

Dokumentation

# Automationsprojekt 2008

Transferprozess

Verfasser:  
Benno Jung  
Martin Züger

Dozent:  
Günter Nagel

Dezember 2008

Diese Projektdokumentation wurden im Verlauf des Herbstsemesters 2008 von **Benno Jung** und **Martin Züger** erstellt.

Dieses Dokument darf unter den Bedingungen der [Creative Commons Attribution-Noncommercial-Share Alike 2.5 Switzerland Lizenz](#) kopiert, verändert und weitergegeben werden.

Das heisst Sie dürfen:



das Werk vervielfältigen, verbreiten und öffentlich zugänglich machen



Bearbeitungen des Werkes anfertigen

Zu den folgenden Bedingungen:



**Namensnennung.** Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen (wodurch aber nicht der Eindruck entstehen darf, Sie oder die Nutzung des Werkes durch Sie würden entlohnt).



**Keine kommerzielle Nutzung.** Dieses Werk darf nicht für kommerzielle Zwecke verwendet werden.



**Weitergabe unter gleichen Bedingungen.** Wenn Sie dieses Werk bearbeiten oder in anderer Weise umgestalten, verändern oder als Grundlage für ein anderes Werk verwenden, dürfen Sie das neu entstandene Werk nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch oder vergleichbar sind.

## Bewertung durch Prof. Günter Nagel:



Automation SPS: Bewertung Projekt

### Bewertung Automation SPS-Projekt:

Für Studentengruppe: Benno Jung – Martin Züger

Aufgabe	Fachliche Beurteilung / Kommentar	Note
<b>1. Konzeption SPS-Programm</b> <ul style="list-style-type: none"> <li>SW-Konzept, SW-Struktur</li> <li>Zustandsdiagramm</li> <li>Bedienkonzept – HMI</li> <li>Fehlerbehandlung</li> </ul>	Gutes SW-Konzept und gute SW-Struktur, modularer Aufbau der SW Zustandsdiagramm in Ordnung. Gutes Bedienkonzept (Galileo) Gute und klare Fehlerbehandlung.	6
<b>2. Ausführung</b> <ul style="list-style-type: none"> <li>Funktionsfähigkeit SPS-Programm</li> <li>Selbständigkeit</li> </ul>	SW läuft korrekt ab. Fehlerfall funktioniert einwandfrei. Sinus-Förmige Rampe implementiert.	6
<b>3. Dokumentation</b> <ul style="list-style-type: none"> <li>Formal (Inhaltsverzeichnis, ...)</li> <li>Zustandsdiagramm</li> <li>I/O-Liste</li> <li>Bedienung</li> <li>SW-Dokumentation (Listing)</li> <li>Nachvollziehbarkeit</li> </ul>	Zusammenfassung fehlt, sonst formal ok. Anlagenübersicht Gesamtsystem, SW-Übersicht und SW-Struktur wurden dokumentiert. I/O-Liste vorhanden. Gutes Zustandsdiagramm. Gute Beschreibung HMI. Gute Beschreibung der Bedienung im Fehlerfall. Gute SW-Dokumentation in Programm.	5.5
<b>Gesamtnote</b>		<b>5.8</b>

Allgemeine Bemerkungen:

