Zusammenfassung IAUT

Eric Lindegger – HS25

Danke fürs Herunterladen und Verwenden meiner IAUT Zusammenfassung! Hoffe ihr findet sie nützlich. Falls ihr Fehler findet oder Inputs habt, schreibt mir doch auf eric.lindegger@stud.hslu.ch

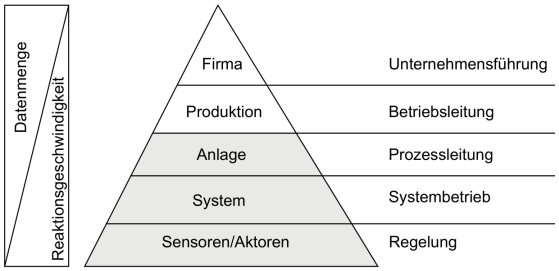
Inhaltsverzeichnis

1 Automatisierung

Definition Automatisierung

Unter Automatisierung (DIN19222) versteht man das gezielte Ausrüsten einer Einrichtung, so dass sie ganz oder teilweise ohne Mitwirkung des Menschen geschieht und arbeiten kann.

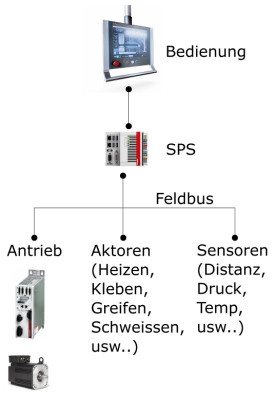
Die Automatisierungspyramide ist der Versuch die gesamte Automatisierungskette auf ein Bild abzubilden:



Automatisierungspyramid

Die Darstellung ist wie folgt zu interpretieren: Je weiter unten, desto schneller müssen die Komponenten reagieren und je weiter oben, ist die Datenmenge grösser. Die Absicht der Automatisierung ist, dass die automatisierte Anlage bis ins ERP der Firma eingepflegt wird. So muss z.B. keine spezielle Benutzeroberfläche geladen werden um ein neuer Prozess auf der Anlage zu starten.

1.1 Steuerung - Überblick



Die Bedienung ist wo das GUI (Graphical User Interface) dargestellt wird. Hier werden vom Benutzer Eingaben getätigt, wie z.B. Job oder Rezeptwahl.
Der Controller innerhalb der Steuerung/Anlage ist die CPU/SPS. Eine Automatisierung ist häufig eine (PID-)Regelungsaufgabe. Die SPS ist mit Aktoren und Sensoren verbunden:

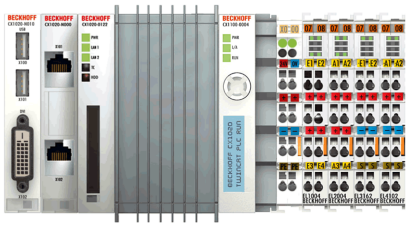
- Sensoren liefern Informationen
- Aktoren lösen Aktion aus

Die Schnittstellen der Sensoren/Aktoren ist entweder eine Spannung oder einen Strom. Die Spannung ist jedoch eher weniger gängig im Moment, da die Spannung anfälliger auf Störeinflüsse ist als ein Strom. Die gängigsten Schnittstellen sind. Ein weiterer Vorteil der Stromschnittstelle ist, dass auch ein Kabelbruch detektiert werden kann, da der Strom nie 0 wird:

- $0V - 10V$
- $4mA - 20mA$

1.1.1 Klemmen

Die Klemmen stellen die Ein- und Ausgänge einer SPS dar und können auf der Feldebene der SPS über ein Bussystem erweitert werden:



Klemmen können über RS485 abgesetzt werden und an einem anderen Ort als direkt bei der SPS eingesetzt werden.

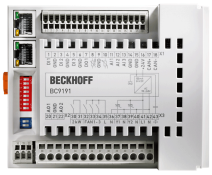
1.2 Allgemeine Aufgaben

Echtzeitanforderung	Regler Prozessabläufen usw.
keine Echtzeitanforderung	Mensch-Maschine-Interface (HMI) Produktionsplanung Archivierung usw.

