Software-Module

Die Flexibilität der Anlagenplanung durch die Verwendung von Maschinenmodulen sollte tunlichst durch eine entsprechende Flexibilität auf dem Softwaresektor unterstützt werden. plan *& simulation* bietet daher ein modulares Softwarekonzept, das die einfache "Konstruktion" des Steuerungsprogramms erlaubt.

Zwischen den Maschinenmodulen und den Softwaremodulen besteht eine direkte Beziehung, die auch schon durch die Namensgebung verdeutlicht wird:

Beispiele:	Maschinenmodul	Softwaremodul
Maschinentisch	MT/2WKSf	MT_2WKSf
Maschinenständer	MS/1	MS_1
Revolverkopf	WSS/R	WSS_R
Förderband	FB/270mm	FB_270_405
Förderband	FB/405mm	FB_270_405

Lediglich funktionsgleiche Module sind in einem Softwaremodul zusammengefaßt, wie die letzten beiden Beispiele zeigen.

Dieses Datenblatt enthält eine Erläuterung zur Benutzung der Softwaremodule unter Turbo-Pascal® Version 5.0 (oder folgende) sowie ein Anwendungsbeispiel.

Staudinger GmbH Schönbühler Straße 5, D 8311 Loiching 1
Telefon 08731/5069-0 Telefax 08731/5069-10

Hinweise

Die Sammlung der Software-Module wird mit der Entwicklung eines neuen Maschinen-Moduls auch immer um ein dazugehöriges Software-Modul ergänzt werden.

Die gemeinsamen Eigenschaften der Software-Module werden in künftigen Entwicklungen weiter gepflegt, soweit technisch vertretbar. Investitionen in Software-Ebenen oberhalb der Module werden sich daher auch in Zukunft amortisieren.

Soweit erforderlich oder dringend wünschenswert werden auch bestehende Software-Module weiterentwickelt. Dies wird ebenfalls unter Wahrung bestmöglichster Kompatibilität geschehen.

Über Weiterentwicklungen werden wir unsere registrierte Kundschaft informieren.

Die derzeit gelieferten Softwaremodule erfordern die Verwendung eines Turbo-Pascal®-Compilers der Version 5 (oder folgende). Compiler früherer Versionen können ebenfalls verwendet werden, wenn im Konstantendefinitionsteil von INDTREIB.INC und SOFTMOD.INC die berechneten Konstanten durch feste Konstanten ersetzt werden.

Die Fehlerfreiheit der gelieferten Software kann durch Staudinger GmbH nicht garantiert werden. Insbesondere muß die Software durch Softwareteile des Kunden ergänzt werden, um als Ganzes in Betrieb genommen zu werden. Staudinger GmbH übernimmt daher keine Haftung für Schäden, sei es an dem mechanischen Aufbau, den Interface-Modulen, der Computerausstattung, sonstigen Folgeschäden oder Personenschäden.

Für Rückmeldungen über bestehende Fehler oder Unverträglichkeiten ist die Staudinger GmbH dankbar.

Programmiertechnik

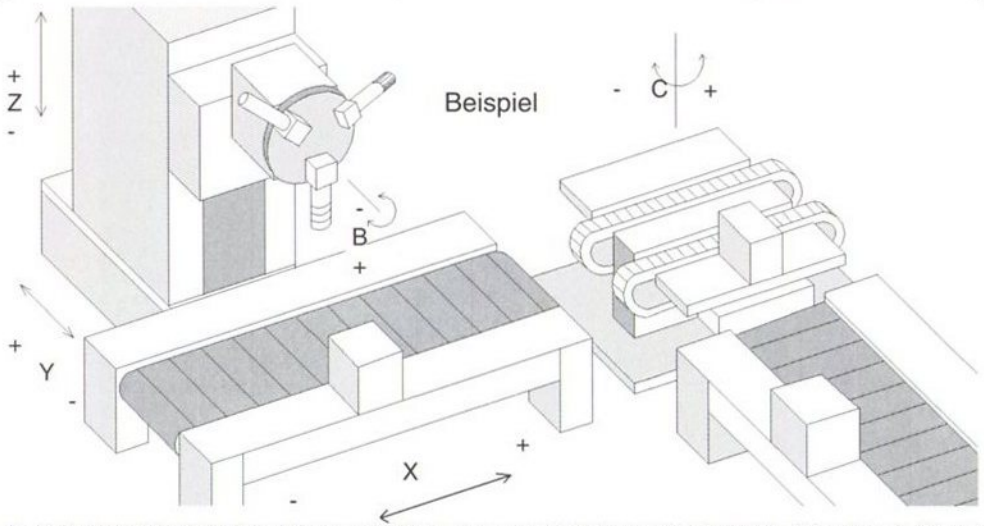
Die Programmierung einer Anlage aus Modulen von plan *& simulation* in Turbo-Pascal® erfolgt durch Aufruf von Softwaremodulen. Nahezu alle Softwaremodule besitzen folgenden schematisch gleichen Prozedurkopf:

```
Procedure Modulname(Nr : Integer; Auftrag : t_Auftraege; Var: Schritt : Integer);
```