## Inhaltsverzeichnis

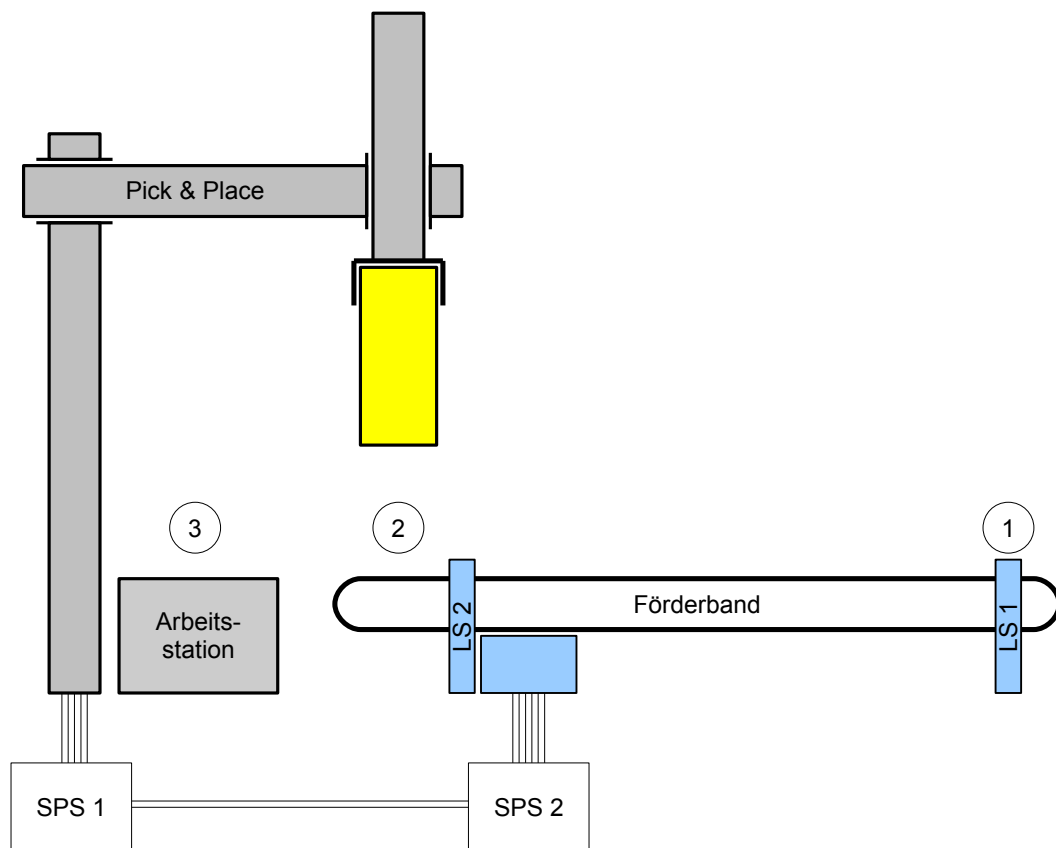
<b>1 Ziel</b>	<b>3</b>
<b>2 Aufbau und Ablauf</b>	<b>3</b>
<b>3 Förderband</b>	<b>4</b>
3.1 Zustandsdiagramm . . . . .	4
3.2 Softwarestruktur . . . . .	5
3.3 Fehlerbehandlung . . . . .	5
3.4 Hardware-Belegung . . . . .	6
3.4.1 Eingänge . . . . .	6
3.4.2 Ausgänge . . . . .	6
3.5 Programmcode . . . . .	6
3.6 Bedienoberfläche . . . . .	9
<b>4 Pick and Place</b>	<b>10</b>
4.1 Zustandsdiagramm . . . . .	10
4.2 Softwarestruktur . . . . .	10
4.3 Fehlerbehandlung . . . . .	11
4.4 Hardware-Belegung . . . . .	11
4.4.1 Eingänge . . . . .	11
4.4.2 Ausgänge . . . . .	11
4.5 Programmcode . . . . .	12
4.6 Bedienoberfläche . . . . .	15

## 1 Ziel

Ziel dieses Projektes ist ein Förderbandsystem mit einem Pick-And-Place-Gerät zu verbinden. Dazu sollen zwei SPS-Programme mit grafischer Bedienoberfläche erstellt werden. Das zu handhabende Objekt ist ein Aluminium Klotz mit den Abmassen 50x50x100 mm. Bei beiden Steuerungen handelt es sich um eine xSystem SPS der Firma Micro Innovation. Die Aufgabe wurde wie folgt aufgeteilt:

- Benno Jung: Förderband
- Martin Züger: Pick-And-Place-Gerät

## 2 Aufbau und Ablauf



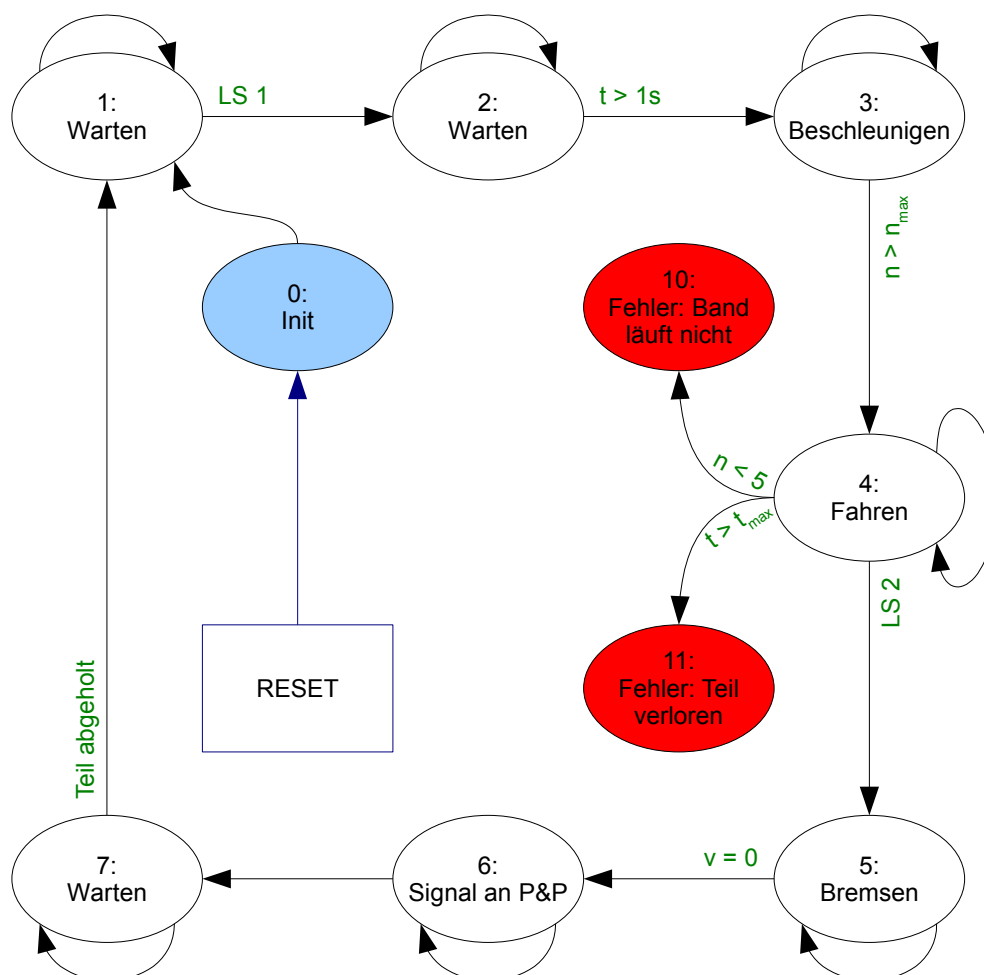
Das Objekt wird bei Position ① auf das Förderband gelegt. Erkennt die Steuerung den Klotz mit Hilfe der Lichtschranke, so wird das Förderband in Bewegung versetzt. Am Ende des Bandes (etwas rechts von Pos. ②) durchfährt der Alu-Klotz wieder eine Lichtschranke und wird anschliessend unter dem Greifer des P&P-Gerätes positioniert. Dieser Greift sich das Objekt und legt es in die Arbeitsstation ③ ein.

### 3 Förderband

Sobald ein Objekt auf das Förderband gelegt wird, wird dieses mittels Lichtschranke erkannt. Nach einer kurzen Wartezeit beschleunigt das Band S-Rampenförmig auf die maximale Bandlaufgeschwindigkeit. Die S-Rampe ist nötig damit es keine sprunghaften Geschwindigkeitsänderungen gibt.

Sobald das Werkstück auf der anderen Seite des Bandes mit der zweiten Lichtschranke detektiert wird, bremst das Band wieder S-Rampenförmig ab. Nun gibt die Bandsteuerung dem Pick and Place Gerät die Information das ein Objekt abholbereit ist. Sobald das Objekt vom Pick and Place Gerät weggehoben wurde gibt dieses ein Signal zurück und das Förderband geht wieder in den Anfangszustand über.

#### 3.1 Zustandsdiagramm



### 3.2 Softwarestruktur

Die Software für das Förderband wurde in ST geschrieben und in drei Funktionsblöcke unterteilt:

1. **Ereignisse**

In diesem Funktionsblock werden die Sensoren ausgewertet und der nächste Zustand berechnet.

2. **Aktionen**

Hier werden die Ausgänge gesetzt.

3. **Timer**

Der Timer erzeugt ein 20 ms-Takt welcher für verschiedene Aufgaben verwendet wird. Dazu gehören ein Time-Out, Verzögerung beim Start, Beschleunigung und Bremsen des Förderbandes.

