	nur bei Bausteinelementen
Datentyp	`cp → Kennung`	Datentyp für Einzel- oder Blockelement
Länge	`cp → Len`	Anzahl der zu lesenden Elemente bzw. die Bitnummer bei Einzel-Bitelement lesen

Ergebnis

Auftragsnummer	`cp → Handle`	Eindeutige Auftragsnummer
Status	`cp → Fehler`	Status der Verbindung

Es kann ein Einzelelement oder ein Blockelement gelesen werden. Die Funktion startet den Auftrag und kehrt sofort wieder zum Aufrufer zurück. Die Daten selbst können daher erst durch einen Aufruf der Statusfunktion "CP_pollread" gelesen werden.

Hinweis

Um sicherzustellen, daß ein fertiger Auftrag nicht durch einen neuen Auftrag überschrieben wird bevor die Daten abgeholt werden, wird der Auftrag blockiert. Nach dem Starten eines Leseauftrags muß dessen Status solange abgefragt werden, bis der Auftrag mit oder ohne Fehler fertig ist.

Die Daten werden bei der Status-Abfrage am Ende der Struktur ab "cp → Daten[0]" kopiert. Erfolgt keine Status-Abfrage, so bleibt der Auftrag blockiert und weitere Aufträge können nicht bearbeitet werden.

3.3.2.4 CP-Funktion Abfrage lesen fertig

`unsigned short WENTRY_C` **`CP_pollread`** `(CP386_PARAMETER W_POINTER cp)`

Aufrufparameter

`CP386_PARAMETER W_POINTER cp`	Zeiger auf Struktur, mit den Einträgen über die zu lesenden Daten aus der SPS.

Rückgabe

`0`	kein Fehler
<> 0	Systemfehler
`cp → Fehler`	Fehler, siehe Kapitel 3.3.4

Mit dieser Funktion können Sie überprüfen, ob zu einem Leseauftrag die angeforderten Daten schon vorliegen. Vor einem Aufruf dieser Funktion ist immer ein "CP_startread" erforderlich.

Die Funktion schreibt in einige Strukturvariablen und liefert die angeforderten Daten.

Strukturvariablen

Status	`cp → Fehler`	Status der Verbindung
Daten	`cp → Daten[0]`	ab hier werden die Daten angeliefert

Die Daten werden von dieser Funktion in das Datenfeld "cp → Daten" am Ende der Struktur "CP386_PARAMETER" geschrieben. Beachten Sie, daß Sie ein entsprechend großes Datenfeld reservieren. Die Größe des Datenfeldes können Sie folgendermaßen angeben:

```
CP_386_PARAMETER *cp;
int bufferSize=1024;
.
.
cp=(CP386_PARAMETER *) malloc(sizeof(CP386_PARAMETER *) +
bufferSize));
```

Hinweis

Wortbreite Daten und Datenblöcke werden bei der Übertragung byteweise vertauscht.

Bei doppelwortbreiten Daten und Datenblöcken wird die Reihenfolge der 4 Bytes umgedreht.

3.3.2.5 CP-Funktion lesen abbrechen

```
unsigned short WENTRY_C CP_stopread (CP386_PARAMETER W_POINTER cp)
```

Aufrufparameter

`CP386_PARAMETER W_POINTER cp`	Zeiger auf Struktur, mit den Einträgen über die zu lesenden Daten aus der SPS.

Rückgabe

`0`	kein Fehler
<> 0	Systemfehler
`cp → Fehler`	Fehler, siehe Kapitel 3.3.4

Diese Funktion bricht den laufenden Leseauftrag ab.

3.3.2.6 CP-Funktion schreiben

```
unsigned short WENTRY_C CP_startwrite (CP386_PARAMETER
         W_POINTER cp)
```

Aufrufparameter

`CP386_PARAMETER W_POINTER cp`	Zeiger auf Struktur mit den Einträgen über die zu schreibenden Daten in die SPS.

Rückgabe

`0`	kein Fehler
<> 0	Systemfehler
`cp → Fehler`	Fehler, siehe Kapitel 3.3.4

Diese Funktion startet einen Schreibauftrag. Die entsprechenden Einträge in den Strukturvariablen der zu übergebenden Struktur müssen zuvor vorgenommen werden (siehe auch Kapitel 3.3.3).

Strukturvariablen

Übergabe

Elementtyp	`cp → Elementtyp`	Einzel- oder Blockelemente
Bausteinnummer	`cp → Bausteinnummer`	nur bei Bausteinelementen
Datentyp	`cp → Kennung`	Datentyp für Einzel- oder Blockelement
Elementoffset	`cp → Adresse`	Offset des Elementtyps
Länge	`cp → Len`	Anzahl der zu schreibenden Elemente bzw. die Bitnummer bei Einzel-Bitelementen
Daten	`cp → Daten`	zu schreibende Daten

Ergebnis

Auftragsnummer	`cp → Handle`	Eindeutige Auftragsnummer
Status	`cp → Fehler`	Status der Verbindung

Mit dieser Funktion kann ein einzelnes Element oder ein Datenblock in den Speicher des AGs geschrieben werden. Die Funktion überträgt deren Wert in die Kachel und kehrt sofort wieder zum Aufrufer zurück, wartet also nicht, bis das AG die Daten abholt. Der Status kann daher erst durch einen Aufruf der Statusfunktion "CP_pollwrite" gelesen werden.

Hinweis

Nach dem Starten eines Schreibauftrages muß dessen Status solange abgefragt werden bis der Auftrag "Fertig mit Fehler" oder "Fertig ohne Fehler" ist. Die Daten werden immer am Ende der Struktur ab "p → Daten[0]" kopiert. Erfolgt keine Statusabfrage bleibt der Auftrag blockiert und weiteren Aufträge können nicht bearbeitet werden.

3.3.2.7 CP-Funktion Abfrage schreiben fertig

```
unsigned  short  WENTRY_C  CP_pollwrite (CP386_PARAMETER
          W_POINTER  cp)
```

Aufrufparameter

`CP386_PARAMETER W_POINTER cp`	Zeiger auf Struktur, mit den Einträgen über die zu schreibenden Daten in die SPS.

Rückgabe

`0`	kein Fehler
<> 0	Systemfehler
`cp → Fehler`	Fehler, siehe Kapitel 3.3.4

Diese Funktion prüft, ob für einen gestarteten Schreibauftrag die Daten abgeholt wurden. Vor einem Aufruf dieser Funktion ist immer ein "CP_startwrite" erforderlich. Davor sind die entsprechenden Einträge in den Strukturvariablen der zu übergebenden Struktur vorzunehmen (siehe Kapitel 3.3.3). Diese Funktion liefert die zu schreibenden Daten an den Windows NT-Treiber und beschreibt die folgende Strukturvariable.

Status	`cp → Fehler`	Status der Verbindung

Die Daten werden von dieser Funktion aus dem Datenfeld "cp → Daten" am Ende der Struktur "CP386_PARAMETER" abgeholt. Beachten Sie, daß die entsprechende Größe des Datenfeldes von Ihnen reserviert werden muß, und daß die Daten noch vor dem Aufruf der Funktion "CP_startwrite" bereitstehen. Die Größe des Datenfeldes kann folgendermaßen gesetzt werden:

```
CP_386_PARAMETER *cp;
int bufferSize=1024;
.
.
cp=(CP386_PARAMETER *) malloc(sizeof(CP386_PARAMETER *) + bufferSize));
```

Hinweis

Wortbreite Daten und Datenblöcke werden bei der Übertragung byteweise vertauscht. Bei doppelwortbreiten Daten und Datenblöcken wird die Reihenfolge der 4 Bytes umgedreht.

3.3.2.8 CP-Funktion schreiben abbrechen

```
unsigned short WENTRY_C CP_stopwrite (CP386_PARAMETER
         W_POINTER cp)
```

Aufrufparameter

`CP386_PARAMETER W_POINTER cp`	Zeiger auf Struktur, mit den Einträgen über die zu schreibenden Daten aus der SPS.

Rückgabe

`0`	kein Fehler
<> 0	Systemfehler
`cp → Fehler`	Fehler, siehe Kapitel 3.3.4

Diese Funktion bricht den laufenden Schreibauftrag ab.

3.3.2.9 CP-Funktion allgemein

```
unsigned short WENTRY_C CP_Call (CP386_PARAMETER W_POINTER cp)
```

Aufrufparameter

`CP386_PARAMETER W_POINTER cp`	Zeiger auf Struktur mit den Einträgen in der Struktur vom Typ "CP386_PARAMETER"

Rückgabe

`0`	kein Fehler
<> 0	Systemfehler
`cp → Fehler`	Fehler, siehe Kapitel 3.3.4

Diese Funktion ersetzt alle oben beschriebenen Funktionsaufrufe und kann an deren Stelle verwendet werden. Für die Benutzung dieser Funktion ist die entsprechende Funktionsnummer (siehe Kapitel 3.3.4) als Auftragsart vorzubesetzen.

