

## Market-Driven Systems

Marknadsstyrda System

FRTN20

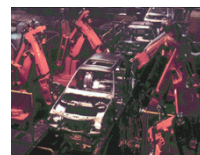
### Lecture 2: Discrete Production Processes

1

## Discrete Production Processes

General Characteristics of discrete production processes:

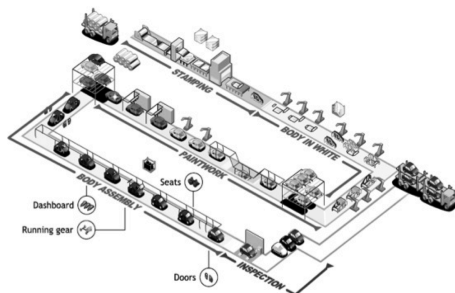
- Discontinuous production of product, i.e. discrete output.
- Discontinuous flow of material (often pieces and parts).
- Assembly-oriented production.
- Staged production through work cells
- Well defined production runs.
- The product is most often "visible".
- The equipment operates in on-off manner.



2

## Discrete Production Processes

A discrete production process is the assembly of piece parts into products. The product is a discrete entity.



3

## Discrete Event Systems

Definition:

A *Discrete Event System (DES)* is a discrete-state, event-driven system, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events over time.

Sometimes the name *Discrete Event Dynamic System (DEDS)* is used to emphasize the dynamic nature of DES.

4

## Discrete Event Systems

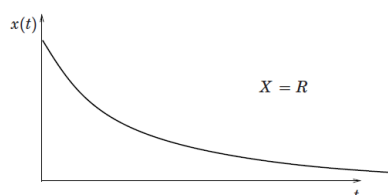
1. The state space is a discrete set
2. The state transition mechanism is event-driven
3. The events can be synchronized by a clock, but they do not have to be (i.e., the system can be asynchronous)

5

## Continuous-Time Systems

State trajectory is the solution of a differential equation

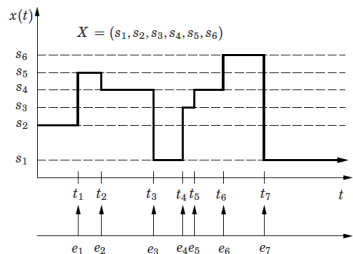
$$\dot{x}(t) = f(x(t), u(t), t)$$



6

Discrete-Event Systems

State trajectory (sample path) is piecewise constant function that jumps from one value to another when an event occurs.



7

How do we control a machine?

Automation of the discrete operations (on-off) is largely a matter of a series of carefully timed on-off steps. The equipment performing the operations operates in an on-off manner.

- Discrete signals
- Control parameters: true or false
  - Actuators: on or off

- Interlocks ("föreglingar")
- Output = function(input)
  - Boolean algebra

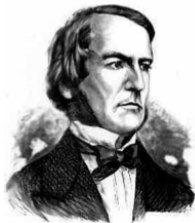
8

George Boole (1815-1864)

Boole approached logic in a new way reducing it to a simple algebra, incorporating logic into mathematics.

He also worked on differential equations, the calculus of finite differences and general methods in probability.

*An investigation into the Laws of Thought, on Which are founded the Mathematical Theories of Logic and Probabilities (1854)*



9

Logic: Operations and symbols

Three types of operations			
AND	$a \cdot b$	$a$ and $b$	$a \wedge b$
OR	$a + b$	$a$ or $b$	$a \vee b$
NOT	$\bar{a}$ (or $a'$ )	not $a$	$\neg a$

a)  $a \cdot b$

b)  $a + b$

c)  $\bar{a}$

d)  $a \cdot b$

e)  $\bar{a}$

a)  $a + b$

b)  $a \cdot b$

c)  $\bar{a}$

d)  $a \cdot b$

e)  $\bar{a}$

10

Logic: Rules

Boolean Algebra:

- Example:  $1+a=1$  and  $0+a=a$
- Example:  $a+a=1$  and  $a \cdot a=a$
- Example:  $a+a=a$  and  $a \cdot a=a$

Logical Rules

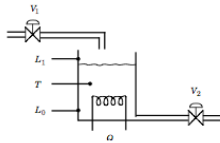
Commutative	$a \cdot b = b \cdot a$
Associative	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
Distributive	$a \cdot (b + c) = a \cdot b + a \cdot c$
De Morgan	$a + b = \overline{\overline{a} \cdot \overline{b}}, \quad a \cdot b = \overline{\overline{a} + \overline{b}}$

11

Logics: Example

Discrete logics can also be used for other types of applications, e.g., alarms.

**Alarm for a batchreactor:**  
Give an alarm if the temperature in the tank is too high, T, and the cooling is closed, not-Q, or if the temperature is high and the inlet valve is open, V1.