### 3.3 Fehlerbehandlung

Mit dem Reset-Taster kann aus jedem beliebigen Zustand wieder in den INIT-Zustand gewechselt werden. Das wird hauptsächlich bei Fehlermeldungen nötig. Alle Ausgangssignale werden zurückgesetzt und die Maschine befindet sich danach wieder im funktionsbereiten Startzustand.

1. **Teil verloren:**

Wenn das eingelegte Objekt nach einer gewissen Zeit auf der Greiferseite noch nicht detektiert wurde, wird das Band abgebremst und eine Fehlermeldung ausgegeben. Es kann in diesem Fall davon ausgegangen werden dass entweder das Objekt verloren gegangen ist, vom Maschinenbediener entfernt wurde oder dass die linke Lichtschranke defekt ist. Das Band wird so abgebremst, dass bei einem Lichtschrankendefekt das Teil nicht über das Bandende hinaus befördert wird.

2. **Bandstillstand:**

Ist die Bandgeschwindigkeit nach der Beschleunigungsphase noch Null, so ist der Sensor zur Bandgeschwindigkeitsüberwachung defekt oder das Band hat sich verklemmt. Um Schäden am Motor oder der Anlage Vorzubeugen wird in diesem Fall das Band sofort gestoppt und eine entsprechende Fehlermeldung ausgegeben.

## 3.4 Hardware-Belegung

### 3.4.1 Eingänge

	Typ	Verwendung	Bemerkung
IW2	analog	aktuelle Geschwindigkeit	0..10 V
I0.0	digital	Lichtschranke rechts	high aktiv
I0.1	digital	Lichtschranke links	high aktiv

### 3.4.2 Ausgänge

	Typ	Verwendung	Bemerkung
QW2	analog	Geschwindigkeits Vorgabe	0..10 V
Q0.0	digital	Rückwärts	
Q0.1	digital	Vorwärts	

## 3.5 Programmcode

Listing 1: Globale Variablen

```

1  VAR_GLOBAL
2  t_max: INT := 120;           (* max. Zeit fuer Bandlaufzeit bis Sig_l kommen
   t_max: INT := 120;           muss *)
3  v_max: REAL :=120;         (* max. Geschwindigkeit *)
4
5  TFLAG: BOOL := FALSE;
6  v AT %QW2: WORD;           (* Theor. Geschwindigkeit *)
7  Count: INT := 0;
8  beschl_zeit: REAL := 20;   (* Beschleunigungszeit = beschl_zeit * 20ms *)
9
10 richtungl AT %QX0.1: BOOL:=TRUE; (* Richtung links *)
11 richtungr AT %QX0.0: BOOL:=FALSE; (* Richtung rechts *)
12 END_VAR

```

Listing 2: HMI Variablen

```

1  VAR_GLOBAL
2  Error_Stillstand: BOOL;
3  Error_Teilweg: BOOL;
4  Sig_r AT %IX0.0: BOOL;     (*Signal rechts Start*)
5  Sig_l AT %IX0.1: BOOL;     (*Signal links Start*)
6  Reset: BOOL := FALSE;
7  Zustand: INT := 0;
8  END_VAR

```

Listing 3: Hauptprogramm

```

1  VAR
2  F1: Ereignisse;
3  F2: Aktionen;
4  F3: Timer;

```

```

5 END_VAR
6
7 F1(enable := TRUE);
8 F2(enable := TRUE);
9 F3(enable := TRUE);