Auftragsarten

Auftragsarten, einzutragen in "cp → Auftrag":

`CP_START_READ`	Lesen Starten
`CP_POLL_READ`	Lesestatus abfragen und Daten lesen
`CP_START_WRITE`	Schreiben starten
`CP_POLL_WRITE`	Schreibstatus abfragen und Daten schreiben
`CP_GETKACHEL`	Kachelinformationen lesen
`CP_SETKACHEL`	Kachelbasisnummer setzen

Zusätzlich zu den oben beschriebenen Funktionen kann das Prozeßabbild auch mit den folgenden Funktionen gelesen werden:

`READ_PA`	Prozeßabbild lesen
`STAT_PA`	Status Prozeßsabbild lesen

Hinweis

Mit der Prozeßabbildfunktion kann ein Bereich des aktuellen Prozeßabbilds gelesen werden. Bei Zugriff auf die einzelnen Bereiche wird die Länge des Bereichs überwacht, z.B. kann ab EB126 nicht mit der Länge von 4 Bytes gelesen werden, da nur 128 Byte EB vorhanden sind.

Bei Zugriff auf Timer und Zähler erfolgt die Längenangabe in Worten, bei allen anderen Typen in Bytes. Bei Timer und Zähler wird auch beim Transfer das High- und Low-Byte jedes Worts getauscht, so daß die Daten im CP korrekt als Worte verarbeitet werden können.

Durch Angabe des Typs "ABS_SNG" kann ein beliebiger Ausschnitt des Prozeßabbilds bereichsübergreifend gelesen werden. Die Längenangabe erfolgt in Bytes, auch wenn aus dem Timer oder Zählerbereich gelesen wird. Wird ein Bereich der Timer und Zähler gelesen, so werden dabei High- und Low-Byte ebenfalls wieder getauscht !

Elementtypen, einzutragen in "cp → Elementtyp" und die Anzahl in "cp → Len"

Z_SNG	Zähler (Länge in Worten)
T_SNG	Timer (Länge in Worten)
MB_SNG	Merker (Länge in Bytes)
EB_SNG	Eingänge (Länge in Bytes)
AB_SNG	Ausgänge (Länge in Bytes)
ABS_SNG	Absoluter Zugriff auf Prozeßabild (Länge in Byte)

3.3.2.10 CP- Kachelparameter setzen

```
unsigned short WENTRY_C CP_setkachel (KACHEL_STRUCT W_POINTER
         kachel)
```

Aufrufparameter

`KACHEL_STRUCT W_POINTER kachel`	Zeiger auf Struktur, mit den Einträgen über die zu setzende Kachel.

Rückgabe

`0`	kein Fehler
<> 0	Systemfehler
`kachel → Fehler`	Fehler, siehe Kapitel 3.3.4

Vor dem Aufruf dieser Funktion ist die zu übergebende Struktur vorzubesetzen (siehe Kapitel 3.3.3). Die Kachel-Basisnummer kann in der Struktur "KACHEL_STRUCT" übergeben werden.

Die Kachel-Basisnummer "kachel → Kachelbasis" kann beim CP4BG6x in 8er Schritten von 0-240 gesetzt werden. Beim CP4BG7x wird die Kachel-Basisnummer in 16er Schritten eingestellt.

3.3.2.11 CP- Kachelparameter lesen

```
unsigned short WENTRY_C CP_getkachel (KACHEL_STRUCT W_POINTER
        kachel)
```

Aufrufparameter

`KACHEL_STRUCT W_POINTER kachel`	Zeiger auf Struktur, die mit Daten über die Kachel beschrieben wird.

Rückgabe

`0`	kein Fehler in
<> 0	Systemfehler
`kachel → Fehler`	Fehler, siehe Kapitel 3.3.4

Vor dem Aufruf dieser Funktion ist es sinnvoll, die zu übergebende Struktur mit "0" zu initialisieren. Informationen über die verwendete Kachel werden zurückgeliefert (siehe Kapitel 3.3.3).

3.3.3 Strukturbeschreibungen

Hier finden Sie die Strukturen, die Ihnen als Datenschnittstelle für die Benutzung der CP-Funktionen zur Verfügung stehen. Die Typ-Definitionen befinden sich in den Include-Dateien CP486DEF.H und CPHWDEF.H. Beachten Sie auch die im Kapitel 3.3.4 aufgeführten Definitionsvereinbarungen, die Sie für die Parametrierung der Strukturelemente verwenden sollten.

Auszug aus "CP486DEF.H":

```
typedef struct {
unsigned char Elementtyp;           //Elementtypen der SPS auf die zugegriffen
                                          //werden kann
unsigned char Auftrag;              //Auftragsart (muß vom Anwender nur bei
                                    //Funktion "CP_call" gesetzt werden.
unsigned char Bausteinnummer;       //Nummer des Bausteins sonst 0 bei Merkern,
                                    //Ausgängen usw.
unsigned char Kennung;              //gibt die Elementbreite an (Bit, Byte, Wort
                                    //Doppelwort)
unsigned short Adresse;             //Startadresse im Element beim schreiben
unsigned short Len;                 //Anzahl der übergebener Elemente bzw.
                                    //Bitnummer bei Einzelelement Bit
unsigned char far *ptr;             //16:16 Realmode-Zeiger auf Daten DOS.
                                    //Bei Verwendung der Daten in der Struktur,
                                    //"ptr" auf NULL setzen
unsigned char far *ptr_win;         //16:16 Realmode Zeiger auf Daten
                                    //Windows. Bei 32 Bit Flat-Model 32Bit Daten
                                    //auf NULL setzen
unsigned short far pascal*ConfirmFunction)
(unsigned char Auftragsnummer);
                                    //Rückruffunktion wenn Auftrag fertig
unsigned char Cpu                   //CPU Nummer in der SPS 0 - 3, bei einer CPU
                                    //immer 0
unsigned char Handle;               //Handle über den Auftrag
unsigned short Fehler;              //Fehlercode. 0 = kein Fehler sonst siehe
                                    //Konstanten für Fehlermeldungen Kachel 2
unsigned char Daten[1];             //Platz für die Daten. Wird verwendet wenn
                                    //ptr==NULL und ptr_win==NULL
}CP386_PARAMETER;
```

Unter Windows-NT sind die folgenden Strukturelemente nicht benutzbar:

- `*ptr`
- `*ptr_win`
- `*ConfirmFunction`

Statt *ptr bzw. *ptr_win ist das Feld Daten als Datenbuffer zu verwenden. Beachten Sie dabei, daß "cp → Daten" entsprechend den zu lesenden bzw. schreibenden Daten zu reservieren ist (siehe auch Funktionsbeschreibung "CP_pollread" und "CP_pollwrite").

Auszug aus "CPHWDEF.H":

```
typedef struct
{

  unsigned short Cb;                //Länge der Struktur in Bytes(wird intern
                                    //besetzt)
  unsigned short Fehler;            //Fehlercode 0=kein Fehler
  unsigned short Seriennummer;      //Seriennummer der CP
  short Kachelbasis;                //Eingestellte Kachelbasis
  short KachelAnzahl;               //Anzahl der Kacheln
  short TreiberVersion;             //Version des Treibers 100 -> 1.00
  shortCPTyp;                       //CP3, CP4, BG81, ..
  short KachelTyp;                  //Kachel, Typen

}KACHEL_STRUCT;
```

3.3.4 Allgemeine Definitionen und Fehlerdefinitionen

Die folgenden Konstanten sind bereits vordefiniert. Aufgrund der besseren Lesbarkeit und Übersichtlichkeit wird empfohlen, diese Konstanten auch im Programmtext zu verwenden. Eine Anpassungen an spätere Änderungen des Kacheltreibers kann so leichter vorgenommen werden.

3.3.4.1 Allgemeine Definitionen

Funktionsnummern

nur bei Funktionsaufruf "CP_Call" zu verwenden und in "cp → Auftrag" zu setzen

```
#define CP_START_READ         Leseauftrag starten
#define CP_POLL_READ          Leseauftrag pollen starten
#define CP_START_WRITE        Schreibauftrag starten
#define CP_POLL_WRITE         Schreibauftrag pollen starten
#define CP_GETKACHEL          Auftrag Kachelinfo lesen
#define CP_SETKACHEL          Auftrag Kachelinfo schreiben
#define READ_PA               Prozeßabbild lesen
#define STAT_PA               Status des Prozeßabbildes lesen
```

Elementtypen bei Einzelelementen

zu setzen in "cp → Elementtyp"

```
#define DB_SNG  0x00     //DB
#define DX_SNG  0x01     //DB im Externspeicher
#define BA_SNG  0x02     //BA
#define BB_SNG  0x03     //BB
#define BS_SNG  0x04     //BS
#define BT_SNG  0x05     //BT
#define Z_SNG   0x06     //Zähler
#define T_SNG   0x07     //Timer
#define MB_SNG  0x08     //Merker
#define EB_SNG  0x09     //Eingangsbereich
#define AB_SNG  0x0A     //Ausgangsbereich
#define PY_SNG  0x0B     //P-Peripherie
#define QY_SNG  0x0C     //Q-Peripherie
#define ABS_SNG 0x0F     //Absoluter Speicher
#define FB_SNG  0x10     //FB
#define FX_SNG  0x11     //FB im Externspeicher
#define OB_SNG  0x12     //OB
#define PB_SNG  0x13     //PB
#define SB_SNG  0x14     //SB
```

Elementtypen bei Blockelementen
zu setzen in "cp → Elementtyp"

```
#define DB_BLK          0x00        //DB
#define DX_BLK          0x01        //DB im Externspeicher
#define BA_BLK          0x02        //BA
#define BB_BLK          0x03        //BB
#define BS_BLK          0x04        //BS
#define BT_BLK          0x05        //BT
#define Z_BLK           0x06        //Zähler
#define T_BLK           0x07        //Timer
#define MB_BLK          0x08        //Merker
#define EB_BLK          0x09        //Eingangsbereich
#define AB_BLK          0x0A        //Ausgangsbereich
#define PY_BLK          0x0B        //P-Peripherie
#define QY_BLK          0x0C        //Q-Peripherie
#define ABS_BLK         0x0F        //Absoluter Speicher
#define FB_BLK          0x10        //FB
#define FX_BLK          0x11        //FB im Externspeicher
#define OB_BLK          0x12        //OB
#define PB_BLK          0x13        //PB
#define SB_BLK          0x14        //SB
```

