Programmable Logic Controller (PLC)

Seminar: Distributed Real-time Systems
Outline

- History and basic idea
- General structure of a PLC based system
- Programming a PLC - Basic structure
- Programming languages
- Real-life examples
History and basic idea
History and basic idea

Closed-Loop Control

- Control
  - Input-Signal
  - Control-Signal
- Actor
  - Control-Signal
- Process
- Sensor

Programmable Logic Controller | Stefan Reichel | 23.06.2010
History and basic idea

- Situation:
  - Hundred or thousands of relays, closed-loop controllers in cars
  - Complete rewiring for new model creation needed
History and basic idea

- General Motors Hydramatic requests proposal for an alternative
  - Winner “Bedford Associates” with 084
  - Foundation of Modicon, MOdular DIgital CONtroller
  - Richard (Dick) Morley inventor of PLC
  - 1969 invention of solid-state sequential logic solver
A programmable logic controller (PLC) or programmable controller is a digital computer used for automation of electromechanical processes.
Structure of PLC based systems
Structure of PLC based systems

BUS e.g. Ethercat, CAN
Structure of PLC based systems

- Several SPS types available
  - Hardware SPS
  - Soft SPS
  - Slot SPS
Structure of PLC based systems

Programmable Logic Controller | Stefan Reichel | 23.06.2010
Structure of PLC based systems

- Two ways of working
  - Cyclic
  - Event based
Programming a PLC system
- PLC Programming process:

  - Creation of new project
  - Definition of labels
  - Creation of ST program
  - Write to PLC CPU
  - Online debugging

Creating a new ST project.

Defining the labels to be used in an ST program.

Creating an ST program.

Converting (compiling) the created ST program into an executable sequence program.

Correcting the program if a convert (compile) error occurs.
Programming a PLC

- IEC 61131-3 international standard defines PLC programming languages and concepts

IEC 61131-3 Programming languages

- Text-based
  - Instruction-List
  - Structured-Text

- Graphical
  - Ladder-Diagram
  - Sequential Function Chart
    - Function Block Diagram
Programming a PLC

- Functions are reusable
- Various predefined functions:

<table>
<thead>
<tr>
<th>Class</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bool</td>
<td>AND, NOT, OR, XOR</td>
</tr>
<tr>
<td>Mathematic</td>
<td>ADD, SUB, NEG, DIV, MUL, MOD</td>
</tr>
<tr>
<td>Casting</td>
<td>BYTE_TO_WORD, INT_TO_REAL</td>
</tr>
<tr>
<td>Numeric</td>
<td>SIN, TAN, COS, LN, LOG</td>
</tr>
<tr>
<td>Comparator</td>
<td>EQ, LT, GT</td>
</tr>
<tr>
<td>Bit-Operations</td>
<td>SHL, SHR, ROL, ROR</td>
</tr>
<tr>
<td>Selection</td>
<td>LIMIT, MIN, MAX</td>
</tr>
</tbody>
</table>
Function blocks are reusable
- Accessable as instances with own state → object orientation
- Various predefined functions blocks available

<table>
<thead>
<tr>
<th>Class</th>
<th>Function blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer</td>
<td>TP, TON, TOF</td>
</tr>
<tr>
<td>Trigger</td>
<td>F_TRIG, R_TRIG</td>
</tr>
<tr>
<td>Flip-Flops</td>
<td>SR, RS</td>
</tr>
<tr>
<td>Counter</td>
<td>CTU, CTD, CTUD</td>
</tr>
</tbody>
</table>
Programming a PLC

- PLC resource addressing

<table>
<thead>
<tr>
<th>I/O Identification</th>
<th>Source</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT %</td>
<td>I for Input</td>
<td>X Bit</td>
</tr>
<tr>
<td></td>
<td>Q for Output</td>
<td>B Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W Word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D DoubleWord</td>
</tr>
</tbody>
</table>

- Addressing example:
  - AT %IX1.2
  - AT %IW6
Variable declaration:

- **NAME**  ADDRESS:  **DATATYPE**  :=**INIT**;  (***)

```plaintext
VAR
  Sensor AT%IX0.1 :Bool :=false; (*Beispiel*)
END_VAR;

VAR INPUT
  X: REAL;
  Y: REAL;
END_VAR;

VAR_OUTPUT
  ERGEBNIS: INT;
END_VAR;
```
Several variable types:

- **VAR_INPUT** *Input variables*
- **VAR_OUTPUT** *Output variables*
- **VAR** *Local variables*
- **VAR_GLOBAL** *Global variables*
- **VAR_IN_OUT** *Variable can be changed and returned*
- **VAR_RETAIN** *Variable keeps value after power off*
- **VAR_PERSISTANT** *Variable keeps value after software redeployment*
# Programming a PLC - Datatypes