```

Listing 4: Funktionsblock Ereignisse

```

1 VAR
2   n AT %IW2: WORD;           (* Aktuelle Drehzahl *)
3   Sig_PP_finish AT %IX0.3: BOOL; (* Signal Teil vom Foerderband abgehoben *)
4 END_VAR
5
6 IF Reset THEN
7   Zustand := 0;
8 ELSE
9   CASE Zustand OF
10    0: Zustand:=1; (* Initialisierung *)
11
12    1: IF Sig_r THEN (* Warten bis Lichtschranke Rechts betaetigt wird *)
13       Count:=0;
14       Zustand:=2;
15       END_IF;
16
17    2: IF Count >= 40 THEN (* Warten auf Abfahrt *)
18       Zustand:=3;
19       Count:=0;
20       END_IF;
21
22    3: IF v >= v_max THEN (* Beschleunigen bis Reisegeschwindigkeit *)
23       Zustand:=4;
24       Count:=0;
25       END_IF;
26
27    4: IF Sig_l THEN (* Fahren bis Lichtschranke links betaetigt wird *)
28       Zustand:=5;
29       Count := 0;
30       ELSIF n <= 5 THEN (* Fehler wenn Band steht *)
31          Zustand:=10;
32       ELSIF Count > t_max THEN (* Fehler Zeitueberschreitung *)
33          Zustand:=11;
34          Count := 0;
35       END_IF;
36
37    5: IF v = 0 THEN (* Bremsen bis Stillstand *)
38       Zustand:=6;
39       END_IF;
40
41    6: IF Sig_PP_finish THEN
42       Zustand:=7; (* Signal an P and P *)
43       END_IF;
44
45    7: IF Sig_PP_finish THEN (* nach Signal wieder in Wartezustand wechseln *)
46       Zustand:=1;
47       END_IF;
48   END_CASE;
49 END_IF;

```

Listing 5: Funktionsblock Aktionen

```

1  VAR
2    Sig_PP_go AT %QX0.3: BOOL; (* Signal Teil abholbereit *)
3  END_VAR
4
5  CASE Zustand OF
6    0: (* INIT *)
7      v := 0;
8      Sig_PP_go := FALSE;
9      Error_Stillstand := FALSE;
10     Error_Teilweg := FALSE;
11
12     3: (* Beschleunigen *)
13     IF Count >= beschl_zeit THEN
14       v := REAL_TO_WORD(v_max);
15     ELSIF TFLAG THEN
16       v := REAL_TO_WORD(v_max*SIN(Count*1.57/beschl_zeit)*SIN(Count*1.57/beschl_zeit)
17         );
18     END_IF;
19
20     5: (* Bremsen *)
21     IF Count >= beschl_zeit THEN
22       v := 0;
23       Count := 0;
24     ELSIF TFLAG THEN
25       v := REAL_TO_WORD(v_max*SIN(1.57+Count*1.57/beschl_zeit)*SIN(1.57+Count*1.57/
26         beschl_zeit));
27     END_IF;
28
29     6: (* Signal an PandP*)
30     Sig_PP_go := TRUE;
31
32     7: (* Warten *)
33     Sig_PP_go:=FALSE;
34
35     10: (* Fehler: Stillstand *)
36     v := 0;
37     Error_Stillstand := TRUE;
38
39     11: (* Fehler: Teil weg *)
40     IF Count*2 >= beschl_zeit THEN
41       v := 0;
42     ELSIF TFLAG THEN
43       v := REAL_TO_WORD(v_max*SIN(1.57+Count*3.14/beschl_zeit)*SIN(1.57+Count*3.14/
44         beschl_zeit));
45     END_IF;
46     Error_Teilweg := TRUE;
47  END_CASE;

```

Listing 6: Funktionsblock Timer

```

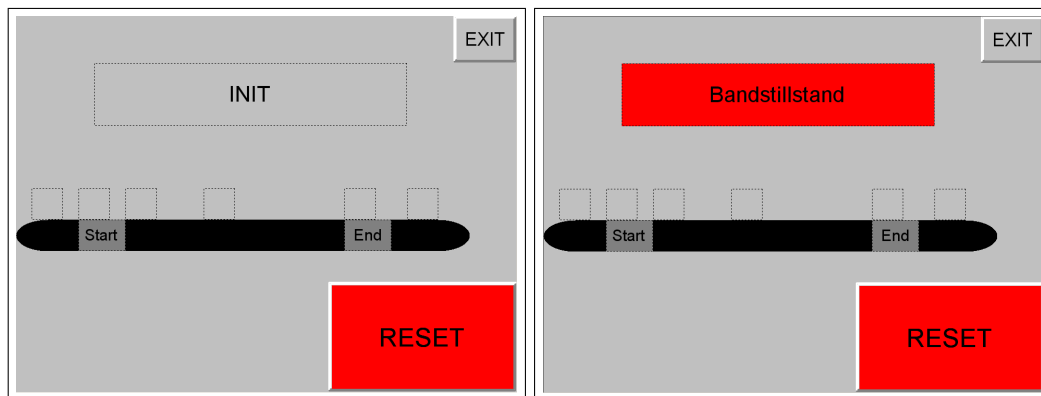
1  VAR
2    TON1: TON;
3    endTimer1: BOOL;
4    Takt: TON;
5  END_VAR
6
7  (* Taktgenerator *)
8  Takt(IN := NOT TFLAG, PT := T#20ms);

