S T E P 5

S T E P 5 - P L C - P R O G R A M M I E R U N G

Stand: Dokumentation 20.08.2009
Software Step5-Assembler v1.41 bis v1.52
Dokumentenname: Step5

- PLC-Programmierung in Siemens-Step5®-Sprachsyntax
- spezielle Erweiterungen für JBG-Steuerungen
- deutsche und englische Symbolik wählbar
- Assembleraufrufe in JBG-PC-Anwendungen integriert (z.B. J-CAM, ISOCAM)

Inhalt:

1 Aufgabe und Zweck von PLC–Anwendungen

Der Begriff PLC stammt aus dem Englischen und bedeutet soviel wie Programmierbare Logik–Kontrolle. Die PLC stellt, einfach gesagt, die verbindende Einheit zwischen CNC–Abläufen (sequentielle Fahrinformationen) und der Ein-/Ausgabeeinheit dar. Ein CNC- Programm kann mittels M-Funktionen Aktionen in der PLC auslösen; umgekehrt kann der PLC Einfluss auf das auszuführende CNC-Programm nehmen. Im Normalfall besteht die Hauptaufgabe der PLC darin, eine, sonst durch Hardware aufwendig zu erstellende (elektrische oder elektronische) Logik, durch Programmierung in einer passenden Sprache nachzuempfinden. Derartige PLC–Einheiten sind in den JBG–Produkten GMS, SML33 und GMS-M vorhanden.
Analog zur Programmierung des CNC-Ablaufs (z.B. DIN 66025 in J-CAM) existiert eine eigene Sprache: STEP5®. STEP5 stammt von Siemens® und wird zur Programmierung der Simatic® SPS–Steuerungen eingesetzt. Viele SPS-Programmierlehrgänge haben diese oder darauf aufbauende Sprachen zur Grundlage.

Im weiteren finden Sie hier eine Beschreibung der speziellen Eigenschaften dieser Implementation von STEP5 und den Unterschieden zur Programmierung bei Siemens-Steuerungen.

Wenn Sie weiterführende Informationen benötigen können Sie das Handbuch 'Siemens: S5-115U Automatisierungsgerät' mit der Bestellnummer 6ES5998-0UF12 zu Rate ziehen. STEP und SIMATIK S5 sind eingetragene Warenzeichen der Siemens AG und gesetzlich geschützt.

2 Funktionsweise der PLC

Ein PLC-Programm besteht aus einer Reihe einzelner Befehle, die zyklisch, parallel zu den restlichen Aufgaben in der Steuerung abgearbeitet werden. Da der ausführende Teil der PLC völlig firmwaremäßig gelöst ist, ist die Ausführungsgeschwindigkeit geringer als die einer 'echten' SPS-Steuerung. Jedoch ist sie leistungsfähig genug, um kleinere bis mittlere Steueraufgaben mit vollem Komfort zu bewältigen (siehe auch 4.12. Ausführungsgeschwindigkeit).Die Befehle der PLC-Steuerung sind in mehrere Gruppen aufgeteilt, innerhalb derer ähnliche Operationen zusammengefasst sind:

- Gruppe 1: bitorientierte Verknüpfungs- und Speicheroperationen
- Gruppe 2: byte-/wortorientierte Lade-/Transferoperationen
- Gruppe 3: Zeiten- und Zähleroperationen
- Gruppe 4: Arithmetik
- Gruppe 5: Vergleichsoperationen
- Gruppe 6: Baustein-Aufruf-Operationen
- Gruppe 7: Ablaufsteuerung + Randgruppen

Bis die PLC für eine bestimmte Anwendung betriebsbereit ist, sind folgende Schritte nötig:

- Schreiben eines STEP5-Programms in einem Texteditor bzw. im Editor der Oberfläche.
- Konvertieren (Assemblieren) des zuvor geschriebenen Quellprogramms mit dem Hilfsprogramm STEP5.EXE.
- Laden des konvertierten Programms in die Steuerung (Download) und aktivieren des PLC-Zyklus.
- Austesten des Programms mit dem Debugger.

2.1 Aufbau eines PLC-Programms

Grundsätzlich enthalten PLC-Programme zwei unterschiedliche Bereiche von Information: Daten- und Programmtext. Diese sind in zwei oder mehreren Programm-Blöcken untergebracht, die Daten- und Funktionsbausteine genannt werden. Grundsätzlich sind im Programmtext Daten- vor den Funktions-bausteinen einzugeben. Im Gegensatz zur STEP5-PLC in Siemens-Steuerungen sind folgende Eigenschaften vorhanden:

- Es wird kein spezielles Programmiergerät (PG) benötigt. Die Eingabe sowie das Entwickeln der Programme geschieht am normalen Personal-Computer.
- Die Unterteilung des Programmtexts in Organisations- (OB) und Funktionsbausteine (FB) sowie PBs und SBs entfällt. Alle Befehle werden in Funktionsbausteinen programmiert.
- Datenbausteine (DB) werden nur in ihrer Grösse definiert. Es findet keine automatische Initialisierung statt.
- Die Unterteilung in Netzwerke entfällt. Der Editor bietet eine Suchfunktion, mit der bestimmte Programm-

stellen gefunden werden.
- Zeilen-Formatierung und weitere Ratschläge siehe 5.3.

Beispiel für ein komplettes PLC-Programm:

```
DB 100
      DS 6            in DB100 Platz für 6 Byte. DW0 bis DW2
      BE              Baustein (DB100) Ende

FB 21                 Automatisch nach Einschalten...
      A  DB 100       Datenbaustein DB100 selektieren
      L  KH 2         Konstante 2H in Accu1 laden
      T  MB 30        Inhalt Accu1 in Merkerbyte 30 transferieren
      BE

FB 1                  Standard-Zyklus...
NAME: main            Name der Routine
STRT: U  E 0.0        Lade Eingang Nummer 0.0 in VKE
      U  E 0.1        Verknüpfe mit Eingang Nummer 0.1
      =  A 0.0        Verknüpfungsergebnis in Ausgang 0.0
      BE
```

Im ersten Abschnitt (DB 100) stellt das Programm einen Datenbereich von 6 Byte (3 Datenworte) zur Verfügung. Im *Funktionsbaustein* FB21 werden Operationen ausgeführt, die nur einmal (nach Einschalten der PLC) ausgeführt werden sollen. Im Beispiel wird DB100 als Datenbaustein selektiert und das Merkerbyte MB30 mit 2 (Hex.) belegt. FB1 stellt in jedem PLC-Programm das Hauptprogramm dar (muss immer vorhanden sein!). Wenn alle Befehle im FB1 abgearbeitet sind (BE erreicht ist), beginnt die Ausführung wieder mit dem ersten Befehl in FB1, was ein zyklisches Bearbeiten des gesamten Programms zur Folge hat.

2.2 Sinn und Funktionsweise der Prozess-Abbilder

Die Steuerung ist nicht in der Lage bei jedem Befehl, der auf Eingänge zugreift, den momentanen Pegel des Eingangs zu lesen. Ebenso wird ein Befehl, der einen Ausgang verändert nicht unmittelbar den Pegel beeinflussen. Diese Vorgänge würden die PLC zu langsam machen. Somit bedient man sich eines Zwischenspeichers, der die Eingänge vor jedem Zyklus aufnimmt und die Ausgänge nach dem Zyklus stellt. Diese Zwischenspeicher werden Prozessabbilder genannt (PAA für Ausgänge und PAE für Eingänge). Befehle der PLC greifen also auf das PAA/E zu und nicht direkt auf die Hardware.

2.3 Die Transfer- und Rechenregister (Accus)

Die PLC verfügt über zwei Akkumulatoren (Accu1 und Accu2), die dem Transportieren von 8- bzw. 16-Bit-Gruppen dienen. Ausserdem werden die Rechen- und Vergleichsoperationen (Arithmetik) mit den Inhalten der Accus ausgeführt. Beide Accus sind 16-Bit-Register; werden 8-Bit-Gruppen (Merker / Prozessabbilder / Daten) geladen, so werden nur die unteren (niederwertigen) 8 Bits des Accu1 mit dem Operanden gefüllt; die oberen (höherwertigen) 8 Bit des Accu1 werden genullt. Wird ein 16-Bit-Wert geschrieben, so wird das niederwertige Byte an die kleinere**!** Byteadresse platziert (Unterschied zur Siemens-Hardware). Im weiteren wird für eine 8-Bit-Gruppe der Begriff *Byte* und für eine 16-Bit-Gruppe der Begriff *Wort* benutzt.