Datentyp bei Einzelelementen
zu setzen in "cp → Kennung"

```
#define BIT_ELM          0x00      //Bit
#define BYTE_ELM         0x02      //Byte
#define LBYTE_ELM        0x02      //linkes Byte eines Wortes
#define RBYTE_ELM        0x03      //rechtes Byte eines Wortes
#define WORD_ELM         0x04      //Wort
#define DWORD_ELM        0x05      //Doppelwort
#define BLOCK_ELM        0x07      //Block
```

Datentyp bei Blockelementen
zu setzen in "cp → Kennung"

```
#define B_BLOCK         0x07        //Typ: Block von Bytes
#define W_BLOCK         0x17        //Typ: Block von Worten
#define D_BLOCK         0x27        //Typ: Block von Langworten
```

Kennungen für Auftragsstati
Diese Meldungen erhalten Sie für einen Auftrag der mit einer Poll-Funktion abgefragt wird in "p → Fehler" zurückgeliefert

```
#define REQ_NO_ERR      0x00        // Auftrag fertig ohne Fehler
#define REQ_WRKN        0x01        // Auftrag in Bearbeitung
#define REQ_UNDEF       0x02        // Auftragsstatus undefiniert
```

3.3.4.2 Fehlerdefinitionen

Allgemeine Fehlerdefinition:

Nur Funktion "CP_init" im Fehlerfall

```
#define CP_NO_DRIVER     1    //NT-Treiber nicht installiert
```

Fehlermeldungen Kachel 2 und 7:

wird in "p → Fehler" zurückgeliefert

Allgemein

```
#define ILL_TYPE     100  //ungültiger Elementtyp
#define ERR_LEN      101  //Längenfehler
#define LL_ELMSZ     102  //ungültige Elementgröße
#define CPU_ERR      103  //Elementtyp bei CPU nicht möglich
#define ERR_KFULL    104  //Kachel voll
#define ERR_COORD    105  //Zugriff auf Kachel nicht möglich
#define ERR_BLKD     106  //Kachel noch blockiert
#define ERR_REQNR    107  //falsche Auftragsnummer
#define ERR_DPTR     108  //fehlerhafter Quell-/Zieldatenzeiger
#define ERR_NOREQ    109  //Auftrag nicht in Bearbeitung
#define ERR_ILL_CALL 110  //ungültiger Funktionsaufruf
```

SPS-spezifisch

Diese Meldungen erhalten Sie für einen Auftrag der mit einer Poll-Funktion abgefragt wird im Fehlerfall.

```
#define ERR_S5_TYP 0xFF01
```
// ungültiger Elementtyp

Bei Einzelelementzugriff mit Elementtyp DX_SNG, BA_SNG, BB_SNG, BT_SNG oder QB_SNG bzw. bei Blockelementzugriff mit Elementtyp DX_BLK, BA_BLK, BB_BLK, BT_BLK oder FX_BLK wurde versucht auf Daten in einem AG der 115U-Reihe zuzugreifen. Diese Elementtypen existieren in diesem AG-Typ jedoch nicht.

Abhilfe: Strukturvariable "Elementtyp" vor Funktionsaufruf in der Applikation korrigieren.

```
#define ERR_S5_BST 0xFF02
```
// Baustein nicht vorhanden

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementtyp DB_BLK wurde versucht auf einen nicht existierenden Baustein zuzugreifen.

Abhilfe: Datenbaustein im AG anlegen oder Strukturvariable "Bausteinnummer" vor Funktionsaufruf in der Applikation korrigieren.

```
#define ERR_S5_ELM 0xFF03
```
// Element nicht vorhanden

Bei Einzelelementzugriff mit Elementtyp DB_SNG bzw. bei Blockelementzugriff mit Elementtyp DB_BLK wurde versucht auf Daten in einem DB zuzugreifen, die nicht vorhanden sind.

Abhilfe: Datenbaustein im AG entsprechend verlängern oder Strukturvariable "Adresse" bzw. "Len" vor Funktionsaufruf in der Applikation korrigieren.

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten bzw. Zählerstände mit einer Nummer > 127 zuzugreifen.

Abhilfe: Parameter "Adresse" vor Funktionsaufruf in der Applikation korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG wurde versucht auf Merker mit Nummer > 199 bei Elementgröße Byte, mit Nummer > 198 bei Elementgröße Wort oder mit Nummer > 196 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Strukturvariable "Adresse" vor Funktionsaufruf in der Applikation überprüfen.

Bei Einzelelementzugriff mit Elementtyp EB_- oder AB_SNG wurde versucht auf das Prozeßabbild des E/A-Bereichs mit Nummer > 127 bei Elementgröße Byte, mit Nummer > 126 bei Elementgröße Wort oder mit Nummer > 124 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Strukturvariable "Adresse" vor Funktionsaufruf in der Applikation überprüfen.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht auf Elemente der P-Peripherie mit Nummer > 255 bei Elementgröße Byte, mit Nummer > 254 bei Elementgröße Wort oder mit Nummer > 252 bei Elementgröße Doppelwort zuzugreifen.

Abhilfe: Strukturvariable "Adresse" vor Funktionsaufruf in der Applikation überprüfen.

```
#define ERR_S5_SIZE 0xFF04
```
// ungültige Elementgröße

Bei Einzelelementzugriff mit Elementtyp Z_SNG oder T_SNG wurde versucht auf Zeiten oder Zählerstände zuzugreifen, wobei die Strukturvariable "Kennung" nicht auf Wortzugriff (WORD_ELM) gesetzt war.

Abhilfe: Strukturvariable "Kennung" vor Funktionsaufruf in der Applikation korrigieren.

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG wurde versucht, mit der Strukturvariablen "Kennung" RBYTE_ELM auf Merker oder absolute Adressen zuzugreifen.

Abhilfe: Strukturvariable "Kennung" vor Funktionsaufruf in der Applikation korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht, mit der Strukturvariable "Kennung" SEMA_ELM oder RBYTE_ELM auf den Ein- bzw. Ausgängen im Prozeßabbild zuzugreifen.

Abhilfe: Strukturvariable "Elementtyp" vor Funktionsaufruf in der Applikation korrigieren.

Bei Einzelelementzugriff mit Elementtyp PB_SNG wurde versucht mit der Strukturvariable "Kennung" BIT_ELM, SEMA_ELM oder RBYTE_ELM auf die P-Peripherie zuzugreifen.

Abhilfe: Strukturvariable "Elementtyp" vor Funktionsaufruf in der Applikation korrigieren.

Bei lesendem Einzelelementzugriff mit Elementtyp ABS_SNG wurde versucht von absoluten Adressen mit Elementgröße SEMA_ELM zu lesen. Diese Zugriffsart ist unter absoluter Adressierung nur schreibend möglich! Wenn einzelne Bits gelesen werden sollen, ist die Elementgröße BIT_ELM zu verwenden.

Abhilfe: Strukturvariable "Elementtyp" vor Funktionsaufruf in der Applikation korrigieren.

`#define ERR_S5_BIT 0xFF05` // Bit-Nummer zu groß

Bei Einzelelementzugriff mit Elementtyp MB_SNG oder ABS_SNG und der Elementgröße BIT_ELM oder SEMA_ELM wurde versucht, auf ein Merker- oder Absolutadreß-Bit mit einer Bitnummer > 7 (15) zuzugreifen.

Abhilfe: Strukturvariable "Len" vor Funktionsaufruf in der Applikation korrigieren.

Bei Einzelelementzugriff mit Elementtyp EB_SNG oder AB_SNG wurde versucht auf ein E/A-BIT mit einer Bitnummer > 7 zuzugreifen.

Abhilfe: Strukturvariable "Len" vor Funktionsaufruf in der Applikation korrigieren.

`#define ERR_S5_STRT 0xFF06` // ungültige Anfangsadresse

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde ein Blocktransfer über Bausteine versucht, wobei die relative Anfangsadresse im Block > 32767 ist.

Abhilfe: Strukturvariable "Adresse" vor Funktionsaufruf in der Applikation korrigieren.

`#define ERR_S5_LEN 0xFF07` // ungültige Blocklänge

Bei Blockelementzugriff unter allen Elementtypen wurde ein Blocktransfer mit einer Länge > 504 Worten versucht.

Abhilfe: Parameter "Len" vor Funktionsaufruf in der Applikation korrigieren.

`#define ERR_S5_ADR 0xFF08` // Adresse zu groß

Bei Einzel- oder Blockelementzugriff mit Elementtyp ABS_SNG wurde versucht auf eine Adresse > FFFFh in einem AG der 115U-Reihe zuzugreifen. Die CPUs (bis CPU 944) haben jedoch nur 64 KByte Adreßraum.

Abhilfe: Strukturvariable "Adresse" vor Funktionsaufruf in der Applikation korrigieren.