```

```
9 TFLAG := Takt.Q;  
10  
11 IF TFLAG THEN  
12     Count := Count +1 ;  
13 END_IF;
```

### 3.6 Bedienoberfläche

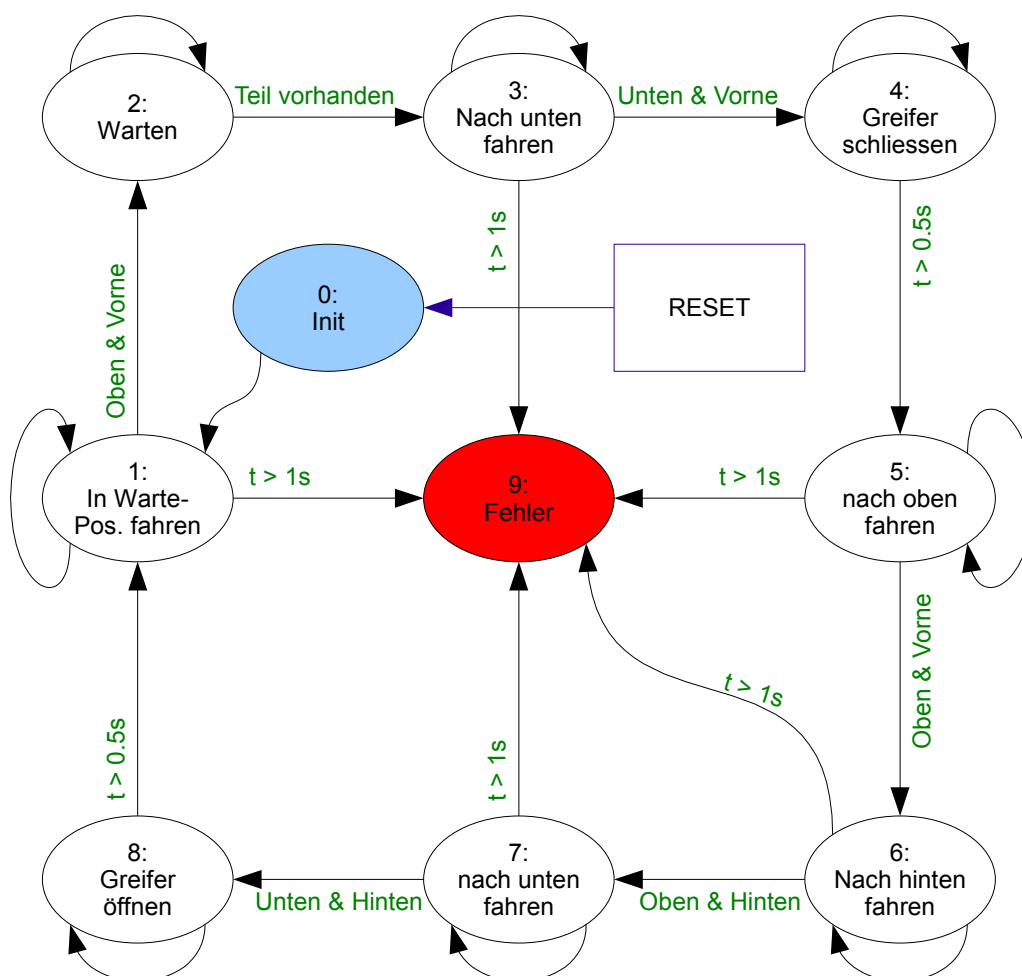
Die Bedienoberfläche wurde mit der Software *Galileo* von *Micro Innovation* erstellt. In der mittleren Box wird der aktuelle Status angezeigt: Was gerade läuft, bzw. was für ein Fehler ist aufgetreten. Der Aluminium-Klotz wird mit den kleinen Boxen auf dem Förderband symbolisch dargestellt. Diese werden je nach Position des Objektes eingefärbt. Im untenstehenden Bild ist links das HMI beim Start gezeigt. Rechts ist der Fehlerfall zu sehen, wenn das Band stillsteht.