Es besteht eine einfache und direkte Korrespondenz zwischen dem Namen des Maschinenmoduls und dem Namen der steuernden Prozedur. Lediglich funktionsgleiche Module werden durch ein einziges Softwaremodul gesteuert.
Der Parameter **Nr** (Typ Integer) bezeichnet die Nummer des Maschinenmoduls. Da in einer Anlage mehrere identische oder funktionsgleiche Module vorkommen können, werden sie mit der im Bereich 1 bis 50 zu wählenden Nummer identifiziert. Sollten mehr als 50 Module eines Typs vorliegen, so kann ein größerer Nummernbereich per Konstante **Max_Modul_Type** in SOFTMOD.INC eingestellt werden.
Der Parameter **Auftrag** bezeichnet die durch das Modul durchzuführende Steuerungsaufgabe. Der Auftrag, ein Maschinenteil in positiver X-Richtung bis zum Betätigen eines Endtasters zu fahren, mag z.B. **Positionieren_Plus_X** lauten. In den Datenblätter der Module ist aufgeführt, welche Aufträge für das jeweilige Modul erlaubt sind. In allen Fällen sind die Aufträge **Break**, **Disable** und **Enable** enthalten. **Break** beendet einen Auftrag vorzeitig und schaltet alle Motoren des Moduls ab, z.B. in einer Not-Aus-Situation. **Disable** unterbricht einen Auftrag. Die Motoren werden abgeschaltet, der Auftrag bleibt jedoch als noch nicht erledigt vermerkt. Der Auftrag wird durch **Enable** wieder aktiviert. **Disable** und **Enable** können dazu verwendet werden, um Aufträge miteinander zu synchronisieren.
Die Variable **Schritt** numeriert die Prozeßschritte für das betreffende Modul. Jede Erledigung eines Auftrags erhöht die Variable **Schritt**.
Manche Aufträge, z.B. Einschalten eines Werkzeugs, sind sofort nach Beendigung der Prozedur abgeschlossen. In diesen Fällen ist die Variable **Schritt** sofort erhöht.
Andere Aufträge erfordern ein Warten, z.B. bis ein Endtaster durch das bewegte Maschinenteil betätigt wird. In diesem Fall kehrt die Programmkontrolle sofort wieder aus der Steuerungsprozedur zurück; die Variable **Schritt** ist jedoch i.a. noch nicht erhöht. Das übergeordnete Steuerungsprogramm muß den Aufruf solange wiederholen, bis der Auftrag beendet ist, erkennbar an der Erhöhung der Variablen **Schritt**.
Auf diese Weise besteht die Möglichkeit parallele Prozesse, d.h. gleichzeitig ablaufende Bewegungen zu programmieren. Dabei werden zwei oder mehr Aufträge gestartet. In zyklischer Abfrage wird jeder Aufruf wiederholt, bis er aus der Überwachung entlassen werden kann. Parallele Prozesse erfordern selbstverständlich getrennte Variablen, z.B. **Schritt_A** und **Schritt_B**. Beispiel (Programmausschnitt):

```
Schritt_A := 5;   Schritt_B := 2;
Repeat
      If Schritt_A = 5 Then MS_1(3,Minus_Y,Schritt_A);                    {Maschinenständer 3 vorfahren}
      If Schritt_B = 2 Then MT_1WKSXfd(2,Tisch_drehen,Schritt_B); {Maschinentisch 2 gleichzeitig drehen}
Until (Schritt_A = 6) AND (Schritt_B = 3);                                {Beide Aufträge beendet}
```

Nur strikt nacheinander ablaufende Prozeßschritte können mit einer einzigen Variablen durchnumeriert werden.
Das Softwarekonzept der Maschinenmodule sollte auf der Ebene der Maschinen wiederholt werden. In der Datei SOFTMOD.INC sind die entsprechenden Datenstrukturen für 50 Maschinen bereits vorbereitet. Die Schrittvariablen sind z.B. als **Maschinenspeicher[n].PC** definiert.
Das Programmbeispiel zeigt die Vorgehensweise.



