School of Electrical Engineering

Department of Electrical Engineering and Automation

ELEC 8201 Control and Automation

Exercise session 9
State-Based Design
Implementation issues

Valeriy Vyatkin
Pranay Jhunjhunwala
Udayanto Dwi Atmojo

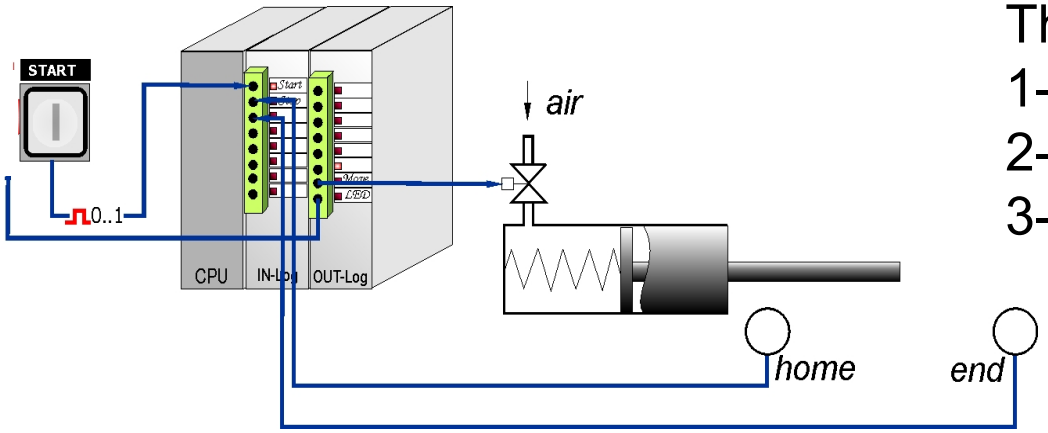
Aalto University

Plan

1. State machine design for single acting cylinder
2. ST language

Example: Pneumatic Cylinder with Return Spring

Single acting cylinder has only one control signal **FWD** (move forward)



This system has 3 input signals
1- HOME
2- START
3- END

Actions and Signals:

- **HOME and END** signals are generated by 2 proximity sensors that are located in 2 ending positions
- **START** signal is generated by Start button.

Required behaviour of the system

Short requirement: On press of the START button, make a single trip of the cylinder to the end position and return it back afterwards.

Scenario:

In the initial state, the HOME proximity sensor is ON.

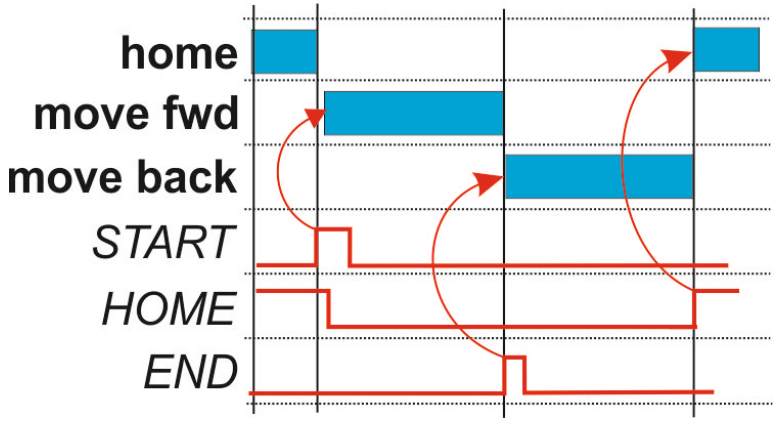
When user pushes the START button, it causes the FWD signal to go ON, which causes the cylinder to move forward.

When the cylinder starts moving, the HOME proximity sensor goes OFF.

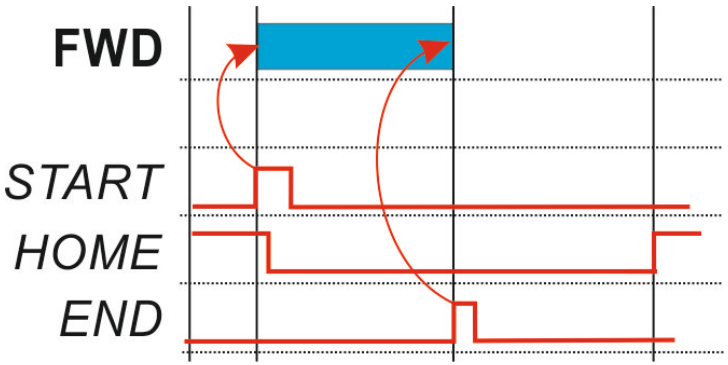
Cylinder moves forward until it reaches the rightmost position, where the END proximity sensor will be triggered, changing its value from 0 to 1.