## 4 Pick and Place

Ist der Aluminium-Klotz am Ende des des Förderbandes angekommen, so erhält die P&P-Steuerung ein Signal von der Förderbandsteuerung. Der Greifer fährt darauf hin senkrecht nach unten und greift das Objekt. Nach dem Anheben des Teils fährt der Arm nach hinten und legt es auf die Arbeitsstation. Die Steuerung des Förderbandes erhält das Signal "Teil abgeholt".

### 4.1 Zustandsdiagramm



### 4.2 Softwarestruktur

Die Software für das Pick-And-Place-Gerät wurde in ST geschrieben und in zwei Funktionsblöcken unterteilt:

### 1. Events

In diesem Funktionsblock werden die Endschalter ausgewertet und der nächste Zustand berechnet. Weiter beinhaltet dieser Funktionsblock die Resetbedingung sowie die Timeout-Funktion.

### 2. Actions

Hier werden die Ausgänge gesetzt.

## 4.3 Fehlerbehandlung

Für die Fehlerbehebung ist eine TimeOut-Funktion implementiert worden. Diese wird bei jeder Bewegung gestartet und wenn der Hub nach einer Sekunde nicht am anderen Endanschlag angekommen ist wird in den Fehler-Zustand gewechselt. Anschliessend wird auf dem HMI angezeigt ob der Fehler in der horizontalen oder in der vertikalen Bewegungsrichtung aufgetreten ist. Nach beheben des Fehlers kann durch Drücken von Reset das Programm wieder neu gestartet werden. Dabei wird der Greifer in die Ursprungsposition gefahren und geöffnet. Ist dies ohne Fehler möglich, so wird er automatisch wieder in die Warteposition gefahren.

## 4.4 Hardware-Belegung

### 4.4.1 Eingänge

	Typ	Verwendung	Bemerkung
I0.0	digital	Horizontalbewegung hinten	
I0.1	digital	Horizontalbewegung vorne (kurz)	nicht benötigt
I0.2	digital	Horizontalbewegung vorne (lang)	
I0.3	digital	Teil vorhanden	Signal von SPS 2
I0.4	digital	Vertikalbewegung oben	
I0.5	digital	Vertikalbewegung unten (kurz)	nicht benötigt
I0.6	digital	Vertikalbewegung unten (lang)	

### 4.4.2 Ausgänge

	Typ	Verwendung	Bemerkung
Q0.0	digital	Horizontalbewegung (ausfahren)	
Q0.1	digital	Hubumschaltung horizontal	aktiv = langer Hub
Q0.2	digital	Vertikalbewegung (ausfahren)	
Q0.3	digital	Teil abgeholt	Signal an SPS 2
Q0.4	digital	Hubumschaltung vertikal	aktiv = langer Hub
Q0.5	digital	Greifer	aktiv = Greifer offen

## 4.5 Programmcode

Listing 7: HMI Variablen

```

1  VAR_GLOBAL
2  state : INT := 0; (* State 0 = Init *)
3  reset : BOOL := FALSE;
4  error : BOOL := FALSE;
5  eCode : INT := 0;
6  sig_pa AT %IX0.3 : BOOL;
7  posDown AT %IX0.6 : BOOL;
8  posUp AT %IX0.4 : BOOL;
9  posFront AT %IX0.2 : BOOL;
10 posBack AT %IX0.0 : BOOL;
11 END_VAR

```

Listing 8: Hauptprogramm

```

1  PROGRAM PLC_PRG
2
3  VAR
4  FBe : Events;
5  FBa : Actions;
6  END_VAR
7
8
9  (* Function block Actions *)
10 FBa(enable := TRUE);
11
12 (* Function block Events *)
13 FBe(enable := TRUE);

```

Listing 9: Funktionsblock Events

```

1  FUNCTION_BLOCK Events
2
3  VAR_INPUT
4  enable : BOOL;
5  END_VAR
6
7  VAR
8  timeOut      : TON;
9  delay        : TON;
10 timeOut_run  : BOOL := FALSE;
11 delay_run    : BOOL;
12 END_VAR
13
14
15 (* Reset *)
16 IF reset THEN
17   state := 0;
18 END_IF;
19
20 (* Time-Out *)
21 timeOut(IN := timeOut_run, PT := T#1s);
22 IF timeOut.Q THEN
23   error := TRUE;
24   state := 9;
25   timeOut.IN := FALSE;