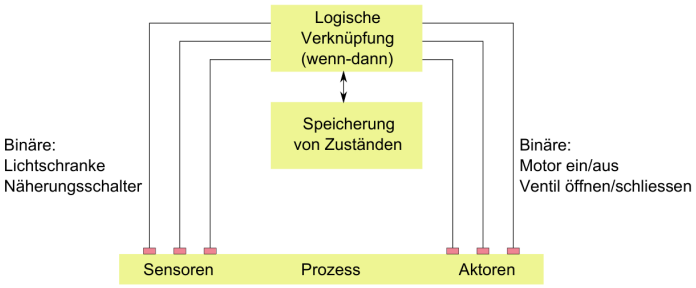
1.3 Allgemeine Anforderungen

Echtzeitfähigkeit	Betriebsysteme Speicherarchitektur
Ein- Ausgabe von Prozesssignalen	Aktoren Sensoren Kommunikationssysteme
Sicherheit Zuverlässigkeit	Hochwertige Komponente Redundanz
Resistenz gegen Umwelteinflüsse	Spezielle Gehäuse Schaltschrankmontage

2 SPS

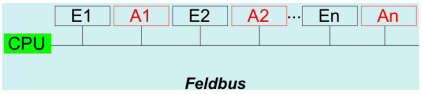


Eine SPS ist eine 'Speicherprogrammierbare Steuerung'. Auf Englisch wird sie PLC (Programmable Logic Controller) genannt. Sehr wichtig: die SPS arbeitet konventionell möglichst auf if/else Statements und verarbeitet sämtlicher Code in jedem Zyklus einmal. Das Arbeitsprinzip ist:



2.1 Feldbus

Der Feldbus ist die Kommunikationsschnittstelle der SPS zu den Klemmen und entsprechend Ein- und Ausgängen. Es hängen alle E/A seriell an einem Bus. Wo der Feldbus aktiv ist, wird Feldebene genannt. Der Aufbau des Feldbusses kann wie folgt vorgestellt werden:



Definition Feldbus

Ein Feldbus verbindet in einer Anlage Feldgeräte wie Sensoren und Aktoren zwecks Kommunikation mit einem Steuerungsgerät (SPS).

Durch den Feldbus entstehen einige wesentliche Vor- und Nachteile:

+ geringer Verkabelungsaufwand
+ Erweiterungen oder Änderungen sind einfach

– Komplexität
– Preis
– Aufwendige Messgeräte (Analyzer)
– Längere Reaktionszeit

2.1.1 Spezifische Feldbusse

Im Gebäude sind die gängigen Feldbusse:

- LON
- EIB
- KNX

Innerhalb der Maschine werden andere Feldbusse verwendet:

- CAN/CANOpen
- Profibus, Profinet
- EtherCAT
- Ethernet/IP

3 Programmierung

Definition SPS-Programmierung

Die EN 61131-3 ist die einzige weltweit gültige Norm für Programmiersprachen von SPS-Steuerungen. Sie definiert die folgenden fünf Sprachen:

- AWL - Anweisungsliste (Assemler)
- KOP - Kontaktplan
- FBS - Funktionsbausteinsprache
- ST - Struktureierter Text
- AS - Ablaufsprache

Die einzige (relevante) Programmiersprache hinsichtlich des IAUT Unterrichts ist ST. Die anderen werden für einfache logische Verknüpfungen gebraucht, oder für z.B. I/O-Zuweisungen. ST wird in Europa oft für die Programmierung von SPS gewählt.

3.1 EN 61499

Diese Norm stellt eine objektorientierte Erweiterung der EN61131 dar.

4 Digitale Regelung

5 FSM - Finite State Machine

6 ST-Language

Dieses Kapitel fasst den Aufbau und Syntax der SPS-Programmiersprache ST zusammen.