`#define ERR_S5_QVZ 0xFF09` // QVZ/ADF im AG bei lesen/schreiben

Es wurde versucht auf einen Adreßbereich zuzugreifen, der physikalisch nicht vorhanden ist. Die AGs der 135 und 155-Reihe stellen diese Fehlermeldungen zu Verfügung. Ein AG der 115U-Reihe würde hierbei in den Stop Zustand versetzt werden.

Abhilfe: Strukturvariable "Elementtyp" bzw. "Adresse" vor Funktionsaufruf in der Applikation korrigieren.

`#define ERR_S5_944 0xFF0A` // CPU 944: Baustein im Prog.Bank

Bei Blockelementzugriff mit Elementtyp "Baustein"_BLK wurde versucht auf einen Baustein zuzugreifen, der nicht in der DB-Bank liegt (Betrifft nur CPU 944 vom AG-Typ 115U).

Abhilfe: Baustein im AG in DB-Bank anlegen (über BIB-Nr. 19285) oder Funktionsaufruf in der Applikation korrigieren.

#define ERR_S52SHRT 0xFF0B // Bereich im AG zu klein

#define ERR_S5_BITSIZE 0xFF0C // Übertragungslänge bei Bitelementen nicht 1

3.3.5 Beispielprogramm

Das folgende Beispiel zeigt die Verwendung der oben beschriebenen Funktionen. Die Information zum Kacheltreiber wird zuerst ausgelesen und angezeigt. Danach werden die Strukturelemente zum Lesen aus der Kachel bzw. zum Schreiben in die Kachel initialisiert. Es sollen 10 Datenwörter aus dem Datenbaustein in DB10 ab Datenwort 0 gelesen bzw. geschrieben werden. Vor dem Schreiben wird das erste Byte der Daten geändert.

```
#include <windows.h>
#include <cp486def.h>
#include <stdio.h>
#include <stdlib.h>

void main(void)
{
  unsigned short *db;
  KACHEL_STRUCT   ks;
  CP386_PARAMETER *cp;

  if(!CP_init())                // CP-Funktionen initialisieren
  {
    ks.Cb = sizeof(KACHEL_STRUCT);

    if(!CP_getkachel(&ks))      // Kachelinfo lesen
    {
      printf("Treiberversion %d, Kachelstart %d,Anzahl %d, Seriennummer %d\n",
        ks.TreiberVersion,ks.Kachelbasis,ks.KachelAnzahl,ks.Seriennummer);
    }
    else
      printf("Kein Kacheltreiber");

    if(cp=(CP386_PARAMETER*)malloc(sizeof(CP386_PARAMETER*) +1024))
    {                                  // Cp-Struktur initialisieren
      cp->Elementtyp=DB_BLK;           // Datenbaustein-Datenblock
      cp->Bausteinnummer=10;           // Bausteinnummer 10
      cp->Kennung=W_BLOCK;             // Wort-Block
      cp->Adresse=0;                   // ab Offset 0 im DB
      cp->Len=10;                      // 10 Wörter lesen
      cp->Cpu=0;                       // Lesen starten

      db=(unsigned short*)cp->Daten;

      if(!CP_startread(cp))        // Lesen starten ->  0 OK,  != 0 Systemfehler
      {
        printf("Read gestartet\n");
        if(cp->Fehler==0) //cp->Fehler wird durch Start Funktion 0-initialisiert
        {
          printf("Handlenummer: %d\n",cp->Handle);

          while(!CP_pollread(cp))      //  0 OK,  != 0 Systemfehler
          {
            if(cp->Fehler==REQ_NO_ERR) // cp->Fehler =  REQ_NO_ERR wenn Daten da
            {
              printf("Daten gelesen : %d %d %d\n",db[0],db[1],db[2]);
              break;
            }
```

```
            else
            {                  // REQ_WRKN, REQ_UNDEF oder SPS-spezifischer Fehler
              printf("CpFehler = %d\n",cp->Fehler);
              Sleep(20);       // 20 ms warten dann nochmal probieren
            }
          }
        }
        else
          printf("CpFehler = %d\n",cp->Fehler);  // allgemeiner Fehler (100-110)
      }
      else
        printf("Kein Startread\n");

      (*db)++;                // Datum[0] inkrementieren

      if(!CP_startwrite(cp))  // Schreiben starten ->  0 OK,  != 0 Systemfehler
      {              // cp->Fehler wird durch die Start Funktion 0-initialisiert
        printf("Write gestartet\n ");
        if(cp->Fehler==0)
        {
          printf("Handlenummer %d \n",cp->Handle);
          while(!CP_pollwrite(cp))     //  0 OK,  != 0 Systemfehler
          {
            if(cp->Fehler == REQ_NO_ERR)
            {
              printf("Daten geschrieben! \n");
              break;
            }
            else
              // REQ_WRKN, REQ_UNDEF oder SPS-spezifischer Fehler
              printf("CpFehler = %d\n",cp->Fehler);
          }
        }
        else
          printf("CpFehler = %d\n",cp->Fehler);  // allgemeiner Fehler (100-110)
      }
      else
        printf("Kein Startwrite\n");

       CP_exit();
    }
    else
      printf("Kein CP486\n");
  }
  else
    printf("Kein Init");
}
```

4 MS-DOS-Utilities für Silicondisk-Betrieb

4 MS-DOS-Utilities für Silicondisk-Betrieb

4.1 Silicondisk-Treiber

Für die Standard-Hardware-Komponenten sind im BIOS die Treiberfunktionen zur Bedienung und Benutzung dieser Komponenten enthalten. Zusätzliche Hardware oder modifizierte Hardware muß ebenfalls durch Treiber bedient werden. Damit nicht bei jeder kleineren Systemänderung ein neues BIOS erstellt werden muß, bietet MS-DOS die Möglichkeit an, Treiber auch noch nachträglich an das BIOS anzuhängen. Dies geschieht durch einen Eintrag in der Datei CONFIG.SYS.

Auf der CP3/4 stehen verschiedene Silicondisks zur Verfügung

- Chip-Silicondisk (IC3, IC4):
  Die Chip-Silicondisk hat eine Kapazität von 256KB bzw. 1MB. Die Chip-Silicondisk ist bestückbar mit SRAM, EPROM, und PEROM.
- Memorycard-Silicondisk
  Memorycards haben mit einer Kapazität von 128KB, 512KB und 1MB. Es stehen OTP-ROM- und SRAM-Memorycards zur Verfügung (bis Leiterplatten-Version V17).
- Silicondisk-Zusatzboard
  Die verschiedenen Zusatzboards haben eine Kapazität von bis zu 7MB und sind mit FLASH-, SRAM- und EPROM-Bestückung erhältlich.

Damit die Silicondisks jeweils als Laufwerk angesprochen werden können, muß der entsprechende Treiber in der Systemkonfigurationsdatei CONFIG.SYS eingetragen werden.
Es stehen verschiedene Treiber zur Verfügung:

SDRAM.SYS	Treiber für SRAM-Silicondisk Mit diesem Treiber können Silicondisks gelesen und beschrieben werden.
SDROM.SYS	Treiber für EPROM-, FLASH-PROM- und OTP-ROM-Silicondisks Mit diesem Treiber können Silicondisks nur gelesen werden.
SDPEROM.SYS	Treiber für Silicondisks mit EEPROM-Bausteinen Diese Silicondisk kann gelesen und beschrieben werden. Die Lebensdauer von EEPROMs hängt jedoch stark von der Anzahl der Schreibzyklen ab. (1.000 bis 10.000 je nach Typ) Es wird deshalb empfohlen, diesen Treiber nur kurzzeitig zu installieren, um die EEPROMs mit Daten zu füllen, oder die Daten zu modifizieren. Anschließend sollte der Treiber SDROM.SYS verwendet werden, um Schreibzugriffe zu verhindern.

Der entsprechende Treiber wird durch folgende Anweisung in der Datei CONFIG.SYS installiert:

```
device=[d:][Pfad]SDxxx.SYS[Bemerkung][/bb][Bemerkung]
       [/llll][Bemerkung][/pp]
```

Funktion der einzelnen Parameter

`[c:][Pfad]` Angabe des Laufwerks und Pfads mit dem Treiber SDxxx.SYS, wobei xxx den verwendeten Speichertyp für die Silicondisk angibt. Zur Verfügung stehen die drei beschriebenen Treiber SDRAM, SDROM und SDPEROM.

`[Bemerkung]` Beliebiger Text außer dem Schrägstrich "/" und Eingabetaste.

`/bb` Basisadresse der Silicondisk (Schrittweite = 64 KB)
(z.B. /C0 entspricht der physikalischen Startadresse C00000h).

`/llll` Größe der Silicondisk (Schrittweite = 1 KB)
(z.B. /256 entspricht einer Diskgröße von 256 KB = 262144 Byte).

`/pp` Parameter für SDPEROM.SYS: Blockgröße der PEROMs
64 Byte bei AT29MC010-Baustein
128 Byte bei AT29C010- und bei AT29MC040-Baustein
256 Byte bei AT29MC040-Baustein

Beispiele:

Chip-Silicondisk mit 1MB EPROM

```
DEVICE = \DEVICE\sdrom.sys Basisadresse=/c0 Grösse=/1024
```

Erklärung Der Treiber befindet sich im Unterverzeichnis "DEVICE" auf dem Bootlaufwerk. Die Silicondisk besteht aus einem Nur-Lese-Speicher. Deshalb wird der Treiber SDROM.SYS gewählt. Die Basisadresse der Silicondisk liegt bei C00000h. Die Silicondisk hat eine Größe von 1024KB = 1 MByte.