```

```
26 END_IF;
27
28 (* Delay for picker *)
29 delay(IN := delay_run, PT := T#0.5s);
30
31 (* States *)
32 CASE state OF
33   0 : (* Init *)
34     error := FALSE;
35     eCode := 0;
36     timeOut_run := TRUE;
37     IF posBack AND posUp THEN
38       state := 1;
39       timeOut_run := FALSE;
40     END_IF;
41   1 : (* Go to pos. A *)
42     timeOut_run := TRUE;
43     IF posFront AND posUp THEN
44       state := 2;
45       timeOut_run := FALSE;
46     END_IF;
47
48   2 : (* Wait for Signal "piece available" *)
49     IF sig_pa THEN
50       state := 3;
51     END_IF;
52
53   3 : (* Go down to pos. B *)
54     timeOut_run := TRUE;
55     eCode := 3;
56     IF posFront AND posDown THEN
57       state := 4;
58       timeOut_run := FALSE;
59     END_IF;
60
61   4 : (* Close picker *)
62     delay_run := TRUE;
63     IF delay.Q THEN
64       delay_run := FALSE;
65       state := 5;
66     END_IF;
67
68   5 : (* Go up to pos. A *)
69     timeOut_run := TRUE;
70     eCode := 3;
71     IF posFront AND posUp THEN
72       state := 6;
73       timeOut_run := FALSE;
74     END_IF;
75
76   6 : (* Go back to pos. C *)
77     timeOut_run := TRUE;
78     eCode := 2;
79     IF posBack AND posUp THEN
80       state := 7;
81       timeOut_run := FALSE;
82     END_IF;
83
84   7 : (* Go down to pos. D *)
```

```

85     timeOut_run := TRUE;
86     eCode := 3;
87     IF posBack AND posDown THEN
88         state := 8;
89         timeOut_run := FALSE;
90     END_IF;
91
92     8 : (* Open picker *)
93     delay_run := TRUE;
94     IF delay.Q THEN
95         delay_run := FALSE;
96         state := 1;
97     END_IF;
98
99     9 : (* Error *)
100    timeOut_run := FALSE;
101    error := TRUE;
102
103    ELSE
104        error := TRUE;
105        eCode := 1;
106    END_CASE;

```

Listing 10: Funktionsblock Actions

```

1  FUNCTION_BLOCK Actions
2
3  VAR_INPUT
4      enable : BOOL;
5  END_VAR
6
7  VAR
8      rangeH AT %QX0.1 : BOOL := TRUE;
9      rangeV AT %QX0.4 : BOOL := TRUE;
10     movH   AT %QX0.0 : BOOL := FALSE;
11     movV   AT %QX0.2 : BOOL := FALSE;
12     picker AT %QX0.5 : BOOL := TRUE;
13     sig_pt AT %QX0.3 : BOOL := FALSE;
14 END_VAR
15
16 (* States *)
17 CASE state OF
18     0 : (* Init *)
19         rangeH := TRUE;
20         rangeV := TRUE;
21         movH := FALSE;
22         movV := FALSE;
23         picker := TRUE;
24         sig_pt := FALSE;
25
26     1 : (* Go to pos. A*)
27         movH := TRUE;
28         movV := FALSE;
29
30     2 : (* Wait for Signal "piece available" *)
31         ;
32
33     3 : (* Go down to pos. B *)
34         movV := TRUE;

```

```
35
36 4 : (* Close picker *)
37     picker := FALSE;
38
39 5 : (* Go up to pos. A *)
40     movV := FALSE;
41     sig_pt := TRUE;
42
43 6 : (* Go back to pos. C *)
44     movH := FALSE;
45
46 7 : (* Go down to pos. D *)
47     movV := TRUE;
48
49 8 : (* Open picker *)
50     picker := TRUE;
51     sig_pt := FALSE;
52 9 : (* Error *)
53     ;
54
55 END_CASE;
```

## 4.6 Bedienoberfläche

Auch die Bedienoberfläche für das Pick-And-Place-Gerät wurde mit der Software Galileo von Micro Inovation erstellt. Während des Betriebs zeigt die Grafik den aktuellen Zustand des Gerätes an. In der Infozeile wird angezeigt ob ein Teil vorganden ist oder ob sich die Anlage im Wartemodus befindet. Tritt ein Fehler auf, so erscheint eine rote Meldung in der Mitte des HMI und informiert den Bediener darüber was passiert ist.