3 Programmierung

3.1 Struktur der Funktionsbausteine

Längere PLC-Programme können nicht mehr sinnvoll als reine sequentielle Liste von Befehlen geschrieben und bearbeitet werden. Dazu steht eine Unterprogrammtechnik zur Verfügung: Aus dem Haupt-FB (immer FB1) können beliebige andere FBs (FB2..FB239) aufgerufen werden. Bei einem Aufruf wird die Bearbeitung mit dem ersten Befehl im aufgerufenen FB begonnen; nach Beendigung des aufgerufenen FBs, kehrt die Bearbeitung wieder in den rufenden FB zurück und jener wird fortgesetzt. Dieses Verfahren ist geschachtelt bis zu einer Tiefe von 16 möglich. Bausteine die als Unterprogramme genutzt werden sollen, werden standardmäßig unterhalb von (hinter) FB1 angeordnet.

Beim Einschalten der PLC (z.B. nach dem Download des PLC-Programms, oder beim Eintritt in eine Notaus-

Situation) wird, falls programmiert, FB21 ausgeführt, bevor das Hauptprogramm (FB1) angesprungen wird. Nach einem Power-Up-Reset (Einschalten des Gerätes) wird FB22 ausgeführt (falls nicht vorhanden → FB21).

3.2 Einsatz von Makros

In einen PLC-Programm können Makros (= Textersetzungen) definiert und eingesetzt werden. Makros dienen dazu, Kombinationen aus Operand und Wert (z.B. ‚E 1.7') oder nur Wert-Angaben (z.B. ‚1.7') im Programmtext zu ersetzen. Der Einsatz von Makros verbessert die Lesbarkeit von PLC-Programmen und kann eine große Hilfe bei Änderungen der E/A-Belegung sein. So kann z.B. die Umlegung eines häufig verwendeten Eingangs auf einen anderen Pin mit der Änderung einer einzigen Programmzeile erledigt werden. Makros werden folgendermaßen definiert und aufgerufen...

Definition:		#DEF	m_name m_text	Bsp1:	`#DEF eStart E 1.7`
				Bsp2:	`#DEF Stop 2.4`
Aufruf:	[sym:]	Befehl	$m_name	Bsp:	`U $eStart`
oder	[sym:]	Befehl	Operand $m_name	Bsp:	`U E $Stop`

Makros müssen definiert sein bevor sie aufgerufen werden können. Aus diesem Grund finden Makro-Definitionen normalerweise am Programmbeginn statt.

Die #DEF-Anweisung muss in der ersten Spalte beginnen, der Makroname *m_name* darf aus max. 12 Zeichen bestehen und sollte mit einem Buchstaben beginnen. Der Makro-Textersatz *m_text* kann bis zu 10 Zeichen lang sein und darf beliebige Zeichen enthalten. Ein möglicher Kommentar kann min. 11 Zeichen nach Beginn des Textersatzes *m_text* oder ab Spalte 27 beginnen. Wird eine dieser Regeln verletzt, so erfolgt eine entsprechende Warnung bzw. Fehler-Meldung.

Aktuelle JBG-Programme mit PLC-Entwicklungsumgebung sind in der Lage, den Textersatz beim Debuggen des Programms anzuzeigen (linke Spalte im Debugger).

3.3 Bedingte Assemblierung

Durch diese Funktion können bestimmte Programmteile von der Assemblierung ausgenommen werden, z.B. wenn Abschnitte eines Standard-Programms bei einer einfacheren Anwendung nicht benötigt werden, man aber den entsprechenden Programmtext nicht entfernen sondern nur unwirksam machen möchte. Die bedingte Assemblierung verfügt über folgende Anweisungen:

`#if wert`	Die folgenden Zeilen werden nur assembliert, wenn *wert* ungleich 0 ist. *wert* wird normalerweise durch ein (einzelnes) Makro repräsentiert, kann aber auch 0 oder 1 sein. Der Assemblierungs-Bereich endet mit einer #else oder #endif-Anweisung
`#else`	Umkehrung der Bedingung für den nachfolgenden Programmbereich.
`#endif`	Ende des bedingt assemblierten Bereichs.

Bedingungen können auch geschachtelt werden (max. 10 #if-Bereiche ineinander). Verneinungen oder andere Verküpfungen in *wert* sind nicht möglich. Ist die Bedingung `wert` nicht definiert, so wird eine entsprechende Warnung ausgegeben.

Beispiel:

```
  #DEF  PRGST  1        durch Eingang CNC-Programm starten, oder nur anzeigen
...
        U  E 1.2        Eingang 1.2 in VKE
  #if PRGST             (wenn PRGST = 1)
        S  M 0.2        Starten eines CNC-Programms
  #else                 (anderenfalls)
        =  A 1.1        (nur) Anzeigen auf Ausgang 1.1
  #endif                (Ende Bedingung)
```

3.4 Erweiterung der Grenzwert-Definitionen

Beim übersetzen (assemblieren) der PLC-Datei werden die Operanden überprüft und beim Verlassen der Bereichsgrenzen werden Fehlermeldungen erzeugt. Die Grenzen für die Wertebereiche der Operanden stammen im Normalfall aus der Datei DEFAULT.S5C (bzw. einer anderen, gerätespezifischen Datei). Für manche Geräte kann es notwendig sein, diese Grenzen in der PLC-Datei zu überschreiben (z.B. GMS96 hat 256 statt 64 Merkerbytes). Hierzu können folgende Anweisungen eingesetzt werden...

`#IBC x`	Zahl zulässiger Eingangsbytes auf x setzen
`#OBC x`	Zahl zulässiger Ausgangsbytes auf x setzen
`#FBC x`	Zahl zulässiger Merkerbytes auf x setzen
`#SFC x`	Zahl der speziellen Merkerbytes auf x setzen
`#DWC x`	Zahl zulässiger Datenwort auf x setzen
`#TRC x`	Zahl zulässiger Zeiten-Register auf x setzen
`#CRC x`	Zahl zulässiger Zähler-Register auf x setzen
`#IND f`	Definitionsdatei f einlesen (z.B. '`#IND DIM_ZGN`')

4 Befehle der PLC

Seit Version v1.10 können die Befehle in deutscher oder wahlweise englischer Schreibweise programmiert werden. Die Grundeinstellung ist Deutsch; um Englisch anzuwählen, programmieren Sie an beliebiger Stelle eine eigene Zeile der Form '`#LE `'. Alle Zeilen unterhalb dieser Programmstelle werden englisch interpretiert. Um wieder Deutsch zu erzwingen ist '`#LG`' zu programmieren. In der weiteren Beschreibung wird nur die deutsche Schreibweise erläutert; eine Umsetztabelle für Englisch finden Sie unter 4.11.

In der folgenden Auflistung wird das *-Zeichen verwendet, um die Buchstaben M, E und A zu ersetzen. Z.B. könnte der Befehl '`L *B 3`' die Kombinationen '`L MB 3`', '`L EB 3`' oder '`L AB 3`' ergeben. Die drei Bereiche Merker, PAE und PAA verhalten sich somit, von der Programmierung her, identisch. Weiterhin werden folgende Symbole verwendet:

- VKE Verknüpfungsergebnis (1-Bit-Accumulator)
- nn Bytenummer innerhalb des jeweiligen Bereichs (0..255max)
- tt Nummer eines Zeit-Registers (Timer)
- zz Nummer eines Zähl-Registers (Zähler)
- pp Nummer eines Bausteins (0..255)
- b Bitnummer (0..7)
- k 4-Bit-Konstante, Schiebezähler 0..15 (Arithmetik)
- kk 8-Bit-Konstante (Bereich 0..255)
- kkkk 16-Bit-Konstante (Bereich 0..65535 oder -32768..+32767)
- aaaa Absolute Offsetadresse im Gesamtprogramm
- ff Nummer eines formalen Operanden

Die Kurzzeichen ABC (?, = und]) hinter der Befehlssyntax beschreiben die Eigenschaften des Befehls:

- A → ? VKE-abhängig: Der Befehl wird nur ausgeführt, wenn VKE gesetzt ist (statischer Zustand).
- A → ^ VKE-flankenabhängig: Ausführung nur, wenn VKE einen Zustandswechsel (061, 160) an diesem Befehl aufweist.
- B → = VKE- beeinflussend: das VKE kann sich durch diesen Befehl ändern.
- C →] VKE-begrenzend: Die Verknüpfungsfähigkeit des VKE endet mit diesem Befehl. Beim nächsten Verknüpfungs-Befehl wird VKE nur geladen (normalerweise wir der U-Befehl verwendet).
- _ Der Unterstrich steht für Eigenschaften A,B oder C, die der entsprechende Befehl nicht hat. Somit stehen bei jedem Befehl 3 der aufgeführten Zeichen.