Chip-Silicondisk mit 256KB SRAM

```
DEVICE = \DEVICE\sdram.sys Basisadresse=/c0 Grösse=/256
```

Erklärung Der Treiber befindet sich im Unterverzeichnis "DEVICE" auf dem Bootlaufwerk. Die Silicondisk besteht aus einem beschreibbaren Speicher. Deshalb wird der Treiber SDRAM.SYS gewählt. Die Basisadresse der Silicondisk liegt bei C00000h. Die Silicondisk hat eine Größe von 256KB.

Memorycard-Silicondisk mit 1MB ROM:

```
DEVICE = \DEVICE\sdrom.sys Basisadresse=/80 Grösse=/1024
```

Erklärung Der Treiber befindet sich im Unterverzeichnis "DEVICE" auf dem Bootlaufwerk. Die Silicondisk besteht aus einem Nur-Lese-Speicher. Deshalb wird der Treiber SDROM.SYS gewählt. Die Basisadresse des Memorycard-Steckplatzes ist 800000h. Die Silicondisk hat eine Größe von 1024KB = 1MB.

Memorycard-Silicondisk mit 128KB SRAM:

```
DEVICE = \DEVICE\sdram.sys Basisadresse=/80 Grösse=/128
```

Erklärung Der Treiber befindet sich im Unterverzeichnis "DEVICE" auf dem Bootlaufwerk. Die Silicondisk besteht aus einem beschreibbaren Speicher. Deshalb wird der Treiber SDRAM.SYS gewählt. Die Basisadresse des Memorycard-Steckplatzes ist 800000h. Die Silicondisk hat eine Größe von 128KB.

Hinweis

Falls die Silicondisk bereits im VIPA-SETUP aktiviert wurde, ist eine Anmeldung über die Datei CONFIG.SYS nicht mehr notwendig. In diesem Fall wurde dieser Disk bereits die Laufwerksbezeichnung A: zugeteilt. Ein zusätzlicher Eintrag im File CONFIG.SYS würde eine zweite Laufwerksbezeichnung zuordnen.

Es können mehrere Silicondisks installiert werden, indem die Anweisung in der Datei CONFIG.SYS mit entsprechend geänderten Parametern wiederholt wird. So kann auch ein physikalisch zusammenhängender Adreßbereich in mehrere logische Silicondisk-Laufwerke unterteilt werden.

Vom Betriebssystem werden alle Treiber in der Datei CONFIG.SYS installiert. Dabei wird jedem Speichermedium eine Laufwerksbezeichnung zugeteilt (z.B. D: oder E: usw.). Diese Laufwerksbezeichnung wird im Anlauf des Systems am Bildschirm als Meldung ausgegeben.

Durch jedes virtuelle Laufwerk wird der residente Teil von DOS um etwa 900 Byte für den Treiber erweitert.

SRAM-Silicondisks müssen mit dem Programm FORMATSD initialisiert werden. Wenn die RAM-Disk batterie- bzw. akkugepuffert ist, muß dies nur einmal geschehen, andernfalls nach jedem Einschalten des Systems. Auf gepufferten RAM-Disks bleiben die Daten auch nach Abschalten des Systems für eine gewisse Zeit erhalten.

Gepufferte RAM-Disks sind bootbar, falls auf der Disk ein Betriebssystem installiert wurde.

4.2 Formatierprogramm für SRAM-Silicondisk

Ähnlich wie eine Diskette müssen auch SRAM-Silicondisks formatiert werden bevor sie unter DOS als Speichermedium nutzbar sind. Zum Formatieren der Silicondisk dient das Programm FORMATSD.EXE. Das Programm wird unter MS-DOS folgendermaßen aufgerufen:

```
[c:][Pfad]FORMATSD d:[/D:xx][/S]
```

Die einzelnen Parameter haben folgende Funktion

`[c:][Pfad]`	Angabe des Laufwerks und Pfads mit der FORMATSD-Befehlsdatei
`d:`	Angabe des Silicondisk Laufwerks, das formatiert werden soll
`/D:xx`	mit diesem Parameter kann angegeben werden, wieviel Platz für Directory-Einträge reserviert werden soll. xx kann Werte zwischen 1 und 99 annehmen. Falls dieser Parameter fehlt, wird ein Platz für 64 Directory-Einträge reserviert.
`/S`	durch diesen Parameter werden die Betriebssystemdateien vom MS-DOS-Bootlaufwerk auf die neue Silicondisk kopiert. Dies ist erforderlich, falls die neue Silicondisk bootfähig sein soll.

Dieses Programm löscht alle Daten auf dem spezifizierten Laufwerk!

Das Programm FORMATSD muß ausgeführt werden, bevor mit anderen Programmen oder Systembefehlen (z.B. DIR) auf das zu formatierende Silicondisklaufwerk zugegriffen wird. Wird dies nicht beachtet, so kann es vorkommen, daß das Silicondisklaufwerk nicht mit der gewünschten Größe formatiert wird.

4.3 Silicondisk-Generator

Mit dem Programm SDGEN werden die Binärdateien für die EPROMs, FLASH-PROMs und PEROMs des Programmspeichers erstellt. Das Programm wird folgendermaßen aufgerufen:

```
[c:][path] SDGEN
```

Das Programm fragt noch nach folgenden Parametern:

EPROM-Größe in Bit (0, 512 ,1M, 2M, 4M, 8M):

Es muß die Größe der EPROMs angegeben werden (z.B. 1M beim EPROM 27C010). Es werden entsprechend große Dateien für ein EPROM-Programmiergerät erstellt. Bei Eingabe von 0 wird eine einzige Datei erstellt wie sie von SDLOAD benötigt wird.

gesplittet (J/N) Gesplittete Dateien werden bei 16 Bit breiten Silicondisks benötigt. Die Silicondisk auf der CP3/4 PC ist 16 Bit breit, die Memorycard ist 8Bit breit. In der 16Bit breiten Silicondisk liegen immer 2 EPROMs (odd und even Byte) parallel.

Quellaufwerk erstelltes Mutterlaufwerk für die Silicondisk

Zieldateiname Dateiname für die Binärfiles für die EPROM-Programmierung.

Entsprechend der EPROM-Größe werden mehrere Binärdateien unter dem eingegebenen Zieldateinamen erzeugt. Die Extension der einzelnen Dateien wird mit einer fortlaufenden Nummer versehen (Zieldateiname.xxx). Bei gesplitteten Dateien werden ODD- und EVEN-Dateien durch ein O bzw. ein E im ersten Buchstaben der Extension markiert (Zieldateiname.Oxx bzw. Zieldateiname.Exx).

Hinweis

Für die Erstellung einer bootbaren Silicondisk mit Hilfe der MS-DOS-Ramdisk RAMDRIVE.SYS, ist zunächst mit dem Systemprogramm LABEL das Label von dieser RAM-Disk zu entfernen. Kopieren Sie die Betriebssystemdateien (IO.SYS, MSDOS.SYS und COMMAND.COM) auf die leere RAM-Disk. Nun können Sie mit SDGEN Ihre übrigen Dateien übertragen. Anwenders, auf kopiert werden.

Zur Übertragung der Daten mit SDLOAD, ist zunächst mit SDGEN eine große Datei zu erzeugen, die Sie dann mit SDLOAD laden können.

Für die direkte Übertragung einer MS-DOS-Ramdisk mit SDLOAD ist immer eine Zwischendatei erforderlich.

4.4 Silicondisk-Loader

Das Programm SDLOAD.EXE dient zum Laden der Silicondisk mit den vorbereiteten Datensätzen. Dieser Lader muß bei FLASH-PROMs und bei PEROMs verwendet werden. FLASH-PROMs können nur komplett gelöscht und mit diesem Programm auch wieder komplett beschrieben werden. Bei PEROMs ist die Anzahl der Schreibzyklen sehr eingeschränkt. Wenn Sie die Dateien mit einem Standard DOS-Kopierprogramm übertragen, würden bestimmte Sektoren, vor allem im Directory-Bereich, sehr oft neu beschrieben, dies würde die Lebensdauer der Bauteile sehr reduzieren. Auch in diesem Fall ist ein kompletter Datentransfer, bei dem jeder Sektor nur einmal beschrieben wird, sinnvoll.

Das Programm SDLOAD wird unter MS-DOS folgendermaßen aufgerufen:

```
[c:][Pfad]SDLOAD
```

wobei unter `[c:][Pfad]` Laufwerk und Pfad des SDLOAD-Programms liegen.

Nach dem Programmaufruf erscheint folgende Liste von Bauteilen auf dem Bildschirm:

```
Es können folgende FLASH-PROMs bzw. EEPROMs/PEROMs programmiert werden:
        1  Am28F010-150,P28F010-150
        2  Am28F020-150,P28F020-150
        3  Am28F040-150,P28F040-150
        4  AT28C010-150
        5  AT28C040-150
        6  AT29C010-150
        7  AT29C040-150
        8  AT29MC010-150
        9  AT29MC040-150
Geben Sie bitte die Nummer des von Ihnen verwendeten ROM-Typs an:
```

Die eingesetzten Bauteile müssen durch Eingabe der Nummer und Betätigung der Eingabetaste ausgewählt werden.

Anschließend wird noch nach der Anzahl der eingesetzten Bauteile gefragt:

```
Bitte geben Sie die Anzahl der oben ausgewählten Bausteine an (2, 4, 6 oder 8):
```

Es muß die entsprechende Stückzahl eingeben und die Eingabetaste betätigt werden.