In dem Beispiel wird eine Anlage gesteuert, die aus zwei Maschinen besteht. Maschine 1 ist eine Bearbeitungsmaschine bestehend aus einem Maschinenbett MB/1, einem Maschinenständer MS/1, einem Revolverkopf WSS/R und einem vorgelagerten Transportband FB/135mm. Die Maschine 2 besteht aus einem Drehtisch DT/135mm und einem Förderband FB/135mm.

Steuerungsaufgabe:

1. Referenzfahrt, um alle Maschinenteile in die Ausgangslage zu bringen:
   a) Werkzeug-Spindelstock der Maschine 1 in +Z-Richtung,
   b) Maschinenständer der Maschine 1 in +Y-Richtung,
   c) Werkzeugwechsel der Maschine 1, Werkzeug Nr. 1 eingeschwenkt.
   d) Drehtisch der Maschine 2 in +C-Richtung.
2. Maschine 1: Bearbeiten eines Werkstückes:
   a) Transportband der Maschine 1 positioniert Werkstück in Richtung +X,
   b) Maschinenständer in Richtung -Y fahren,
   c) Werkzeug Nr. 2 einschwenken,
   d) Bearbeiten (d.h. Werkzeugspindelstock erst in -Z- und dann in +Z-Richtung) mit Werkzeug Nr. 2,
   e) Werkzeug Nr. 3 einschwenken,
   f) Bearbeiten mit Werkzeug Nr. 3,
   g) Werkzeug Nr. 1 einschwenken,
   h) Maschinenständer in Richtung +Y fahren,
   i) Mitteilung an Maschine 2, daß Werkstück abholbereit ist.
3. Maschine 2: Abtransport eines Werkstücks:
   a) Werkstück auf Drehtisch in +X-Richtung positionieren,
   b) Mitteilung an Maschine 1, daß Werkstück übernommen wurde,
   c) Drehtisch in -C-Richtung drehen,
   d) Werstück auf Förderband übernehmen,
   e) Drehtisch-Förderketten anhalten,
   f) Drehtisch in +C-Richtung schwenken.

Die in dem Beispiel verwendeten Richtungsfestlegungen entsprechen dem Standard und liegen allen Datenblättern zugrunde.

```
PROGRAM Demo1;
USES CRT,DOS,PRINTER;
{$I INDTREIB.INC }
{$I SOFTMOD.INC  }
{$I DEMO1.PIN    }

PROCEDURE Init;
VAR i : INTEGER;
  BEGIN
    FOR i := 1 TO 2 DO
    WITH Maschinenspeicher(i) DO
      BEGIN
        Zustand := ready;
        Ladung:= nichts;
        PC      := 1;
      END;
  END;
FUNCTION NOTAUS : BOOLEAN;
  BEGIN
    NOTAUS := KEYPRESSED;
  END;

PROCEDURE Referenzfahrt;
VAR PC : INTEGER;
  BEGIN { Referenzfahrt }
    PC := 1;
    REPEAT
      CASE PC OF
        1 : WSS_R(1,plus_Z,PC);
        2 : WSS_R(1,Werkzeug_ein,PC);
        3 : WSS_R(1,Werkzeugwechsel,PC);
        4 : WSS_R(1,Werkzeug_aus,PC);
        5 : MS_T(1,plus_Y,PC);
        6 : DT_T35(1,Tisch_drehen_plus,PC);
      END; { CASE }
    UNTIL NOTAUS OR (PC = 7);
  END; { Referenzfahrt }

PROCEDURE Maschine_1;
  BEGIN
    WITH Maschinenspeicher(1) DO
      BEGIN
        CASE PC OF
          1
FB_135(1,positionieren_plus_X,PC);
          2 : BEGIN
                Ladung := Neuteil;
                PC := 3;
              END;
          3 : MS_1(1,minus_Y,PC);
          4 : WSS_R(1,Werkzeug_ein,PC);
          5 : WSS_R(1,Werkzeugwechsel,PC);
          6 : WSS_R(1,minus_Z,PC);
          7 : WSS_R(1,plus_Z,PC);
          8 : WSS_R(1,Werkzeugwechsel,PC);
          9 : WSS_R(1,minus_Z,PC);
          10: WSS_R(1,plus_Z,PC);
          11: WSS_R(1,Werkzeugwechsel,PC);
          12: WSS_R(1,Werkzeug_aus,PC);
          13: MS_T(1,plus_Y,PC);
          14: IF Maschinenspeicher(2).Ladung =
              nichts
                THEN BEGIN
                  Maschinenspeicher(2).Ladung :=
                  Neuteil;
                  FB_135(1,plus_X,PC);
                  PC := 15;
                END;
          15: IF Ladung = nichts THEN
                BEGIN
                  PC := 1;
                  FB_135(1,BREAK,PC)
                END;
        END; { CASE }
      END; { WITH Maschinenspeicher(1) }
  END; { Maschine_1 }

PROCEDURE Maschine_2;
  BEGIN
    WITH Maschinenspeicher(2) DO
      BEGIN
        CASE PC OF
          1 : IF Ladung = Neuteil THEN PC := 2;
          2 : DT_135(1,positionieren_plus_X,PC);
          3 : BEGIN
                Maschinenspeicher(1).Ladung :=
                nichts;
                PC := 4;
              END;
          4 : DT_135(1,Tisch_drehen_minus,PC);
          5 : DT_135(1,plus_X,PC);
          6 : FB_135(2,positionieren_plus_X,PC);
          7 : DT_135(1,BREAK,PC);
          8 : DT_135(1,Tisch_drehen_plus,PC);
          9 : BEGIN
                Ladung := nichts;
                PC := 1;
              END;
        END; { CASE }
      END; { WITH Maschinenspeicher(2) }
  END; { Maschine_2 }

BEGIN { HP }
  CLRSCR;
  Interface_initialisieren;
  Pinbelegung;
  Init;
  Referenzfahrt;
  REPEAT
    Maschine_1;
    Maschine_2;
  UNTIL NOTAUS;
  Interface_initialisieren;
  CLRSCR;
END. { HP }
```