<table>
<thead>
<tr>
<th>Datentyp</th>
<th>Größe</th>
<th>Wertebereich</th>
<th>Initialwert/Beispiel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitfolgen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOOL</td>
<td>1 Bit</td>
<td>FALSE/TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>BYTE</td>
<td>8 Bits</td>
<td>16#00...16#FF (Hex-Darstellung)</td>
<td>16#00</td>
</tr>
<tr>
<td>WORD</td>
<td>16 Bits</td>
<td>16#0000...16#FFFF</td>
<td>16#0000</td>
</tr>
<tr>
<td>DWORD</td>
<td>32 Bits</td>
<td>16#00000000...16#FFFFFFFF</td>
<td>16#00000000</td>
</tr>
<tr>
<td>Ganze Zahlen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SINT</td>
<td>8 Bits</td>
<td>-128...127</td>
<td>0</td>
</tr>
<tr>
<td>INT</td>
<td>16 Bits</td>
<td>-32768...32767</td>
<td>0</td>
</tr>
<tr>
<td>DINT</td>
<td>32 Bits</td>
<td>-2147483648...2147483647</td>
<td>0</td>
</tr>
<tr>
<td>Ganze Zahlen ohne Vorzeichen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USINT</td>
<td>8 Bits</td>
<td>0...255</td>
<td>0</td>
</tr>
<tr>
<td>UINT</td>
<td>16 Bits</td>
<td>0...65535</td>
<td>0</td>
</tr>
<tr>
<td>UDINT</td>
<td>32 Bits</td>
<td>0...4294967295</td>
<td>0</td>
</tr>
<tr>
<td>Fließkommazahlen</td>
<td></td>
<td>1,18\times10^{-38}...3,4\times10^{38}</td>
<td>0.0</td>
</tr>
<tr>
<td>Zeit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>32 Bits</td>
<td>t#0s:0ms</td>
<td></td>
</tr>
<tr>
<td>Datum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>32 Bits</td>
<td>d#2003-18-01</td>
<td></td>
</tr>
<tr>
<td>Zeichenfolge</td>
<td></td>
<td></td>
<td>STRING(4)='Text'</td>
</tr>
</tbody>
</table>
Programming Languages
- **Instruction list**, assembler like programming language
- Very lightweight language
- Every line consists of command and operand
- Several commands defined, extract:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>Load variable</td>
</tr>
<tr>
<td>ST</td>
<td>Store variable</td>
</tr>
<tr>
<td>JMP</td>
<td>Jump</td>
</tr>
<tr>
<td>JMPC</td>
<td>Conditional jump</td>
</tr>
<tr>
<td>CAL</td>
<td>Call of a function/program/block</td>
</tr>
</tbody>
</table>
- **Instruction list**, example:

\[ a \land (b \lor (c \land \neg d)) = e \]

```
LD A
AND (B
OR (C
ANDN (D
)
)
)
ST E
```
Structured text, close to high languages like C, pascal

Every command ends with a semicolon

Allows conditions like if/case and loops

Assignments with :=

**Example**

```plaintext
IF (TEMP > 20) THEN
    HEATER := OFF;
    COOLER := ON;
ELSIF (TEMP < 19)
    HEATER := ON;
    COOLER := OFF;
END_IF;

CASE f OF
    1:  a:=3;
    2:  a:=5;
    3:  a:=2;
ELSE a:=0;
END CASE;
```
Structured text support several loop types

FOR a:=0 TO 10 BY 1 DO
  c:=a + 4;
END_FOR;

WHILE b > 1 DO
  b:= b/2;
END_WHILE;

REPEAT a:= b * c;
  UNTIL a > 1000;
END_REPEAT;

Function and function block calls

MYFB(IN:=a, IN:=b);
c:= MYFB.Q;

c:= MYFC(a,b);
Our example:

\[ a \land (b \lor (c \land \neg d)) = e \]

\[ e := a \text{ AND } (b \text{ OR } (c \text{ ANDN } d)) \]
Programming a PLC – Ladder diagram

- **Ladder diagram**, graphical programming language
- Close to circuit diagram
- Power flow from left to right

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>--()</td>
<td>Output relais</td>
</tr>
<tr>
<td>--P--</td>
<td>Detection of positive change 0→1</td>
</tr>
<tr>
<td>--N--</td>
<td>Detection of negative change 1→0</td>
</tr>
</tbody>
</table>
Our example:

\[ a \land (b \lor (c \land \neg d)) = e \]
Programming a PLC – Function block

- **Function block diagram**, graphical programming language
- Based on function and function block composition
Our example:

\[ a \land (b \lor (c \land -d)) = e \]
- **Sequencial Function Chart**, graphical programming language
- Only used for sequencial data flows
- Consists of actions and transitions
Real-Life examples
Real-Life examples
Real-Life examples
Real-Life examples
Real-Life examples
References

- Speicherprogrammierbare Steuerungen, Matthias Seitz,
  - ISBN: 978-3-446-41431-0
  - http://www.sps-lehrgang.de/kontaktplan-kop/
  - http://www.plcmanual.com/plc-programming
  - http://www.plcsimulator.net/plc.php
Thank you for your patience