Applikationsbeispiel: Automatisierung einer Registerstanzmaschine mit NanoJ

Für einen Kunden, der mit einer SMCI47-S-2 und Nanopro den Vorschub des Stanzmessers in einer Registerstanzmaschine steuerte, sollte die Lösung so automatisiert werden, dass eine in der Anlage integrierte SPS die einzelnen Stanzpositionen anfahren kann. Der Teach-in für einen neuen Auftrag soll weiterhin über einen Laptop erfolgen, die Funktion der Ein- und Ausgänge in der SMCI47-S wurde aber mit NanoJ so programmiert, dass das Programm in der übergeordneten SPS sehr einfach gestaltet werden konnte.
Für die Registerstanzung war es notwendig, dass bis zu 31 einzelne Registerpositionen, mit jeweils einzeln einstellbarer Länge angefahren werden können. Diese Positionen werden vom Maschineneinrichter über Nanopro in einzelnen Positioniersätzen der Steuerung gespeichert.
Über fünf digitale Eingänge der SMCI47-S wird der Vorschub der Stanze durch die SPS gesteuert, am sechsten Eingang der Steuerung wird der Referenzschalter angeschlossen:

EINGANG	
1-	„Not-Aus“
2-	„nächste Position“
3-	„Referenzfahrt starten“
4-	„zur letzten Position zurückfahren“
5-	„Reset auf Zielposition 1“
6-	Referenzschalter

Nach dem Einschalten der Maschine wird von der SPS über Eingang 3 zuerst eine Referenzfahrt auf den an Eingang 6 angeschlossenen Referenzschalter durchgeführt. Anschließend wird über ein Signal auf Eingang 2 die nächste Stanzposition angefahren. Falls der Bediener eine unsaubere Stanzung bemerkt, kann er über Eingang 4 jeweils eine Position zurückfahren, und die Stanzung wiederholen.
Sind alle Register eines Buches gestanzt, wird über Eingang 5 der Zähler wieder auf die erste Registerposition zurückgesetzt. Falls der Bediener die Not-Aus-Taste an der Maschine drückt, wird der Motor über Eingang 1 kontrolliert gestoppt.

Abgesichert gegen Ausschuß durch Encoderüberwachung
Als Motor wurde ein AP8918M6404-E in Schutzart IP65 gewählt, dem der beim Stanzvorgang anfallende Staub nichts anhaben kann. Der im Motor integrierte Drehgeber wird durch die SMCI47-S überwacht, und stellt sicher, dass die Steuerung z.B. bei einer mechanischen Schwergängigkeit oder Blockade einen Fehler meldet, um zu verhindern, dass die Register an der falschen Position angebracht werden.

```
1   import nanotec.io;                                          // verwendete Includes einbinden//
2   import nanotec.util;
3   import nanotec.drive;
4
5   class neu_kommentiert
6   {
7       public static void main ()                              //Hauptprogramm aufrufen//
8       {
9           int satz = 1;                                       //Zähler für die Fahrprofilwahl definieren//
10
11          while(true){                                        //Endlosschleife starten//
12
13          //Funktion für Eingang 2, Schritt starten und Nächsten laden//
14              if(( io.GetDigitalInput () == 2)&&(satz==32 ))  //Wenn Eingang 2 aktiviert und
15              {                                               //Zähler gleich 32, dann...//
16                  drive.LoadDataSet(satz);                    //Datensatz entsprechend Zähler aus EEPROM laden//
17                  drive.StartDrive();                         //Fahrprofil starten//
18                  satz =1;                                    //Wert von Zähler auf 1 setzen//
19                  while(io.GetDigitalInput () == 2){};        //Warten bis Eingang 2 nicht mehr aktiv//
20                  util.Sleep(200);                            //zeitliche Sicherheit (Endprellzeit)//
21              }
22
23          //Funktion für Eingang 1, Not-Aus//
24              if( io.GetDigitalInput( ) == 1)                 //Wenn Eingang 1 aktiviert, dann...//
25              {
26                  drive.StopDrive(0);                         //Fahrt sofort stoppen//
27                  while(io.GetDigitalInput () == 1){};        //Warten bis Eingang 1 nicht mehr aktiv//
28                  util.Sleep(200);                            //zeitliche Sicherheit (Endprellzeit)//
29              }
30
31          //Funktion für Eingang 2 wenn letzter Schritt erreicht, Schritt starten und Schritt 1 laden//
32              if(( io.GetDigitalInput () == 2)&&(satz!=32 ))  //Wenn Eingang 2 aktiviert und
33              {                                               //Zähler ungleich 32, dann...//
34                  drive.LoadDataSet(satz);                    //Datensatz entsprechend Zähler aus EEPROM laden//
35                  drive.StartDrive();                         //Fahrprofil starten//
36                  satz =satz+1;                               //Wert von Zähler um 1 vergrößert//
37                  while(io.GetDigitalInput () == 2){};        //Warten bis Eingang 2 nicht mehr aktiv//
38                  util.Sleep(200);                            //zeitliche Sicherheit (Endprellzeit)//
39              }
40
41          //Funktion für Eingang 5, wieder auf Schritt 1 stellen (RESET)//
42              if( io.GetDigitalInput( ) == 16)                //Wenn Eingang 5 aktiviert, dann...//
43              {
```