Inhalt der Modul-Diskette

Die Modul-Diskette enthält folgende Dateien:

INDTREIB.INC
: Pascal-Includedatei mit den Treiberroutinen für das Industrie-Interface.

SOFTMOD.INC
: Pascal-Includedateien mit den Softwaremodulen.

LISTE.DOK
: Textdatei; enthält die Beschreibung zur Erstellung der Datei mit der Verbindungsliste

DEMO1.PAS
: Umseitiges Beispielprogramm.

DEMO1.PIN
: Zu DEMO1.PAS gehörige Verbindungsliste.

DEMO2.PAS
: Ein größeres Programmbeispiel.

DEMO2.PIN
: Zu DEMO2.PAS gehörige Verbindungsliste.

INDTREIB.BAS
: BASIC-Datei mit den Unterprogrammen zur Steuerung des Interface. Die BASIC-Datei ist im ASCII-Format gespeichert, so daß sie mit dem MERGE-Kommando in bestehende Programme eingebunden werden kann.
Dabei muß darauf geachtet werden, daß das Steuerungsprogramm keine Zeilennummern unter 10000 enthält, da sie sonst durch die Programmzeilen von INDTREIB.BAS überlagert und damit gelöscht werden könnten.

Inhalt der Ergänzungsdiskette

Die Ergänzungsdiskette enthält das Programm LISDDIAG zum Test der verwendeten Module. Am Bildschirm erscheint erscheint eine symbolische Darstellung des Baugruppenträgers. Durch interaktive Menübedienung wird dem Programm die Bestückung mit Interfacemodulen mitgeteilt. Die gewählten Module werden am Bildschirm lagerichtig in den Baugruppenträger eingetragen. Die Eingänge bzw. Ausgänge werden mit Leuchtdioden dargestellt, an denen der Schaltzustand abgelesen werden kann.

Im nächsten Schritt werden die verwendeten Maschinenmodule benannt. Das Programm fordert auf, die Verbindungen der Motoren, Schalter, Initiatoren usw. zu den Ein- und Ausgängen der Interfacemodule zu benennen.

Im nächsten Schritt kann das Maschinenmodul interaktiv gesteuert werden. Auf diese Weise kann die Korrektheit der Verkablung, Drehsinns von Motoren usw. überprüft werden. Die Verbindungen jederzeit korrigiert werden.

Im letzten Schritt wird eine Verbindungsliste in Form einer Textdatei erzeugt. Die Verbindungsliste wird als Include-Datei in das Steuerungsprogramm eingebunden. Auf diese Weise wird die Beziehung zwischen den Softwaremodulen und den Maschinenmodulen über die Interfacemodule hergestellt.