Danach folgt die Frage nach der Basisadresse auf der das Silicondiskboard eingesetzt wird. Diese Adresse wird als Hexadezimal-Adresse eingegeben:

Mögliche Eingabewerte sind:
800000
840000
880000
...
FC0000

In die Silicon-Disk kann nun entweder der Inhalt einer Datei, die mit SDGEN erzeugt wurde oder der Inhalt eines logischen Laufwerks übertragen werden. Falls der Inhalt einer Datei übertragen werden soll, muß der Dateiname eingegeben werden. Soll der Inhalt eines Laufwerks übertragen werden, wird nur die Laufwerksbezeichnung angegeben.

Dateinamen werden in folgendem Format angegeben: `[d:][Pfad]Dateiname`
Laufwerksbezeichnung werden im folgenden Format angegeben: `d:`

Bei Angabe einer Datei muß diese mit SDGEN so erzeugt worden sein, daß der komplette Laufwerksinhalt in einer einzigen Datei abgelegt ist. Die Datei hat dann z.B. den Namen: SDISK.000

Das zu programmierende Laufwerk muß eine gleich große oder größere Kapazität haben als die Ausgangsdatei oder das Ausgangslaufwerk.

SDLOAD überträgt das Laufwerk oder den angegebenen File ohne Änderungen. Ein Laufwerk, das nicht bootbar ist wird auch beim Übertragen durch SDLOAD nicht bootbar. In diesem Fall muß erst ein File mit SDGEN angelegt werden.

4.5 Applikationsbeispiele für die Benutzung der Silicondisk

4.5.1 SRAM-Disk erstellen

Ziel Auf der Basisadresse C00000h soll eine SRAM-Silicondisk mit einer Größe von 256KB (bestehend aus 2 Stück 128KB SRAMs (1MBit SRAMs)) erstellt werden.

Das Basisboard wird mit 2 SRAMs von je 128KB-Chips bestückt und die Jumper entsprechend eingestellt.

Das System wird von Festplatte (Laufwerk C:) gebootet. Für die Erstellung werden die Programme SDRAM.SYS, FORMATSD.EXE (im Unterverzeichnis C:\SD) und ein Texteditor benötigt.

Mit dem Texteditor werden folgende Zeilen in die Datei C:\CONFIG.SYS am Ende eingefügt:

```
DEVICE = C:\SD\SDRAM.SYS  Basis=/C0 Grösse=/256
```

Durch gleichzeitiges Betätigen der Tasten <STRG>, <ALT> und <ENTF> wird das System neu gestartet. Beim Hochlaufen liefert das System u.a. folgende Meldungen:

SILICON DISK als Laufwerk D: installiert. Vx.x Datum RAMDISK AB ADRESSE C00000h

Damit ist das Laufwerk vorhanden und muß noch durch folgenden Aufruf formatiert werden:

```
C:\SD\FORMATSD D:  /D:32  /S
```

Das Programm FORMATSD.EXE legt nun auf Laufwerk D: (SRAM-Disk) eine DOS-Struktur an. Es wird Platz für 32 Hauptverzeichniseinträge reserviert. Wenn das Programm den Datenspeicher ohne Fehlermeldung formatiert hat, kann dieser als MS-DOS-Laufwerk angesprochen werden. Durch den Parameter /S werden die Systemdateien auf die SRAM-Disk übertragen, so daß die SRAM-Disk als Bootlaufwerk benutzt werden kann.

4.5.2 FLASH-PROM-Silicondisk erstellen

Ziel Auf der Basisadresse C00000h soll eine FLASH-PROM-Silicondisk mit einer Größe von 1MByte (bestehend aus 4 Stück 256KB FLASH-PROMs (2MBit FLASH-PROMS) erstellt werden.

Es wird ein Silicondiskboard mit 4 Stück 256KB FLASH-PROMs (2MBit FLASH-PROMs) und 2 Stück 512KB SRAMs (4MBit SRAMs) verwendet. Die Basisadresse der FLASH-PROM-Disk wird auf C00000h und auf eine Länge von 1MB eingestellt. Die Basisadresse der SRAM-Disk wird auf 800000 h und ebenfalls auf eine Länge von 1MB eingestellt.

Das System wird von Festplatte (Laufwerk C:) gebootet. Für die Erstellung werden die Programme SDROM.SYS, SDRAM.SYS, FORMATSD.EXE und SDLOAD.EXE (im Unterverzeichnis C:\SD) und ein Texteditor benötigt.

Mit dem Texteditor werden folgende Zeilen in die Datei C:\CONFIG.SYS am Ende eingefügt:

```
DEVICE = C:\SD\SDRAM.SYS Basis=/80 Grösse=/1024
DEVICE = C:\SD\SDROM.SYS Basis=/C0 Grösse=/1024
```

Durch gleichzeitiges Betätigen der Tasten <STRG>, <ALT> und <ENTF> wird das System neu gestartet. Beim Hochlauf liefert das System u.a. folgende Meldungen:

```
SILICON DISK als Laufwerk D: installiert.  Vx.y  Datum
RAMDISK AB ADRESSE 80 0000H

SILICON DISK als Laufwerk E: installiert.  Vx.y  Datum
ROMDISK AB ADRESSE C0 0000H
```

In Laufwerk D: (SRAM-Disk) wird eine Vorlage für den Programmspeicher erstellt. Dazu muß die SRAM-Disk zuerst mit dem Kommando

```
SD\FORMATSD D: /D:32 /S
```

formatiert werden.

Anschließend werden alle auf der FLASH-PROM-Disk erforderlichen Dateien auf dieses Laufwerk D: kopiert. Damit ist die Vorlage für die FLASH-PROM-Disk fertig erstellt und kann getestet werden, dazu muß im Setup die Silicondisk auf Basisadresse 80h 0000h ausgewählt werden.

Zum Übertragen dieser Vorlage auf die FLASH-PROM-Disk wird das Programm SDLOAD aufgerufen und folgende Parameter eingegeben:

```
C:\SD\SDLOAD
```

```
Es können folgende FLASH-PROMs bzw. EEPROMs/PEROMs programmiert werden:
        1  Am28F010-150,P28F010-150
        2  Am28F020-150,P28F020-150
        3  Am28F040-150,P28F040-150
        4  AT28C010-150
        5  AT28C040-150
        6  AT29C010-150
        7  AT29C040-150
        8  AT29MC010-150
        9  AT29MC040-150
Geben Sie bitte die Nummer des von Ihnen verwendeten ROM-Typs an: 2
Bitte geben Sie die Anzahl der oben angegebenen Bausteine an (2,4,6,8): 4

Bitte geben Sie an, welche Basisadresse Sie auf dem Silicon-Disk Board
eingestellt haben. Eingabe als Hexadezimalwert: C00000

In die Silicon-Disk kann entweder der Inhalt einer Datei, die mit SDGEN erzeugt
wurde, oder der Inhalt eines logischen Laufwerks übertragen werden.
Geben Sie den Dateinamen oder die Laufwerksbezeichnung an: D:
```

In die FLASH-PROMs wird nun der Inhalt des Laufwerks D: übertragen. Wenn das Programm ohne Fehler beendet wurde, kann die FLASH-PROM-Disk als Laufwerk E: angesprochen werden. Die Inhalte von Laufwerk D: und E: sind gleich. Laufwerk E: ist jedoch schreibgeschützt. Wenn im BIOS-Setup als Laufwerk A: die Silicondisk auf Adresse C0h 0000h eingestellt wird, kann diese Silicondisk auch zum Booten verwendet werden.

4.5.3 Programmspeicher mit EPROMs erstellen

Ziel Es soll eine EPROM-Silicondisk aus 2 Stück 512KB EPROMs erstellt werden.

Variante 1 über Diskette

Zu diesem Zweck wird eine Diskette benötigt, die formatiert und bootfähig ist (format a: /s). Die Formatierung mit Systemübertragung sollte von einem Rechner aus geschehen, auf dem dieselbe Version des Betriebssystems läuft, die später auf der Silicondisk laufen soll. Danach sollten Sie alle Dateien, wie sie später auf der Silicondisk liegen sollen, auf die Diskette übertragen. Unter Verwendung von einer 1,44MB Diskette sollten Sie nur diejenigen Dateien übertragen, die tatsächlich auf der Silicondisk benötigt werden. Aus dieser Vorlage können jetzt die Dateien für die 2 EPROMs erstellt werden. Hierzu wird auf Ihrem Entwicklungsrechner das Programm SDGEN aufgerufen und es werden unter der Voraussetzung, daß sich Ihre Silicondisk-Diskette in Laufwerk A: befindet, folgende Parameter eingegeben:

```
C:\SD\SDGEN
EPROM-GRÖSSE in Bit (0,512,1M,2M,4M,8M)        : 4M
GESPLITTET IN ODD-EVEN (J/N)                   : J
QUELL-LAUFWERK (A: .... F:)                    : A:
ZIEL-DATEINAME (max. 8 Zeichen)                : EPROM
```

Das Programm erzeugt die Dateien EPROM.O00 und EPROM.E00. Es handelt sich dabei um Binärdateien für das EPROM-Programmiergerät, wobei jede Datei zu einem EPROM gehört. Diese EPROMs werden programmiert und anschließend werden die EPROMs folgendermaßen in die Silicondisksockel eingesetzt:

bis V32:	`EPROM.E00 in IC4 (even)`
	`EPROM.O00 in IC3 (odd)`
ab V33:	`EPROM.E00 in IC2 (even)`
	`EPROM.O00 in IC1 (odd)`

Wenn nun im BIOS-Setup als Laufwerk A: die Silicondisk ROM auf Adresse C00000h eingestellt wird, kann die Silicondisk zum Booten verwendet werden. Nach dem Einschalten und Booten steht das erstellte Speichermedium als Laufwerk A: zur Verfügung.