The FWD signal should be set to 0 and therefore there will be no power on the cylinder's actuator. The spring then will pull the cylinder back to the leftmost position and the first proximity sensor sets the HOME variable to 1.

Sensors and activities:



Sensors and control signals:



State-based Logic Design

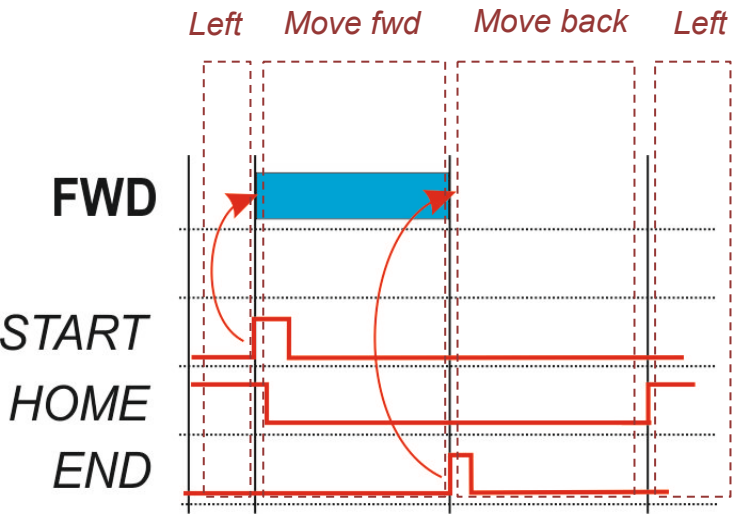
Step1: Identify stable states

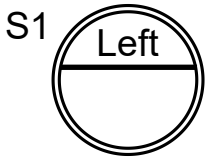
S1: Left (cylinder is in the leftmost, Home position)

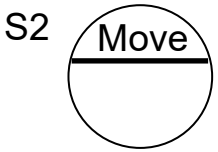
S2: Move (cylinder is moving toward the end position)

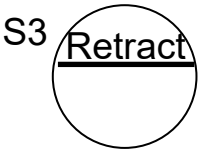
S3: Retract (spring is moving the cylinder back to the home position)

Scenario









State-based Logic Design

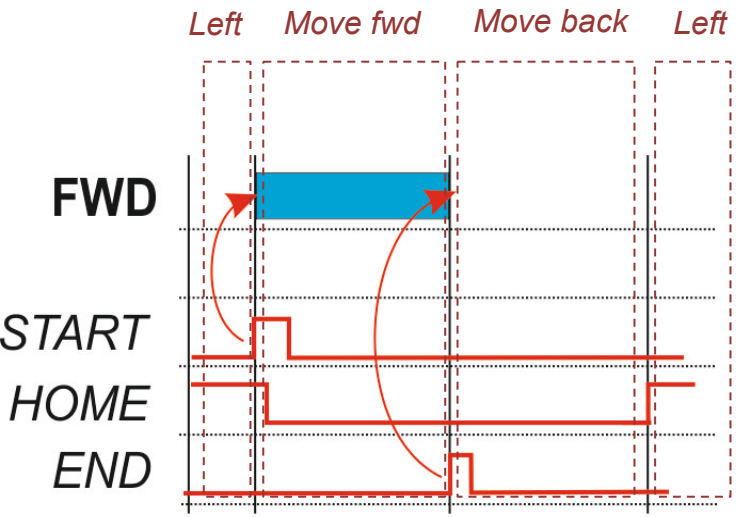
Step2: What control signals are to be set ON in the states? Add actions in the states

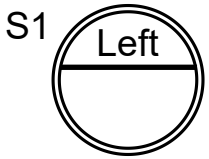
Actions in S1:

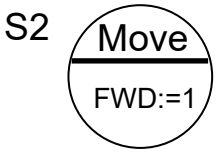
Actions in S2: FWD := 1

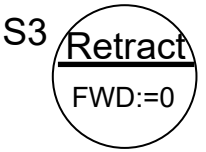
Actions in S3: FWD := 0

Scenario









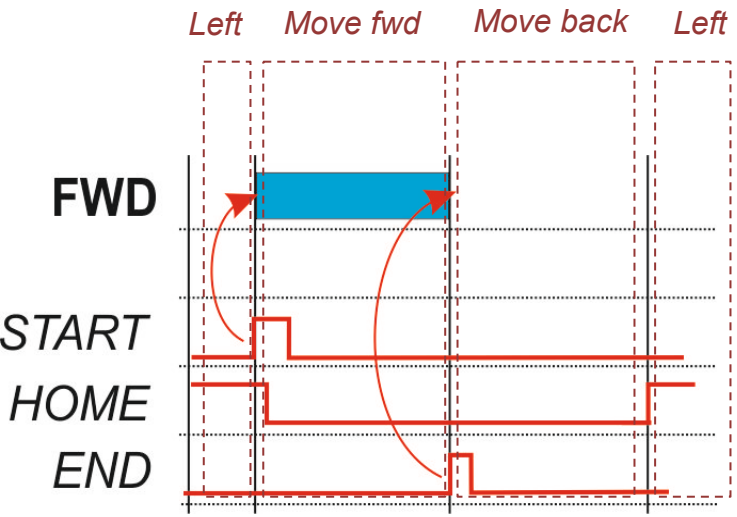
State-based Logic Design

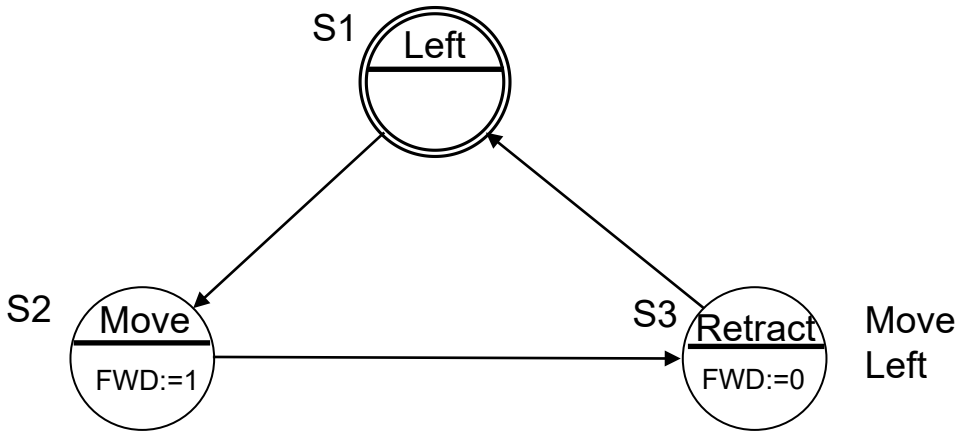
Step3: Add transitions between the states

Transition is an event of system changing its current state.

Transitions have a Boolean condition.