4.1 Gruppe 1: Verknüpfungsoperationen

- U * nn.b _=_ UND-Verknüpfung mit VKE. Abfrage auf Signalzustand "1".
- U T tt _=_ E Abfrage Zustand eines Timers.
- U Z zz _=_ E Abfrage Nicht-Null-Zustand eines Zählers.
- UN * nn.b _=_ UND-Verknüpfung mit invertiertem Operand. Abfrage Signalzustand "0".
- UN T tt _=_ E Abfrage Zustand eines Times (invertiert)

- UN Z zz _=_ E Abfrage Null-Zustands eines Zählers
- O * nn.b _=_ ODER-Verknüpfung mit dem VKE. Abfrage auf Signalzustand "1".
- O T tt _=_ E Abfrage Zustand eines Timers.
- O Z zz _=_ E Abfrage Nicht-Null-Zustand eines Zählers.
- ON * nn.b _=_ ODER-Verknüpfung mit invertiertem Operand. Abfrage Signalzustand "0".
- ON T tt _=_ E Abfrage Zustand eines Times (invertiert)
- ON T zz _=_ E Abfrage Null-Zustands eines Zählers
- U(_=] Das VKE des folgenden Klammerausdrucks wird mit dem vorherigen VKE UND-verknüpft.
- O(_=] ODER-Verknüpfung von Klammerausdrücken.
-) _=_ Abschluss und VKE-Bildung eines Klammerausdrucks
- R * nn.b ?_] Dem Operanden "0" zuweisen, wenn VKE=1
- S * nn.b ?_] Dem Operanden "1" zuweisen, wenn VKE=1
- = * nn.b __] zugewiesen wird der Zustand des VKE.

4.2 Gruppe 2: Lade- und Transferoperationen

- L *B nn ___ Laden des Byte-Operanden Merker/PAA/PAE in Accu
- L *W nn ___ Laden eines Wort-Operanden (16-Bit).
- L DR nn ___ Laden der unteren 8 Bit des Datenworts nn.
- L DL nn ___ Laden der oberen 8 Bit eines Datenworts in die unteren 8 Bits des Accu1.
- L DW nn ___ Laden aller 16 Bit eines Datenworts.
- L T tt ___ Laden Timer-Stand (100'stel-Sekunden).
- L Z zz ___ Laden Zähler-Stand
- L KH kkkk ___ Laden einer 16-Bit-Konstanten in Accu1. Die verschiedenen Konstantentypen (z.B. KF) werden beim assemblieren in 'KH kkkk' umgeformt.
- L KC xy ___ Konstante aus ein oder zwei ASCII-Zeichen. *Hinweis:* bis zur Version 1.33 des Step5-Assemblers waren in KC nur Großbuchstaben möglich.
- T *B nn ___ Transferieren der unteren 8 Bit von Accu1 in das Merker/PAE/PAA-Byte nn.
- T *W nn ___ Transferiere in gewähltes Wort (16-Bit)
- T DR nn ___ Transf. in unteres Byte eines Datenwortes
- T DL nn ___ Transf. in oberes Byte eines Datenwortes
- T DW nn ___ Transf. gesamten Accu1 in Datenwort.

4.3 Gruppe 3: Zeiten- und Zähleroperationen

Zeit–Operationen (Timer = T):

Zur Programmierung von Zeitfunktionen steht eine gesonderte Befehlsgruppe zur Verfügung. Die Zeiten werden auf Grund von Flanken am VKE (Starteingang) gestartet. Eine Flanke bedeutet, dass das VKE bei *zwei aufeinanderfolgenden* PLC-Zyklen am entsprechenden Startbefehl unterschiedliche Zustände aufweist. Eine Flanke ist steigend, wenn VKE zuerst 0 und anschliessend 1 ist; beim umgekehrten Ereignis ist sie fallend. Der Zustand eines Timers (Zeitregisters) wird durch Verknüpfungsfunktionen abgefragt (siehe 4.1.)

- SI T tt ^_] Starten einer Zeit als Impuls. Steigende Flanke an VKE startet die Zeit; VKE=0 löscht die Zeit. Abfrage liefert 1, solange die Zeit läuft.
- SV T tt ^_] Verlängerter Impuls. Zeit wird mit steig. Flanke an VKE gestartet; bei VKE=0 bleibt die Zeit unbeeinflusst. Zustand 1 solange Zeit läuft.
- SE T tt ^_] Einschaltverzögerung. Startverhalten wie SI. Abfragen liefern 1, wenn Zeit abgelaufen und VKE=1 noch ansteht.
- SS T tt ^_] Speichernde Einschaltverzögerung. Startverhalten wie SV. Abfragen liefern 1, wenn die Zeit abgelaufen ist. Zustand wird erst dann 0, wenn der Timer mit dem 'R'-Befehl (siehe unten) rückgesetzt wurde.
- SA T tt *_] Ausschaltverzögerung. Die Zeit wird bei fallender Flanke (Zustandswechsel 160) gestartet. VKE=1 löscht die Zeit. Abfragen liefern 1, wenn VKE=1 oder die Zeit läuft.
- R T tt ?_] Rücksetzen/Löschen einer Zeit. Die Zeit wird auf den Anfangswert gesetzt, wenn VKE=1.
- FR T tt ^_] Startbedingung (Flankenerkennung) für die Zeit rücksetzen.

Zeitkonstanten:
Um eine Zeit zu starten, muss der entsprechende Zeit-Wert in Accu1 hinterlegt werden. Dazu werden Konstanten in einem speziellen Format programmiert:

```
KT  wert.basis   wert: Zeitwert bezogen auf basis
                 basis: 0 / 1 / 2 / 3 → 0.01 / 0.1 / 1 / 10 s
KT  40.0         = 0.01 * 40 = 0.4 s
```

⌦ Alle Zeiten werden intern in Basis 0.01s umgerechnet. Die programmierte Zeit darf jedoch 5 Minuten (bzw. 40.95 Sekunden bei Zeitbasis 0) nicht übersteigen.
max: 30.3 / 300.2 / 3000.1 / 4095.0)

Beispiel:

```
U   E 1.0       Start-Eingang für die Zeit
L   KT 50.1     Zeitwert 5 Sekunden
SI  T 1         Starte Zeit, wenn Flanke an VKE
U   T 1         Abfrage Zustand des Timers
=   A 0.2       Zustand des Timers in Ausgang übertragen
```

Zähler–Operationen (Zähler = Z):

Die Zählerfunktionen dienen dem einfachen Erfassen von Ereignissen. Sie sind, wie die Zeitfunktionen flankengetriggert. Die Inhalte der Zähler sind binär (nicht BCD-) codiert. Der Bereich von Zählern ist 0..999; im April 2000 wurden die Zählfunktionen kompatibel zu Siemens eingeführt, zuvor war der Zählbereich 0..65535 und es gab Überläufe bei den Zählfunktionen ZV und ZR.

- S Z zz ^_] Setzen eines Zählers. Bei einer Flanke an VKE wird der Zählerwert aus Accu1 gesetzt (0..999, wenn > 999 wird 999 gesetzt).
- R Z zz ?_] Der Zähler wird 'genullt', wenn VKE=1.
- ZV Z zz ^_] Zähler wird bei Flanke am VKE um 1 erhöht (bis max. 999).
- ZR Z zz ^_] Zähler wird bei Flanke am VKE um 1 verringert (bis min. 0)

Die Abfrage eines Zählers (z.B. mit U–Befehl) liefert 1, wenn der Inhalt des Zählers ungleich 0 ist.

4.4 Gruppe 4: Arithmetik

Die arithmetischen Funktionen arbeiten mit den beiden Accus. Im Normalfall werden zuerst die Operanden geladen (in Accu1/2) und anschliessend wortweise verknüpft. Das Ergebnis einer solchen Funktion findet sich in Accu1 wieder.

- +F ___ Accu1 = Accu2 + Accu1 (Addition).
- -F ___ Accu1 = Accu2 - Accu1 (Subtraktion)
- UW ___ Accu1 = Accu2 & Accu1 (UND-Verknüpfung)
- OW ___ Accu1 = Accu2 | Accu1 (ODER-Verknüpfung)
- XOW ___ Accu1 = Accu2 ^ Accu1 (Exclusiv-ODER)
- TAK ___ Inhalte Accu1 mit Accu2 vertauschen.