Hinweis

Zur Erzeugung eines Silicondisk-Updates gehen Sie folgendermaßen vor:

Nach der Übertragung der neuen Dateien auf die bereits bestehende Diskette, ist die Diskette zu defragmentieren (Defrag aufrufen). Bei einer Defragmentierung werden Dateien in zusammenhängender Reihenfolge auf der Diskette abgelegt. Zwischen Datenblöcken befindliche Leerräume werden überschrieben. Die zusammenhängenden Datensätze werden byteweise auf die Silicondisk übertragen. So ist gewährleistet, daß die Daten möglichst wenig Speicherplatz auf der Silicondisk belegen und bei Erreichen der 1MB Grenze auf der Silicondisk der Speicher völlig benutzt wird.

Variante 2 über SRAM-Disk

Installieren Sie auf dem Basisboard eine SRAM-Disk mit 2 Stück 512KB SRAMs (4MBit SRAMs) und setzen Sie die Jumper entsprechend. Stellen Sie die Basisadresse C00000h ein.

Das System wird von Festplatte (Laufwerk C:) gebootet. Für die Erstellung werden die Programme SDROM.SYS, SDRAM.SYS, FORMATSD.EXE und SDGEN.EXE (im Unterverzeichnis C:\SD) und ein Texteditor benötigt. Mit dem Texteditor wird folgende Zeile in die Datei C:\CONFIG.SYS am Ende eingefügt:

```
DEVICE = SD\SDRAM.SYS Basis=/C0 Grösse=/1024
```

Durch gleichzeitiges Betätigen der Tasten <STRG>, <ALT> und <ENTF> wird das System neu gestartet. Beim Hochlauf liefert das System u.a. folgende Meldungen:

```
SILICON DISK als Laufwerk D: installiert.  Vx.y  Datum
RAMDISK AB ADRESSE C00000h
```

In Laufwerk D: (SRAM-Disk) wird eine Vorlage für den Programmspeicher erstellt. Hierzu ist die SRAM-Disk zuerst mit

```
SD\FORMATSD D: /D:32 /S
```

zu formatieren.

Kopieren Sie nun die erforderlichen Daten das Laufwerk D. Damit ist die Vorlage für die EPROM-Disk fertig erstellt und kann getestet werden. Wählen Sie hierzu im Setup die Silicondisk ROM auf Adresse C00000 aus und löschen Sie in der CONFIG.SYS folgende Zeile:

```
DEVICE = SD\SDRAM.SYS Basis=/C0 Grösse=/1024
```

Aus dieser Vorlage können Sie nun die Dateien für die 2 EPROMs erstellen. Rufen Sie das Programm SDGEN auf und geben Sie folgende Parameter ein:

```
C:\SD\SDGEN
EPROM-GRÖSSE in Bit (0,512,1M,2M,4M,8M): 4M
GESPLITTET IN ODD-EVEN             (J/N): J
QUELL-LAUFWERK (A: .... F:)             : D:
ZIEL-DATEINAME (max. 8 Zeichen)         : EPROM
```

Das Programm erzeugt die Dateien EPROM.O00 und EPROM.E00. Es handelt sich hierbei um Binärdateien für das EPROM-Programmiergerät, wobei jede Datei zu einem EPROM gehört. Diese EPROMs werden programmiert und anschließend folgendermaßen in die Silicondisksockel eingesetzt:

bis V32:	EPROM.E00 in IC4 (even)
	EPROM.O00 in IC3 (odd)
ab V33:	EPROM.E00 in IC2 (even)
	EPROM.O00 in IC1 (odd)

Wenn nun im BIOS-Setup als Laufwerk A: die Silicondisk ROM auf Adresse C00000h eingestellt wird, kann das Silicondisk EPROM zum Booten verwendet werden.

Nach dem Einschalten und Booten steht das erstellte Speichermedium als Laufwerk E: zur Verfügung.

4.5.4 FLASH-PROM-Silicondisk mit MS-DOS-RAM-Disk erstellen

Ziel Auf dem Silicondiskboard soll ab der Basisadresse C00000h eine FLASH-PROM-Silicondisk mit einer Größe von 2MByte (bestehend aus 8 Stück 256KB FLASH-PROMs (2MBit FLASH-PROMs)) erstellt werden.

Es wird ein Silicondiskboard mit 8 Stück 256KB FLASH-PROMs (2MBit FLASH-PROMs) verwendet. Die Basisadresse wird auf C00000 h und auf eine Länge von 2MB eingestellt.

Das System wird von Festplatte (Laufwerk C:) gebootet. Für die Erstellung werden die Programme SDROM.SYS, SDGEN.EXE und SDLOAD.EXE (im Unterverzeichnis C:\SD) und ein Texteditor benötigt.

Mit dem Texteditor werden folgende Zeilen in die Datei C:\CONFIG.SYS am Ende eingefügt:

```
DEVICE = C:\DOS\RAMDRIVE.SYS 2042 512 64 /E
DEVICE = C:\SD\SDROM.SYS Basis=/C0 Grösse=/2048
```

Durch gleichzeitiges Betätigen der Tasten <STRG>, <ALT> und <ENTF> wird das System neu gestartet. Beim Hochlaufen liefert das System u.a. folgende Meldungen:

Microsoft RAMDrive Version x.y - Virtuelles Laufwerk D:

Plattengröße: 2042K

Sektorgröße: 512 Byte

Blockgröße: 1 Sektor

Verzeichniseinträge: 64

ROM SILICON DISK als Laufwerk E: installiert. Vx.y Datum

Das Laufwerk D: wird als Vorlage für die ROM-Silicondisk verwendet. Dazu muß als erstes das Label entfernt werden, da dieses einen Platz belegt, der vom Betriebssystem benötigt wird. Das Kommando dafür lautet:

```
LABEL D:
```

Dann erscheint am Bildschirm die Meldung:

```
Datenträger in Laufwerk D ist MS-RAMDRIVE
Datenträgerbezeichnung(11 Zeichen, EINGABETASTE für keine)?
```

Es wird keine Datenträgerbezeichnung eingegeben, sondern nur die Eingabetaste betätigt. Daraufhin erscheint die Frage:

```
Die aktuelle Datenträgerbezeichnung löschen (J/N)?
```

Diese Frage wird mit "J" und der Eingabetaste bestätigt.

Danach ist das Programm LABEL abgeschlossen (MS-DOS-Prompt). Als nächstes werden alle benötigten Dateien auf das Laufwerk D: kopiert (folgende Reihenfolge beachten falls das Laufwerk bootfähig werden soll):

1. Io.sys
2. Msdos.sys
3. command.com
4. die restlichen Dateien und Verzeichnisse in beliebiger Reihenfolge

Die Dateien Io.sys und Msdos.sys sind versteckte Dateien und können deshalb nicht mit dem DOS-Befehl COPY übertragen werden.

Übertragungsmöglichkeiten bestehen z.B. mit Hilfe der DOSSHELL oder dem Norton Commander. Damit ist die Vorlage für den Programmspeicher erstellt und als nächstes wird das Programm SDGEN mit folgenden Parametern aufgerufen:

```
C:\SD\SDGEN

EPROM-GRÖSSE in Bit (0,512,1M,2M,4M,8M): 0
GESPLITTET IN ODD-EVEN            (J/N): N
QUELL-LAUFWERK (A: .... F:)            : D:
ZIEL-DATEINAME (max. 8 Zeichen)        : EPROM
```

Das Programm erzeugt die Datei EPROM.000. (Anmerkung für Insider: Bei der Erstellung dieser Datei wird durch SDGEN der Bootsektor und die doppelte FAT erzeugt. Damit wird die Silicondisk bootfähig.). Diese Datei kann anschließend mit dem Programm SDLOAD in den Programmspeicher übertragen werden. Das Programm SDLOAD wird mit folgenden Parametern aufgerufen:

```
C:\SD\SDLOAD

    Es können folgende FLASH-PROMs bzw. EEPROMs/PEROMs programmiert werden:
            1  Am28F010-150,P28F010-150
            2  Am28F020-150,P28F020-150
            3  Am28F040-150,P28F040-150
            4  AT28C010-150
            5  AT28C040-150
            6  AT29C010-150
            7  AT29C040-150
            8  AT29MC010-150
            9  AT29MC040-150
    Geben Sie bitte die Nummer des von Ihnen verwendeten ROM-Typs an: 2

    Bitte geben Sie die Anzahl der oben angegebenen Bausteine an (2,4,6,8): 8

    Bitte geben Sie an, welche Basisadresse Sie auf dem Silicon-Disk Board
    eingestellt haben. Eingabe als Hexadezimalwert: C00000

    In die Silicon-Disk kann entweder der Inhalt einer Datei, die mit SDGEN
    erzeugt wurde, oder der Inhalt eines logischen Laufwerks übertragen werden.
    Geben Sie den Dateinamen oder die Laufwerksbezeichnung an: C:\EPROM.000
```

In die FLASH-PROMs wird nun der Inhalt der Datei C:\EPROM.000 übertragen. Wenn das Programm ohne Fehler beendet wurde, kann die FLASH-PROM-Disk als Laufwerk E: angesprochen werden. Die Inhalte von Laufwerk D: und E: sind gleich. Laufwerk E: ist jedoch schreibgeschützt. Wenn im BIOS-Setup die Silicondisk auf Adresse C00000h als Laufwerk A: eingestellt wird, kann dieser Programmspeicher auch zum Booten verwendet werden.