Scenario



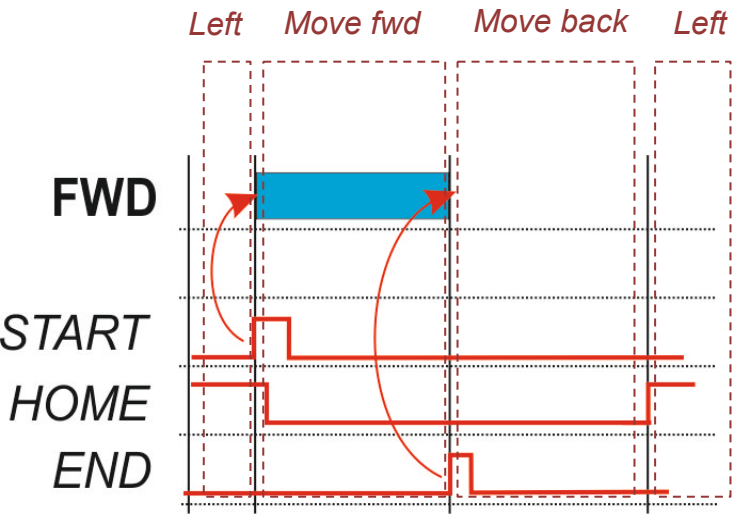


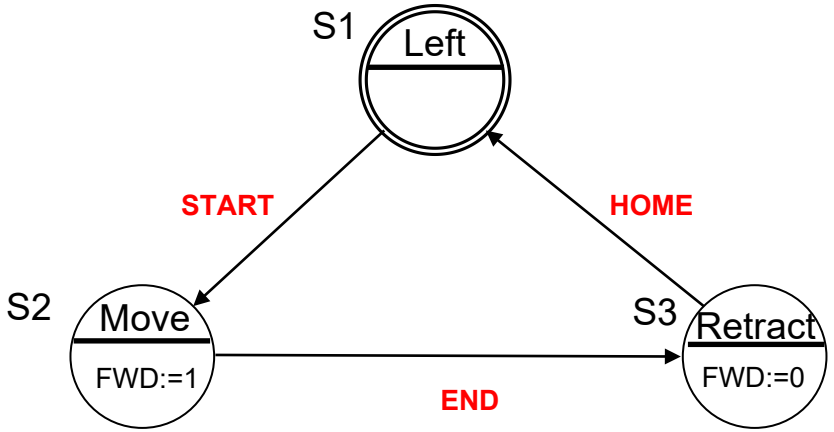
State-based Logic Design

Example: Double acting Pneumatic Cylinder

Step 4: Add Boolean conditions to the transitions.

Scenario





Aalto University

Exercises

- Consider the Cylinder system with changed requirements:
  - What if the motion to be started on release of the START button?
  - What if the second click on the START button during the motion shall abort the process?
  - What if the cylinder should only return to the HOME position, after the START button is again pressed ?

Structured Text - ST

- Structured Text (ST) is a textual high-level programming language (like e.g. C)
- ST language can be used in:
  - ST programs
  - Function and Function block logic coding
  - Action logic coding in SFC programs

- The program code is composed of statements separated by semicolons; and containing expressions
- Numerous constructs can be used for programming e.g. different loops, thus allowing the development of complex algorithms

```
PROGRAM PLC_PRG
VAR
	VarA: BOOL;
	VarB: DWORD;
	VarC: REAL;
	Alpha: REAL;
	DatumTrend: DATE;
	tTimer: TIME;
	szText : STRING;
END_VAR
```

```
IF TIME()-tTimer > t#1s THEN
	tTimer := TIME();
	VarA := NOT VarA;
	VarB := VarB + 1;
	Alpha :=Alpha + 1.0;
	IF Alpha >= 90 THEN
		Alpha := 0;
	END_IF
	VarC := 50.0 + 50.0 * SIN(0.0698131700777 * Alpha);
	DatumTrend := d#2012-06-02;
END_IF
```

ST Basics: Expressions

- An *expression* is a construct which, when evaluated, yields a value corresponding to one of the data types.
  - Expressions are composed of operators and operands.
  - An operand shall be a variable, a function invocation, a constant or another expression.

- For example,
  - Variables A, B, C, and D are of type INT with values 1, 2, 3, and 4

    *A + B – C * ABS(D)*
  - evaluates to -9

ST Basics

Each statement must end with a semi-colon (";")

Basic statements:

- Assignment - :=
  - *Q:=IN;*
  - *Q:=sin(angle);*
  - *Q := (IN1 + (IN2 / IN 3)) * IN4;*

- Conditional statement
  - *IF / THEN / ELSE (simple binary switch)*
  - *CASE (enumerated switch)*

Conditional Statements

IF / THEN / ELSE / ELSIF

```
IF Switch1 = True THEN
        Switch1 := False;
        ELSE
        Switch1 := True;
END_IF;
```

```
IF Switch1 = True THEN
    Switch1 := False;
    ELSIF PumpOn1 = True THEN
        PumpOn1 := False;
END_IF;
```

Conditional Statements

CASE statement Example

```
CONST
    plus:= 1;
    minus:= 2;
    times:= 3;
END_CONST
CASE operator OF
    plus: a := b + c;
    minus: a := b - c;
    times: a := a * b;
END_CASE;
```

ST Basics

Loops:

- While
- Repeat
- For

WHILE Statement

The WHILE statement contains an *expression* whose truth value determines whether or not the statement that follows DO is to be carried out again.

```
//Searching a value from a data array:
i := StartValue;
WHILE Data[i] <> x DO
    i := i + 1;
END_WHILE;
Index := i;
```

REPEAT Statement

The statement is carried out repeatedly until the expression takes on the value TRUE. The statement is carried out at least once

```
j := -4;
REPEAT
    j := j + 2;
UNTIL j > 60 END_REPEAT;
REPEAT
    a := in1 + in2;
    b := 2 * in1;
    c := in1 * in2;
UNTIL EndCondition END_REPEAT;
```

FOR Statement

The FOR statement carries out a DO loop in which new values are assigned to a variable (the controlled variable).

```
//Searching the maximum value in a data array:
Maximum := 0;
FOR lw := 2 TO 63 DO
    IF Data[lw] > Maximum THEN;
        Maximum := Data[lw];
        MaxIdx := lw;
    END_IF;
END_FOR;
```