Beispiel:

```
L   MW 42       Lade Merkerwort 42 (Accu1)
L   KF 1        Lade Konstante 1 (Accu1, zuvor Accu1 → Accu2)
+F              addiere
T   MW  42      Transfer Ergebnis (Accu1) in Merkerwort 42
```

Operationen die nur mit Accu1 arbeiten...

- SLW c ___ c mal Linksschieben des Inhalts von Accu1. Freiwerdende Stellen werden genullt!
- SRW c ___ c mal Rechtsschieben (Accu1).
- KEW ___ 1er-Komplement Accu1 (Negierung aller Stellen in Accu1).
- KZW ___ 2er-Komplement von Accu1 (Accu1 = -Accu1).
- D kk ___ Low-Byte von Accu1 um kk dekrementieren.
- I kk ___ Low-Byte von Accu1 um kk inkrementieren.

Beispiel:

```
L   MW 42       Laden Merkerwort 42 (Accu1)
SRW 1           schiebe Accu1 nach rechts = Teilen durch 2
T   MW 42       Ergebnis zurück in Merkerwort 42
```

4.5 Gruppe 5: Vergleichsoperationen

Die Vergleichsoperationen arbeiten mit den beiden Accus. Als Gedankenstütze können die Funktionen als 'Accu2 (Operation) Accu1' angesehen werden. Das Vergleichsergebnis findet sich im VKE wieder und steht für weitere Verknüpfungen zur Verfügung...

- !=F _=_ Gleichheit (VKE=1, wenn Accu1 = Accu2).
- ><F _=_ Ungleichheit (VKE=0, wenn Accu1 = Accu2).
- >F _=_ Grösser (VKE=1, wenn Accu2 grösser Accu1).
- >=F _=_ Grösser oder gleich (VKE=1, wenn Accu2 grösser oder gleich Accu1)
- <F _=_ Kleiner (VKE=1, wenn Accu2 kleiner Accu1).
- <=F _=_ Kleiner oder gleich (VKE=1, wenn Accu2 kleiner oder gleich Accu1).

Beispiel:

```
  L   MW 42       Lade Merkerwort 42 in Accu1
  L   KF 13       Lade Konstante 13 (Accu1, zuvor Accu1 → Accu2)
  >=  F           Vergleich:  Accu2 >= Accu1; wenn ja → VKE=1 sonst VKE=0
  SPB =GROS       Springe, wenn Accu2 >= Accu1 (= wenn VKE=1)
```

4.6 Gruppe 6: Baustein-Aufruf-/Sprung-Operationen

Sprungoperationen führen gezielte Verzweigungen auf Grund des VKE bzw. von arithmetischen Operationen aus. Ein Sprungbefehl hat als Operand eine Zieladresse, auf die verzweigt wird, wenn die Bedingung des Befehls wahr ist. Sprungziele können im gesamten Programm liegen (keine Bereichsgrenzen). Der Assembler STEP5 verhindert jedoch, dass über Bausteingrenzen hinaus gesprungen wird.

- SPA =aaaa `__]` Springe unbedingt an Adresse aaaa.
- SPB =aaaa `?=]` Springe, wenn VKE=1. VKE wird zwingend auf 1 gesetzt.
- SPZ =aaaa `___` Springe, wenn Accu1 (Ergebnis) = 0.
- SPN =aaaa `___` Springe, wenn Accu1 ungleich 0.
- SPP =aaaa `___` Springe, wenn Accu1 > 0 (positiv ungleich 0)
- SPM =aaaa `___` Springe, wenn Accu1 < 0 (negativ ungleich 0)
- SPO =aaaa `___` Springe, wenn Überlauf aufgetreten. Dies kann bei Operationen +F, -F, und KZW sein.

Beispiele:

```
      U   E 0.0         Bedingung in VKE laden
      SPB =ZIEL         wenn VKE=1 → springe zu 'ZIEL'
      ...               Programm, wenn Sprung nicht ausgeführt
ZIEL: ...               Sprungziel
      L   MW 42         Lade Merkerwort 42 ind Accu1
      SPZ =ENDE         wenn Accu1 = 0 → springe zu 'ENDE'
      ...
ENDE: ...
```

Baustein-Aufruf-Funktionen werden benötigt, um PLC-Programme in logische und funktionelle Einheiten zu unterteilen. Sie dienen der Strukturierung und der besseren Lesbarkeit eines Programms.

- SPA pp `__]` Aufruf des Funktionsbausteins 'FB pp' ohne Bedingung. Das VKE bleibt unbeeinflusst.
- SPB pp `?=]` Baustein 'FB pp' wird aufgerufen, wenn VKE=1. Falls VKE=0 (kein Sprung) wird es für die weitere Verarbeitung auf 1 gesetzt.
- BE `__]` Bausteinende. Schliesst einen Baustein ab und führt in den aufrufenden Baustein zurück ohne das VKE zu beeinflussen.
- BEA `__]` Rücksprung ohne Bedingung. Der Programmtext des gerufenen Bausteins wird durch diese Operation nicht beendet (Ende des Bausteintextes nur mit BE). VKE unbeeinflusst.
- BEB `?=]` Rücksprung, wenn VKE=1. Falls VKE=0 (kein Rücksprung) folgt Weiterbearbeitung des gerufenen Bausteins mit VKE=1.
- A pp `___` Festlegen eines Daten-Bausteins 'DB pp' als aktuellen Datenbaustein.

In Funktionsbausteinen werden Programmtexte mit unterschiedlicher Bedeutung programmiert. Jedes PLC-Programm verfügt z.B. über einen 'FB 1', in dem das Hauptprogramm untergebracht ist. Nach jedem erfolgreich beendeten Zyklus wird FB 1 erneut (automatisch) aufgerufen. Aus FB 1 heraus werden, dann Funktionsbausteine aufgerufen, die z.B. M-Funktionen der CNC-Einheit bedienen oder Überwachungsfunktionen übernehmen.

Beispiel:

```
  FB 1                  Funktionsbaustein 1 (Hauptprogramm)
      ...
      U     E  0.0      Laden des Eingangs 0.0 in VKE
      SPB   FB  40      wenn VKE (=E0.0) =1, rufe FB 40
      ...               (weiterer Programmtext in FB1)

  FB 40                 Funktionsbaustein 40 (Unterprogramm)
      U     E  0.1      Lade E0.1
      U     E  0.2      UND-Verknüpfung mit E0.2
      =     A  0.0      Lade Ergebnis in Ausgang 0.0
      BE                Beende Baustein FB 40
```

Das gezeigte Programm ruft FB 40 nur dann auf, wenn der Eingang 0.0 auf 1-Pegel ist. Somit wird die UND-Verknüpfung innerhalb FB 40 durch Eingang 0.0 'maskiert'.

Durch das Anlegen mehrerer Datenbausteine kann ein und dasselbe Programm mit unterschiedlichen Daten arbeiten. Dazu ist es wichtig dem PLC-Programm die Anfangsadresse des jeweiligen Datenbausteins mitzuteilen. Ausgehend von dieser Anfangsadresse werden die Datenworte, die durch 'DW nn' adressiert sind, angewählt. Die Anweisung DS reserviert eine bestimmte Anzahl von *Bytes;* da der Datenbereich *wortweise* organisiert ist, sollte der angegebene Wert immer geradzahlig sein.

Beispiel:

```
DB 100                  Datenbaustein DB 100
    DS    2             erstes Datenwort in DB 100 (ALPHA)
    BE                  Ende DB 100
DB 101                  zweiter Datenbaustein DB 101
    DS    2             erstes Datenwort in DB 101 (BETA)
    BE

FB 1                    Funktionsbaustein FB 1 (Hauptprogramm)
    A     DB  100       Anwahl Datenbaustein DB 100
    L     DW  0         Lade Datenwort Nr.0 (=ALPHA) aus DB 100
    A     DB  101       Selektiere DB 101
    L     DW  0         Lade Datenwort Nr.0 (=BETA) aus DB 101
    ...
```

4.7 Gruppe 7: Ablaufsteuerung und Randgruppen

In dieser Gruppe befinden sich Befehle die programmsteuernde Funktionen haben, bzw. sich nicht direkt in die anderen Gruppen einordnen lassen...

- STP ___ Stop. Laufender Zyklus wird noch zu Ende bearbeitet. Anschliessend steht die PLC.
- STS ___ Stop sofort. Zyklus nicht zu Ende bearbeiten.
- NOP i ___ Befehl ohne Funktion. Nur aus Kompatibilitätsgründen zu anderen Sprachen vorhanden. Parameter i kann 0 oder 1 sein.
- B DW nn ___ Indizierte Bearbeitung. Die nachfolgende Operation wird mit Byte-
- B MW nn ___ und (falls nötig) Bitnummer ausgeführt, die im angegebenen Merker- oder Datenwort angegeben sind.