6.1 Grundaufbau

Der Aufbau eines Programmes/Funktionsblockes oder Funktion ist immer in zwei Bereiche aufgeteilt: Deklaration und Implementation. Im Deklarationsteil werden alle Inputs, Outputs deklariert und ggf. initialisiert. In der Implementation wird der eigentliche Code geschrieben. Wichtig: die Implementation wird in jedem Clock Zyklus wiederholt.

```
// Deklarationsteil
VAR_INPUT
    // input-variablen
    iVarIn AT %I* : INT;
END_VAR
VAR_OUTPUT
    // output-variablen
    iVarOut AT %Q* : INT;
END_VAR
VAR_IN_OUT
    // in/out-variablen
END_VAR
VAR
    // lokale variablen
    tmp : INT;
END_VAR

// Implementationsteil
IF iVarIn > 10 THEN
    tmp := 0;
ELSE
    tmp := iVarIn;
END_IF
```

6.2 Variablentypen

6.2.1 DUT - Data Unit Type

Über DUT können benutzerdefinierte Datentypen angelegt werden. Die relevanten sind `STRUCT` und `ENUM`. Durch diese Typen kann ein Programm bedeutend lesbarer gestaltet werden.

6.2.2 Programm

Hallo das ist das erste Mal wo ich von Linux aus das hier mache. So far so good, es geht wirklich gut.

6.2.3 Funktionsblock

6.2.4 Funktion

6.3 Variablen-Scope

6.3.1 Globale Variablen

In einer globalen Variablenliste `GVL` werden global verwendete Variablen deklariert. Sinnvoll ist dies zum Beispiel für: SPS Zykluszeit, Ein-/Ausgänge oder sonstige Konstanten, welche überall im Code wieder verwendet und gegebenenfalls ändern.

6.4 Conditional Statements

6.4.1 If/Else

```
IF <cond1> THEN
    ...
ELSIF <cond2> THEN
    ...
ELSE
    ...
END_IF
```

6.4.2 Switch-Case

```
CASE iVar OF
    1:
        // do something for 1
    2..5:
        // do something for range 2 - 5
ELSE
    // do this as default-assignment
END_CASE;
```

Die Switch-Bedingung kann auch für Enum-Typen gebraucht werden:

```
CASE eState OF
    Init:
        // handle init state
    Running:
        // handle running state
    Error:
        // handle error state
ELSE
    // it is beneficial to always have a default
    statement
END_CASE;
```

6.5 Schleifen

Wie in C gibt es in TwinCat Schleifen. Dabei ist es von zentraler Bedeutung, dass diese Schleife nicht Endlos ist und eine Abbruchbedingung enthält. Ansonsten kann es vorkommen, dass auf die SPS nicht mehr zugegriffen werden kann. Die verschiedenen Schleifen werden hier aufgeführt.

6.5.1 WHILE

WHILE-Schleife

Sofern die Bedingung beim ersten Erreichen nicht zutrifft, wird der Inhalt der Schleife nicht ausgeführt.

```
WHILE iCounter <> 0 DO
    iVar1 := iVar2;
    iCounter := iCounter - 1;
END_WHILE;
```

6.5.2 REPEAT

REPEAT-Schleife

Diese Schleife wird mindestens einmal durchgeführt.

```
REPEAT
    iVar1 := iVar2;
    iCounter := iCounter - 1;
UNTIL
    iCounter = 0;
END_REPEAT;
```

6.5.3 FOR

```
FOR iCounter = 1 TO 5 BY 1 DO
    iVar1 := iVar2;
END_FOR;
```

6.5.4 Schleifen Modifikatoren

Mit diesen Modifikatoren kann wie in C beeinflusst werden, wie das Verhalten in der Schleife ist.

```
EXIT;       // Ausbruch aus aktuellem Loop
CONTINUE;   // Direkter Sprung zu Loop Anfang
```

7 TwinCat

TwinCat ist die Entwicklungsumgebung von Beckoff für deren SPS Systeme. TwinCat liefert diverse Funktionen vorgefertigt für den Benutzer. In diesem Kapitel werden die relevantesten aufegelistet.

8 Beispiel Code