5 Hilfsprogramme

5 Hilfsprogramme

5.1 Programm CPLINK zur Rechnerkopplung

5.1.1 Allgemeines

Mit diesem Programm können Dateien über eine serielle Schnittstelle in die CP3/4 geladen oder aus der CP3/4 ausgelesen werden. Auf diese Weise können auch CP3/4-Baugruppen ohne Diskettenlaufwerk oder ohne wechselbare Memorycard mit Daten und Programmen versorgt werden oder aufgezeichnete Daten können ausgelesen werden.

Anwählbare Programmfunktionen finden Sie in der Statuszeile (unterste Zeile). Farblich zurückgesetzte Befehle sind nicht ausführbar. Weitere Tastenkürzel sind dem Menüsystem zu entnehmen.

Die Menüzeile (oberste Zeile) wird mit <F10> angewählt. Von hier aus kann man mit Hilfe der Cursortasten und der Eingabe-Taste den gewünschten Menüpunkt anwählen. Schneller ist dies möglich, indem man einen Menüpunkt direkt mit der Tastenkombination <Alt+Kennbuchstabe> selektiert. Der Kennbuchstabe ist hierbei der hervorgehobene (großgeschriebene) Buchstabe. Daraufhin klappt ein Untermenü auf, in dem man mit Kennbuchstabe den gewünschten Befehl aufrufen kann.

Die Dateilistenfenster dienen der Anzeige und Bearbeitung von Laufwerksverzeichnissen. Ist die Verbindung zwischen zwei Rechnern aufgebaut, kann mit <Tab> zwischen beiden Fenstern umgeschaltet werde. Das aktive Fenster ist durch einen doppelten Rahmen gekennzeichnet. In ihm können mit den <Cursor-Tasten> Dateien markiert (<Leertaste>) und in ein Unterverzeichnis (<Eingabe-Taste>) verzweigt werden.

5.1.2 Beschreibung der Funktionen

Info
Gibt Auskunft über die Versionsnummer des Programms und den noch zur Verfügung stehenden Speicherplatz.

Ende
Beendet das Programm und unterbricht eine bestehende Verbindung.

Selektieren
Markiert alle Dateien, die der angegebenen Suchmaske entsprechen.

Unselektieren
Hebt die Markierung bei allen Dateien, die der Suchmaske entsprechen auf.

Betrachten
Die Datei, auf der der Auswahlbalken steht, wird angezeigt. Man kann zwischen der Anzeige in hexadezimaler und ASCII-Form wählen.

Kopieren
Ist ein zweiter Rechner angeschlossen und die Verbindung aufgebaut, so werden die markierten Dateien auf den anderen Rechner übertragen.

Umbenennen
Die angewählte Datei erhält einen neuen, einzugebenden Namen.

Löschen
Die selektierten Dateien werden nach Rückfrage gelöscht.

Laufwerk wechseln
Man kann das anzuzeigende Laufwerk aus einer Liste der angeschlossen Laufwerke wählen.

Verzeichnis erstellen
Ein neues Unterverzeichnis wird angelegt.

Verzeichnis umbenennen
Ein Unterverzeichnis erhält einen neuen Namen.

Verzeichnis löschen
Ein leeres Verzeichnis wird aus dem Verzeichnisbaum entfernt.

Verbindung aufbauen

In einem Fenster kann man die Schnittstelle (COM1/COM2) und die Arbeitsweise des Rechners (aktiv/passiv) wählen. Desweiteren kann die Verbindung mit verminderter Geschwindigkeit aufgebaut werden. Dies kann nötig sein, wenn ein verhältnismäßig langsamer Rechner verwendet wird oder die Verbindung gestört ist. Es muß stets ein aktiver mit einem passiven Rechner verbunden werden. Der aktive Rechner kann vom Anwender bedient werden.

Verbindung unterbrechen

Die Verbindung zwischen den Rechnern wird unterbrochen.

Optionen einstellen

Hier können die Zeilenzahl und die Farbpalette ausgewählt werden.

Dateibetrachter

Diese Funktion dient zum Betrachten einer Datei, die zuvor angewählt wurde.

Wechsel des Betrachters	<F2>
Scrollen durch den Text:	<Zeile rauf/runter>
	<Zeichen rechts/links>
Seitenweise	<Bild rauf/runter>
Beginn/Ende der Zeile	<Pos1>, <Ende>
Beginn/Ende des Textes	<Strg+Bild rauf/runter>
Verlassen des Betrachters	<ESC>

Kommandozeilen-Parameter

Die Verbindungseinstellungen können schon während des Programmstarts in der Kommandozeile eingegeben werden.

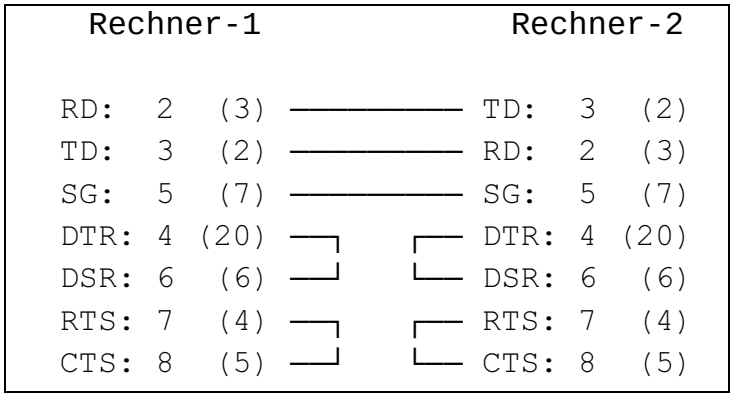
Der Programmaufruf muß folgende Form haben:

CPLINK [COMx A/P S/L]

Wobei COMx die Schnittstelle angibt (x: 1 oder 2) und A/P den Betriebsmodus (Aktiv oder Passiv) festlegt. Der dritte Parameter legt die Übertragungsgeschwindigkeit fest, also ob die Daten schnell oder langsamer übertragen werden sollen. Es müssen stets alle drei Parameter oder keiner angegeben werden!

5.1.3 Verbindungskabel

Die serielle Schnittstelle eines PCs stellt entweder einen 9- oder 25-poligen Anschluß zur Verfügung. Um Daten zwischen zwei Rechnern austauschen zu können, müssen diese wie folgt über ihre seriellen Schnittstellen verbunden werden:



Die Zahlen geben den Pin des Anschlusses an einem 9-(bzw. 25)-poligen Stecker an.

5.2 Programm zur Visualisierung des SPS-Prozeßabbilds

Das Programm S5KOP dient im aktuellen Ausgabestand zur Visualisierung des Prozeßabbilds im Prozessor des Automatisierungsgeräts.

Das MS-DOS-Programm S5KOP.EXE hat in der Version 1.1 vom 8.4.1991 eine Größe von 75677 Bytes. In der SPS muß der VIPA-Hantierungsbaustein FB1 (aus CP386COM oder CP486COM) für die CP3/4 vorhanden sein.

Der Aufruf von MS-DOS aus erfolgt mit dem Befehl "S5KOP <Return>". Es wird ein Titelbild angezeigt, das durch Drücken einer beliebigen Taste verschwindet. Sie befinden sich nun im Hauptmenü.

Durch Drücken von <F1> wird das Menü "Prozeßabbild" aktiviert. Dort können Sie über die Funktionstasten wählen, was zyklisch angezeigt werden soll:

<F1> Anzeige der Eingänge
<F2> Anzeige der Ausgänge
<F3> Anzeige der Merker 0..127
<F4> Anzeige der Merker 128..255
<F5> Anzeige der Timer
<F6> Anzeige der Zähler
<F8> Anzeige der Kachel

Die zyklische Anzeige kann durch Betätigen der Pause-Taste <F7> unterbrochen werden bis zum nächsten Tastendruck.

Nach Betätigen einer Auswahltaste F1 .. F6 oder F8 wird solange zyklisch das entsprechende Prozeßabbild angezeigt, bis Sie eine neue Wahl treffen.

Durch Betätigen der Taste <F7> ("Pause") wird der aktuelle Zustand bis zum nächsten Tastendruck eingefroren.

Mit <ESC> verlassen Sie das Menü Prozeßabbild und gelangen wieder ins Hauptmenü sowie wiederum mit <ESC> von dort zurück auf die DOS-Kommandozeile.

5.3 System-Testprogramm

Dieses System-Testprogramm ist auf der optional erhältlichen Diskette mit Original-Quadtel-Software enthalten (Softwarepaket SW583, MS-DOS-Diskette, 3.5", 720KB). Diese Software wird nur als Einfachnutzen verkauft, d.h. für jedes System, auf dem dieser Treiber eingesetzt wird ist eine Lizenz notwendig.

Das Paket enthält u.a. ein Diagnose-Programm für die CP3/4. Das Diagnose-Programm kann unter MS-DOS geladen werden und bietet eine Liste von Tests aller wichtigen Systemkomponenten an. Diese Tests können einzeln oder auch als Paket durchlaufen werden.

Anhang

Anhang

A Tabellenverzeichnis

B Index