Die Funktion 'indizierte Bearbeitung' (auch Bearbeitungsfunktion genannt) wird folgendermaßen programmiert:

Beispiel:

```
    L   KH 0205     Lade Hex.-Konstante 205 (02=Bit, 05=Byte-Nr.)
    T   DW 12       Transferiere in Datenwort 12
    B   DW 12       indizierte Bearbeitung mit DW 12
    U   E 0.0       Bearbeitete Funktion. Wird zu 'U E 5.2'
    ...
```

Die Byte-Nummer steht im Low-Byte des Daten-/Merkerwortes, die Bit-Nummer (falls erforderlich) im High-Byte. Im indiziert programmierten Befehl (unmittelbar auf B-Funktion folgend) wird die Byte-/Bit-Adresse 0 programmiert. Indiziert bearbeiten können Sie die Befehle:

U, UN, O, ON, S, R, =, SI, SA, SV, SS, SE, L, T.

➔ Die Beeinflussung von Merker–Werten unterhalb *M 2.0* kann bei den Geräten SML33, GMS92/94, GMS-I und GMS-M blockiert sein!

4.7.1 Sprünge mit B-Anweisung, Sprungleiste

Diese Programmiertechnik ist vor allem in Schrittketten einsetzbar. Das (durch die B-Anweisung referenzierte) Datum enthält einen Zahlenwert = Argument im Bereich 0..255; nach der B-Anweisung wird eine Reihe von Sprungbefehlen SPA= geschrieben auf die eine NOP -Anweisung folgt. Das Argument entscheidet darüber, welcher der Sprungbefehle zur Ausführung kommt, 0 bedeutet dabei, dass der erste Sprungbefehl ausgeführt wird. Ist das Argument größer oder gleich der Anzahl der Sprungbefehle, so wird die Programmstelle nach dem NOP –Befehl ausgeführt. Geräte, deren Firmware vor August 2000 erstellt wurde enthalten diese Eigenschaft nicht, in diesem Fall wird der erste Sprungbefehl nach der B-Anweisung und der dort folgende Befehl indiziert ausgeführt.

Beispiel:

```
      B     DW  12      Datenwort 12
      SPA   =L001       Sprung auf L001, wenn DW 12 = 0
      SPA   =L002       Sprung auf L002, wenn DW 12 = 1
      SPA   =L003       Sprung auf L003, wenn DW 12 = 2
      NOP               DW 12 enthält Wert 3..255
```

4.8 Formalparametrierte Befehle

Funktionsbausteine, die als Unterprogramme aufgerufen werden, können mit Parametern zur Ausführung ihrer Funktion versorgt werden. Diese Parameter werden im Programm des gerufenen Bausteins definiert und müssen beim Aufruf des Bausteins in gleicher Weise angegeben werden.

Beispiel:

```
      ...
      SPA   FB  120     Aufruf FB 120
NAME:       VERK        Name von FB 120 = 'VERK'
PAR1:       E  4.5      erster Parameter Name 'PAR1' Wert E4.5
PAR2:       E  3.2      zweiter Parm. Name 'PAR2' Wert E3.2
ERGE:       A  1.2      dritter Parm. Name 'ERGE' Wert A1.2
...
FB 120                  Unterprogramm FB 120
NAME: VERK              Name des Unterprogramms 'VERK'
BEZ:  PAR1  E           erster Parm. Typ: Eingangs-Bit-Adresse
BEZ:  PAR2  E           zweiter Parm. Typ:  "
BEZ:  ERGE  A           dritter Parm. Typ: Ausgangs-Bit-Adresse
      U     =PAR1       eigentliche Funktion. Lade Parm1 in VKE
      UN    =PAR2       UN mit zweitem Parm.
      =     =ERGE       übertragen in 3.Parm. 'ERGE'
      BE                Baustein-Ende
```

Beim Aufruf eines Unterprogramms mit Parametern ist es sehr wichtig, dass die Zahl und die Art der Parameter mit den Angaben im gerufenen Baustein exakt übereinstimmen (eine Prüfung beim Assemblieren findet nicht statt). Folgende Parameterarten sind zulässig:

- E Eingangsparameter. Bit-/Byte-/Wort-Adresse (E nn.b / EB nn / EW nn / M nn.b / MB nn / MW nn / DR nn / DL nn / DW nn).
- A Ausgangsparameter. Bit-/Byte-/Wort-Adresse (A nn.b / AB nn / AW nn / M nn.b / MB nn / MW nn / DR nn / DL nn / DW nn).
- T/Z Timer- oder Zählernummer (T tt / Z zz).
- D Konstante (16-Bit) KF/KH/KC/KT/KB. *Hinweis:* bis zur Version 1.33 des Step5-Assemblers waren in KC nur Großbuchstaben möglich.
- B Baustein-Nummer (intern 16-Bit) DB pp / FB pp

Die zusätzliche Angabe von BI (für Bit-Adresse), BY (für Byte) und W (für Wort), wie in der Siemens-Syntax notwendig, kann hier entfallen. Ebenso ist es nicht notwendig, die Parameterart D durch KH... genauer zu spezifizieren.

Beispiele:

Aufruf-Parm:	**DW 93** →	Definition:	BEZ: beta A (W)
		Ausführung:	T =beta
Aufruf-Parm:	**KF 127** →	Definition:	BEZ: inp1 D (KF)
		Ausführung:	L =inp1
Aufruf-Parm:	**M 5.2** →	Definition:	BEZ: memo E (BI)
		Ausführung:	O =memo

Ausnahmen:

Die Zeiten- und Zähler–Operationen sind nicht direkt mit Formalparametern aufrufbar. Die folgenden Befehle entscheiden durch den Typ des Operanden, welchen Zweck (Zeit- oder Zähler–Funktion) sie erfüllen.

- SSV SS (mit T-Operand) oder ZV (mit Z-Operand)
- SAR SA (mit T-Operand) oder ZR (mit Z-Operand)
- SVZ SV (mit T-Operand) oder S (mit Z-Operand)

4.9 Nicht verfügbare Befehle

Nicht alle, in der Siemens-Step5-Sprache enthaltenen Befehle sind in PLC-Programmen verfügbar. Dazu gehören z.B. alle Befehle, die BCD–Zahlen verarbeiten. Übersicht:

- LC, LC=, LW= Laden BCD–codierter (oder Bitmuster-) Zahlen
- E Erzeugen eines Datenbausteins zu Laufzeit
- P, PN Bit eines Zählworts prüfen
- LIR, TIR, TNB Spezial–Transfer–Befehle
- BI Indirektes Bearbeiten von Formaloperanden
- ADD Addieren von Konstanten zu Accu1
- xF, :F Multiplizieren, Dividieren (FB 242/243 siehe 4.10.)
- Doppelwort, Floatzahlen Arithmetik nicht integriert
- Bit-Test-Funktionen nicht vorgesehen

Manche der nicht enthaltenen Befehle werden auch nicht benötigt, da die Struktur nicht exakt gleich der SIMATIK S5 ist:

- SPR Langsprung (alle Sprungbefehle sind ‚langsprungfähig')
- SU, RU Unbedingtes Setzen von (System-)Datenbits
- BLD Befehle zum Bildaufbau

4.10 Integrierte Funktionsbausteine

Integrierte Funktionsbausteine (IF) sind bereits in den jeweiligen Steuerungen vorhanden, brauchen also nicht vom Benutzer programmiert zu werden. IF belegen die Nummern 240 bis 255 und werden mit den Befehlen SPA FB nnn oder SPB FB nnn gerufen. Derzeit sind in den Produkten SML33 / GMS / GMS-M und GMS-I folgenden IF enthalten...

FB 242: Multiplikation
Ein Aufruf von FB 242 bewirkt eine Multiplikation von Accu1 mit Accu2. Die beiden Zahlen werden als vorzeichenlose 16-Bit-Werte angesehen. Das Ergebnis der Operation (Produkt) ist in Accu1 (niederwertige 16 Bit) und in Accu2 (höherwertige 16 Bit) zu finden. Durch diese Strategie wird der Einsatz von Formal-Parametern und das Umladen der Operanden vermieden.

FB 243: Division
Der Aufruf von FB 243 dividiert Accu2 (Dividend) durch Accu1 (Divisor). Es handelt sich auch hier um zwei vorzeichenlose 16-Bit-Zahlen. Das Ergebnis wird in Accu1 (Quotient) und Accu2 (Rest) abgelegt. Eine Division durch Null führt zu einem Ergebnis von 65535 (KH FFFF) und sollte im Normalfall durch eine Abfrage des Divisors vor der Operation abgefangen werden.

FB 244: Sende-Unterstützung für Meldungen
Bei gesetztem VKE wird versucht, die Daten in Accu1 in das jeweilige Sende-Register (SR = MB4) zu übertragen. Dies erfolgt, wenn das SR leer angetroffen wird (VKE bleibt gesetzt). Kann der Übertrag wegen des nicht leeren SR nicht erfolgen, so wird VKE rückgesetzt. Mit dieser Funktion wird das Senden von Meldungen bei Flanken an Fehler-Merkern erleichtert. FB 244 ist in folgenden Geräten verfügbar: SML33 ab v3.19, GMS ab v1.51, GMS-I ab v2.00 und GMS-M ab v1.04.

FB 254: Multifunktions-Baustein
Dieser Baustein umfasst eine Vielzahl von Funktionen, die teils gerätespezifisch sind. Einen Überblick über die einzelnen Funktionen finden Sie im separaten Dokument FB254.

24-Bit Modulo-Division (Funktion 23)

Diese Funktion ist universell ausgelegt, dient jedoch normalerweise der Untersuchung von Achspositionen. Es kann festgestellt werden, ob die Position auf einer geraden Teilung einer Gesamtstrecke liegt. Hierzu werden der Funktion folgende Parameter in einem Datenbaustein übergeben (offs = Wert in Accu2):

DW offs+0:	Low-Word der zu untersuchenden Position (P = 24-Bit)
DW offs+1:	High-Byte der zu untersuchenden Position
DW offs+2:	Low-Word der Gesamtstrecke (S = 24-Bit)
DW offs+3:	High-Byte der Gesamtstrecke
DW offs+4:	Teilung (D = 2..999*)
DW offs+5:	Toleranz (T = 1..32767)
DW offs+6:	Rückgabewert der Funktion: letzte Raster-Position (Low-Word) (R)
DW offs+7:	Rückgabewert der Funktion: letzte Raster-Position (High-Byte)

Die Daten bis offs+5 müssen vor dem Aufruf von FB 254 eingetragen werden. Nach dem Aufruf sind folgende Daten vorhanden: VKE=1/0 → Position *liegt / liegt nicht* auf einer Teilung; Accu1 = Nummer der Teilung, wenn VKE=1 oder -1, wenn VKE=0; im Rückgabewert ist eine naheliegende Teilungsposition zu finden.

Beispiel: Eine Drehachse hat 100000 Inkremente/Umdrehung. Es soll untersucht werden, ob die aktuelle Achsposition auf einem 30 Grad-Raster liegt → S=100000 (offs+0 = KF 86A0, offs+1 = KF 01), D=12 (= 360° / 30°), T=5 Inkremente Toleranz. P=12000 (43.2°) → VKE=0, Accu1= -1, R=8333. P=25003 (90°) → VKE=1, Accu1=3, R=25000.

Setzen des Positionswertes einer Achse (Funktion 24)

Diese Funktion setzt einen 24-Bit Positionswert, der im Datenbereich abgelegt ist, in einer Achse. Da diese Operation mehrere Millisekunden in Anspruch nimmt, ist sie mit eine Status-Abfrage ausgestattet, die über das VKE gewählt wird (VKE=0 → *Status-Abfrage*, VKE=1 → *Auftrag* zum Setzten der Position). Die *Status-Abfrage* liefert VKE=1, wenn die Funktion verfügbar bzw. abgeschlossen ist. Für den *Auftrag* werden folgende Parameter in einem Datenbaustein übergeben (offs = Wert in Accu2):

DW offs+0:	Low-Word der zu setzenden Position (P = 24-Bit)
DW offs+1:	High-Byte der zu untersuchenden Position
DW offs+2:	Achsindex (Slot-Nummer 0..5 bei GMI99, 0 bei Achskarten)

Beachten Sie, dass diese Funktion eine direkte Wechselwirkung mit dem CNC-Programm hat und niemals während einer laufenden Fahrt ausgeführt werden sollte; am sichersten ist die Ausführung im Rahmen einer M-Funktion. Nach dem *Auftrag* muss gewartet werden, bis die Funktion abgeschlossen ist *(Status-Abfrage)*. Anschließend sollte absolut gefahren werden (G90).

ML-Funktionen (Funktion 26)

ML-Funktionen stellen eine komfortable Methode dar, Zusatzgeräte zu bedienen und zu beobachten, die von der PLC kontrolliert werden. Hierzu übermittelt das übergeordnete Gerät (PC, MCC) eine Funktionsnummer sowie Tastenbefehle (F2..F6) und erwartet Status-Informationen zurück.
Beim Aufruf entscheidet Accu2.15, ob es sich um eine Abfrage oder eine Status-Übermittlung handelt...

Abfrage:	Aufruf:	Accu2 = KH 8000
	Rückgabe:	VKE → 0=Funktion nicht aktiv 1=Funktion aktiv
		Accu1 → Nummer der ML-Funktion (1000..9999)
		Accu2.4 - .0 → Tastatur-Info F6 - F2.
Status-Überm.:	Aufruf:	Accu2.15 → 0
		Accu2.14 → Statusanzeige B ein / aus
		Accu2.12 - .8 → Stati der Statusanzeigen der F-Tasten
		Accu2.7 - .0 → Statusanzeige A ein / aus / Wert (8-Bit)
	Rückgabe:	keine

Folgende Geräte beherrschen ML-Funktionen: GMI99 ab v4.67, ED-Reihe ab v5.21.

Achs-Schutzfunktionen (Funktion 27)

Dies ist ein spezieller Mechanismus in der Steuerung (Achs- und Interpolationskarten sowie FIS), der Achsbewegungen blockiert, solange bestimmte Bedingungen in der PLC anstehen. Die PLC kann Freigaben für *geschützte* Achsen erteilen und bekommt Informationen darüber, ob Achsbewegungen blockiert wurden.

Hinweis: Bewegungen aus dem eigenen *Indexer* interpolierter Achskarten können nicht blockiert werden.

Abfrage:	Aufruf:	Accu2 = KH 8000
	Rückgabe:	VKE → 0=keine Achse wurde blockiert, 1=mindestens eine Achse wurde blockiert Accu1 → blockierte Achsen (.0=Slot0, .1=Slot1 ...)
Freigaben:	Aufruf:	Accu2 → einzelne Freigaben (.0=Slot0, .1=Slot1 ...)
Anwahl:	Aufruf (nur FIS):	Accu2 = KH 4000 + einzelne Anwahlen ((.0=Slot0, .1=Slot1 ...)

Folgende Geräte unterstützen Achs-Schutzfunktionen: GMI99 ab v4.67, ED-Reihe ab v5.22, GMS96 ab v3.97. Die FIS-Steuerung beherrscht die Achs-Schutzfunktion ab v1.45; hierbei werden die jeweiligen Fahr-Kommandos an die Achs-/Interpolationskarten blockiert und evtl. laufende CNC-Programme abgebrochen. Zu schützen den Achsen müssen in der FIS mit Accu2 = KH 4xxx (siehe oben) angewählt werden.

4.11 Englische Schreibweise der Befehle

Alle oben beschriebenen Befehle können statt der deutschen auch mit englischen Abkürzungen geschrieben werden. Hier nun eine Tabelle der jeweiligen Entsprechungen...

Dt.	*Engl.*	*Dt.*	*Engl.*
U E/A/M	A I/Q/F	UN E/A/M	AN I/Q/F
O E/A/M	O I/Q/F	ON E/A/M	ON I/Q/F
S E/A/M	S I/Q/F	R E/A/M	R I/Q/F
= E/A/M	= I/Q/F	U(	A(
L EB/EW	L IB/IW	L AB/AW	L QB/QW
L MB/MW	L FY/FW	L KZ/KC	L KC/KS
T EB/EW	T IB/IW	T AB/AW	T QB/QW
T MB/MW	T FY/FW	B DW	DO DW

Dt.	*Engl.*	*Dt.*	*Engl.*	*Dt.*	*Engl.*	*Dt.*	*Engl.*
B MW	DO FW						
SI T	SP T	SV T	SE T	SE T	SD T	SA T	SF T
ZV Z	CU C	ZR Z	CD C	S Z	S C	R Z	R C
SPA	JU	SPB	JC	BEA	BEU	BEB	BEC
A	C	U=	A=	UN=	AN=	SI=	SP=
SE=	SD=	SVZ=	SEC=	SSV=	SSU=	SAR=	SFD=
FR Z	FR C	KEW	CFW	KZW	CSW	SPA=	JU=
SPB=	JC=	SPZ=	JZ=	SPN=	JN=	SPP=	JP=
SPM=	JM=	SPO=	JO=	B=	DO=		

➔ Befehle mit identischer Entsprechung sind nicht aufgeführt
➔ Ein Hilfsprogramm zum Übersetzen von PLC-Programmen ist vorhanden (siehe 6.4)

4.12 Ausführungsgeschwindigkeit

Der ausführende Teil der PLC ist komplett in der Firmware der Geräte realisiert. Dadurch ist die Ausführungsgeschwindigkeit geringer als bei echten SPS-Steuerungen. Im Gegenzug bedeutet dies aber wiederum, dass keine Kosten durch aufwendige Hardware entstehen. Es sind nur Ein- und Ausgabekarten hinzuzufügen.

Genaue Angaben über Ausführungszeiten einzelner Befehle können hier nicht gemacht werden, da sie stark von den restlichen Aufgaben abhängen, welche die Steuerung parallel bearbeiten soll. Darum werden hier Zeiten für 1000 PLC-Befehle aus einem durchschnittlichen Programm angegeben, wobei nach jeweils 100 Befehlen die Prozess-Abbilder aufgefrischt werden. Vorausgesetzt ist bei diesen Zeiten, dass das Gerät keine anderen Aufgaben parallel ausführt:

Gerät	*Randbedingung(en)*	*Zeit für 1000 Befehle*
GMS	Lageregler aktiv, Indexer inaktiv	190 ms (ohne Lageregler 135ms)
GMS	Lageregler und Indexer aktiv	280 ms

GMS96	Lageregler und Indexer aktiv	80 ms (siehe Dokument GMS96)
E-Reihe	Lageregler und Indexer aktiv	30 ms
GMS-M	mit 1 SPE und 1 SPA	140 ms
GMS-I	Indexer aktiv	200 ms
GMI99	Indexer aktiv	45 ms
SML33	Indexer inaktiv	210 ms (mit 1 SPE und 1 SPA)
SML33	3-Achs-Interpolation 5/12 kHz	270 / 410 ms
SML33	Kreisinterpolation 5 kHz	440 ms
SML34	Indexer inaktiv	150ms
SML34	3-Achs-Interpolation 5/12/20 kHz	180 / 210 / 290 ms
SML34	Kreisinterpolation 5/10 kHz	220 / 370 ms

Eine Siemens-SPS-Steuerung mit der CPU 941 würde für das selbe Testprogramm ca. 43ms (pro 1000 Befehle) benötigen (Betrachtung ohne Bedienen der Prozessabbilder; reine Addition der Befehlszyklus-Zeiten).
Bei SML33/34-Karten und sehr kurzen PLC-Programmen begrenzt die Bedienung der E/A-Baugruppen die Ausführungsgeschwindigkeit. Bei längeren PLC-Programmen kann dieser Vorgang parallel ausgeführt werden; sind nach einem PLC-Durchlauf die Eingänge noch nicht gelesen, so muss die PLC auf diese Information warten. Die SML33 benötigt im Stillstand (Indexer inaktiv) ca. 500 us, um eine Ausgabekarte und 1ms, um eine Eingabekarte zu bedienen (SML34: 300/650 us). Diese Zeiten wachsen, im selben Verhältnis wie die Ausführungszeiten, mit steigender Schrittausgabefrequenz.

5 Der Assembler STEP5

Zur Umsetzung des Klartext-PLC-Programms (Quell-Datei) in eine Form, welche die Steuerungen leicht interpretierten können, steht der Assembler STEP5 zur Verfügung. Er überprüft das Programm (Quell-Code) und erzeugt den, von der Steuerung ausführbaren Maschinen-Code. Ausserdem erzeugt er eine Listing-Datei, die für das Debuggen (Austesten) des PLC-Programms nötig ist.

5.1 Aufruf von STEP5.EXE

Im Normalfall wird STEP5 automatisch aus der CNC-Oberfläche (z.B. J-CAM) aufgerufen. Sie können STEP5 jedoch auch explizit zum Übersetzen eines PLC-Programms nutzen. Aufruf beim DOS-Prompt:

STEP5 *dateiname* [Cn] [Llll] [Ddefname]

wobei:

STEP5	Name der ausführbaren MS-DOS-Datei (STEP5.EXE)
dateiname	Name des PLC-Programms (Erweiterung .PLC kann entfallen)
Cn	Erzeugen einer Command-Datei zur Ausgabe mit MAN. Der Maschinen-Code wird für eine Steuerkarte mit Adresse 'n' erzeugt.
Llll	Sprachanpassung zur Ausgabe von Fehlermeldungen. Eine Sprachdatei namens 'TEXT_lll.S5C' muss vorhanden sein. (z.B.TEXT_D.S5C)
Ddefname	Name einer Definitionsdatei, in der die Eingabegrenzen für PAA/PAE/Merker... angegeben sind. (z.B. DEFAULT.S5C)
Aa Ee	Zahl der Ausgangsbytes a und Eingangsbytes e. Diese Angaben überschreiben die Werte aus der Definitionsdatei.
/w	Warte nach Ende der Assemblierung auf Tastendruck
/j*m*	*m* ist ein Zahlenwert mit Steuerflags für die Zusammenarbeit mit J-CAM: .0 = .cmj statt .cmd erzeugen.

5.2 Funktionsweise des Assemblers

Der Assembler benötigt zwei Durchläufe, um den *Maschinencode* zu erzeugen. In Lauf 1 (= PASS1) wird eine Zwischendatei angelegt, die Syntax geprüft und die Symbole gesammelt. Ab v1.21 findet auch eine Prüfung der Klammerebenen statt. Im zweiten Lauf werden die, durch Zahlenwerte ersetzten, Symbole eingefügt. In diesem zweiten Lauf entsteht auch die Listing-Datei mit der Erweiterung .lst. Wird eine Command-Datei für

den Download in eine Steuerung (Erweiterung .cmd bzw. -cmj) gewünscht, so erstellt STEP5 diese anschliessend aus der Listing-Datei.

5.3 Konventionen bei der Programmierung

Um ein einwandfreies Funktionieren des Assemblers zu erreichen, muss der Quelltext einigen Richtlinien entsprechen:

- Befehle bzw. Operationen beginnen in der 7. Spalte (Zählweise mit 1 beginnend).
- Symboldefinitionen (Sprungziele, FB-, NAME- und BEZ-Anweisungen) beginnen in Spalte 1.
- Ab Spalte 27 wird jeder Text als Kommentar betrachtet. So ist keine spezielle Markierung für Kommentare nötig.

Somit Zeilenaufbau:

```
                    1         2         3         4        6
Spalte:   12345678901234567890123456789012345678901234 // 901
Text:     symb: bef oprnd/wert      kommentar.............]
Beispiel: L000: U   E 0.0           Und-Verknüpfung
```

- Befehl (bef) und Operand (operand) müssen durch mind. 1 Leerzeichen getrennt sein. Operand und Wert (wert) sollten durch exakt 1 Leerzeichen getrennt sein; dies verbessert das Ergebnis von Suchoperationen.
- Der Programmtext (Kommentar) sollte nicht über Spalte 59 gehen, das dieser Text im PLC-Debugger nicht eingesehen werden kann.
- Kleinbuchstaben bei Labels und Befehlen sind erlaubt, werden aber intern in Grossbuchstaben gewandelt. Vorsicht: 'Symb' = 'SYMB'.
- Es muss ein Hauptprogramm 'FB 1' vorhanden sein.
- Die speziellen Funktionsbausteine FB 21 und 22 werden folgendermaßen eingesetzt:
  FB 21 und 22 vorhanden: FB 22 läuft nach Einschalten der Steuerung (Power-Up-Reset), 21 nach Download oder Notaus.
  nur 21 oder 22 vorh.: Läuft nach Einschalten + Download / Notaus.
  weder 21 noch 22 vorh.: es wird sofort in FB 1 gesprungen.
- Datenbausteine sind vor Funktionsbausteinen zu programmieren.

5.4 Meldungen beim Assemblieren

STEP5 generiert zwei unterschiedliche Arten von Meldungen, wenn eine Unstimmigkeit im Quellprogramm entdeckt wird:

1. Warnungen
Sie sind Hinweise auf nicht einwandfrei programmierte Stellen. Das Maschinen-Programm kann jedoch erstellt werden.

2. Fehler
Falls Fehler gemeldet werden, kann das Maschinenprogramm nicht erstellt werden. Das Quellprogramm muss korrigiert werden.

Alle Meldungen werden in einheitlicher Form auf dem Bildschirm ausgegeben:

Warnung/Fehler: Zeile #nnn, Beschreibung der Meldung

Durch die angegebene Zeilennummer finden sie leicht die fehlerhafte Stelle im Quellprogramm. STEP5 zeigt ausserdem Informationen über den aktuell durchgeführten Lauf (PASS1/PASS2) und die momentan bearbeitete Zeile an. Ein korrekt assembliertes Programm hinterlässt folgende Anzeige auf dem Schirm:

STEP5 Assembler v*V.VV* (c)JBG 1992-2007
PASS1... [*dateiname*.PLC]
PASS2...
nnn Zeilen assembliert, *0* Fehler, *0* Meldung(en)
Schreibe Down-Load-Datei *dateiname*.CMJ

6 Zusatzfunktionen in weiteren Anwenderprogrammen

6.1 Der Debugger

Mit dem Debugger (Austester), der in verschiedenen JBG-Programmen vorhanden ist, können Sie die Funktion des PLC-Programms überprüfen und Fehlreaktionen einfach und schnell erkennen. Sie erhalten Ihre Quelldatei auf dem Bildschirm (genauer gesagt, die Listing-Datei), in der Sie eine Zeile auswählen können. Für diese Zeile erhalten Sie Anzeigen über:

- die Ausführung dieser Programmstelle. Bei jedem 'Vorbeikommen' der PLC an der gewählten Stelle wird eine Wechselanzeige umgeschaltet.
- Den aktuellen Zustand des VKE.
- Einen 2-Byte Ausschnitt aus dem PAA-Bereich.
- Einen 2-Byte Ausschnitt aus dem PAE-Bereich.
- Einen 2-Byte Ausschnitt aus dem Merker-Bereich.

Die 2-Byte-Ausschnitte können Sie selbst wählen. Somit ist es Ihnen möglich einen Einblick in (nahezu) die gesamten Daten der PLC zu bekommen.

6.2 Festlegen statischer Daten

Wenn die Anwendung (J-CAM, ISOCAM...) über einen Menüpunkt „Daten und Randbedingen“ im PLC-Menü verfügt, so können Sie dort (vom PC aus) einen Teil des Datenbereichs der PLC einsehen und editieren. Dazu muss jedoch eine spezielle Programmierung (Schreibweise) der Datenbausteine erfolgen:

Beispiel:

```
DB 100
   DS    2          ?K Zahl der Durchläufe
   DS    2          ?T Zeit bis zum Einschalten (s)
   DS    4          ?R Registerwert
   BE
```

Es können nur jeweils 16-Bit-Werte (1 Datenwort) bzw. 24-Bit-Werte (Register) als „editierbar“ definiert werden (DS 2 bzw. DS 4). Ein editierbares Datenwort wird kenntlich gemacht, indem in den ersten Zeichen des Kommentars (ab Spalte 27) ein Fragezeichen und ein Konstantenbezeichner eingetragen wird. Als Bezeichner stehen 'K' (für Konstante 16-Bit, DS 2), 'T' (Zeitwert, programmierbar in Sekunden) und 'R' (Registerwert 24-Bit ohne Umrechnung, DS 4) zur Verfügung.
Im Anwenderprogramm sehen sie dann unter der Funktion „Daten und Randbedingungen“ den aktuellen Wert des Datenwortes (bzw. der Datenworte) und den Text des Kommentars hinter dem Bezeichner. Manche Anwendungen unterscheiden zwei getrennte Eingabemasken bzw. unterschiedliche Schutzstufen bei der Eingabe, die durch Kleinbuchstaben als Bezeichner angewählt werden (z.B. J-CAM: „?K“ → *PLC-Daten*, „?k“ → *Allgemeindaten*, die Werte werden in getrennten Tabellen angezeigt).

Die Typen K und T müssen auf 2er-Adressen, der Typ R auf 4er-Adressen liegen. Wird gegen diese Regel verstoßen, so erfolgt möglicherweise die Meldung ‚Falsche Position des Wertes’ beim Aufruf der Funktion *Daten und Randbedingungen*. Die Adresse einer DS-Anweisung ermitteln Sie, indem Sie die Werte aller DS-Anweisungen (aller Datenbausteine) vor der betreffenden DS-Anweisung addieren; eine 2er-Adresse muss durch 2, eine 4er-Adresse durch 4 ohne Rest teilbar sein.

Die jeweils zwei Datenworte eines Registerwerts müssen in der PLC entsprechend aufwendiger gehandhabt werden. Überträge in das höherwertige Wort (nur untere 8-Bit gültig) sind nur durch zusätzliche Befehle möglich. Normalerweise werden Registerwerte nur für die Beobachtung von CNC-Registern von Achs- oder Interpolationskarten verwendet.

Wichtig: Nach Änderungen im Quelltext zuerst assemblieren.

6.3 M-Funktionen in CNC-Programmen

Ein CNC-Programm kann mittels M-Funktionen Aktionen in der PLC auslösen. Dazu wird im Merkerbereich der PLC ein, zu der entsprechenden M-Funktion gehörendes Bit gesetzt. Die PLC muss nun die vorgesehene Reaktion ausführen und anschliessend das bereits erwähnte Bit wieder löschen. Das CNC-Programm läuft erst nach dem Löschen dieses Bits weiter. In unseren CNC-Oberflächen stehen für die Steuerungen folgende

M-Funktionen bereit:

PLC-Merker:	M6.0	M6.1	M6.2	M6.3	M6.4	M6.5	M6.6	M6.7
M-Funktion:	M50	M51	M52	M53	M54	M55	M56	M57
PLC-Merker:	M7.0	M7.1	M7.2	M7.3	M7.4	M7.5	M7.6	M7.7
M-Funktion:	M58	M59	M60	M61	M62	M63	M64	M65
PLC-Merker:	M8.0	M8.1	M8.2	M8.3	M8.4	M8.5	M8.6	M8.7
M-Funktion:	M66	M67	M68	M69	M70	M71	M72	M73
PLC-Merker:	M9.0	M9.1	M9.2	M9.3	M9.4	M9.5	M9.6	M9.7
M-Funktion:	M74	M75	M76	M77	M78	M79	M80	M81
PLC-Merker:	M10.0	M10.1	M10.2	M10.3	M10.4	M10.5	M10.6	M10.7
M-Funktion:	M82	M3	M4	M5	M86	M7	M8	M9

Zusätzlich wird in Merkerbyte MB11 die Erweiterung der zuletzt aktivierten M-Funktion eingeblendet (Bsp. im CNC-Programm: 'M67.8' → MB11 = 8).

6.4 CRF.EXE und P-Trans.EXE

Diese Zusatzprogramme zum Step5-Assembler erzeugen...

eine Kreuzreferenzliste des PLC-Programms (CRF.EXE)

Das Programm CRF.EXE erstellt eine Querverweisliste (CRF-Datei) aus einer Listdatei (LST-Datei). Dabei werden die Bereiche Eingänge, Ausgänge, Merker, Zeiten, Zähler und Daten analysiert. Jede Referenz aus dem entsprechenden Bereich wird in der CRF-Datei mit FB-Nummer und Zeilen-Nummer vermerkt. Ein * hinter der Zeilen-Angabe bedeutet, dass auf die Referenz hier schreibend (verändernd) zugegriffen wird.

CRF wird normalerweise mit nur einem Parameter (Name der List-Datei, Erweiterung .LST kann entfallen) aufgerufen. Weitere Parameter sind optional für die Druck-Steuerung (Papiergröße): Lnn [Zahl der Zeilen pro Seite](68), Wnn [Zahl der Spalten pro Zeile](77) und Rnn [Zahl der Leerzeichen für den linken Rand](8). Die Angaben in runden Klammern sind die Default-Werte, die benutzt werden, wenn der Parameter fehlt.

eine Datei mit Befehlscodes in einer bestimmten Sprache (P-Trans.EXE)

Um die Befehle eines PLC-Programms in eine einheitliche Sprache zu übersetzen benutzen Sie P-Trans.EXE. Alle Befehleszeilen (ausgenommen wegkommentierte Zeilen) werden in die Ziel-Sprache (deutsch oder englisch) übersetzt. Als Ausgabe erzeugt P-Trans eine Datei mit dem Namen der Quell-Datei und der Erweiterung .E oder .G.

Die Programme erklären ihre Aufruf-Parameter, wenn sie ohne Parameter aufgerufen werden.