Truth table:

T	Q	V1	y = alarm
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	1
0	1	1	0
1	1	1	1

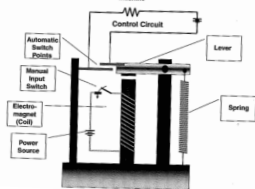
Logic:

$y = TQV + TQ\bar{V}$   
 $+ TQV @ y = TQ(V + \bar{V})$   
 $+ TQV @ y = TQ$   
 $+ TQV y = T(Q + QV)$

12

### Electro-Mechanical Relays

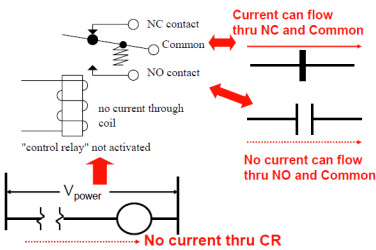
The basic device invented for control of discrete production processes is the automatic switch and interconnected sequences of automatic switches.



Today, the automatic switches are replaced by computer programs. This technological innovation took place in the 1960s, since then discrete systems have become more automated.

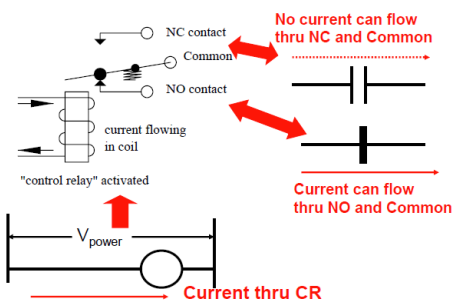
13

### Control Relay – Not Activated



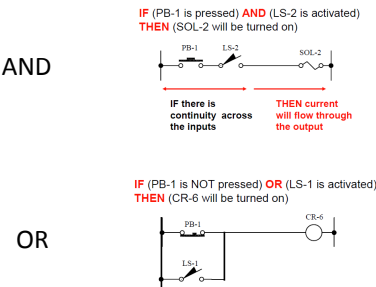
14

### Control Relay - Activated



15

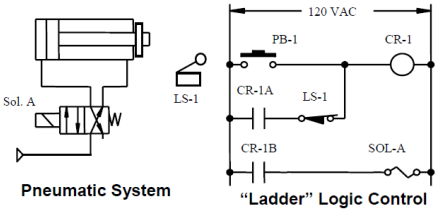
### Logic Control



16

### Single Cylinder Stroke #1

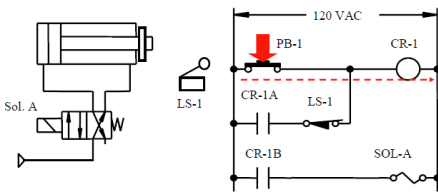
- Pressing the pushbutton PB-1 will cause the cylinder to extend and retract one time



17

### Single Cylinder Stroke #2

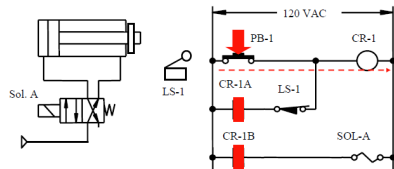
- Pressing the momentary contact pushbutton PB-1 energizes the control relay CR-1



18

### Single Cylinder Stroke #3

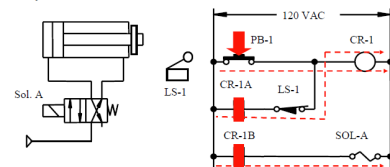
- After control relay CR-1 energizes, normally open contacts CR-1A and CR-1B activate



19

## Single Cylinder Stroke #4

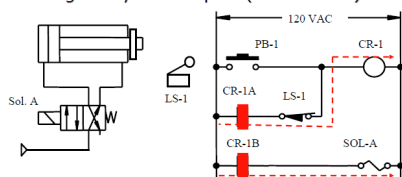
- ▶ Control relay CR-1 is now energized by a 2nd path, solenoid SOL-A also activates



20

## Single Cylinder Stroke #5

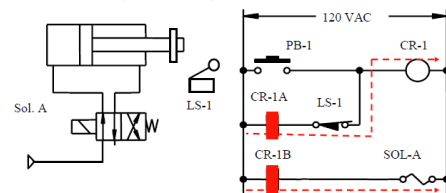
- PB-1 is released, but control relay CR-1 is still energized by the 2nd path ("hold" circuit)



21

## Single Cylinder Stroke #6

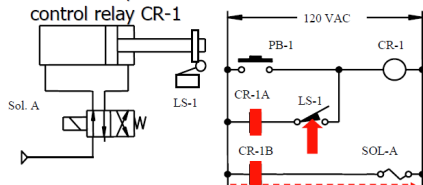
- Solenoid A shifts the valve spool to the right, and the cylinder begins to extend



22

## Single Cylinder Stroke #7

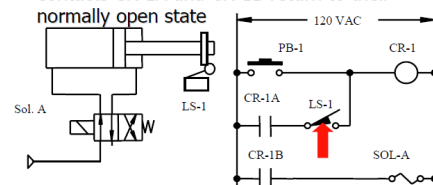
- Cylinder activates the normally closed limit switch LS-1, which "kills" the hold circuit for control relay CR-1



23

### Single Cylinder Stroke #8

- With control relay CR-1 de-activated, the contacts CR-1A and CR-1B return to their normally open state

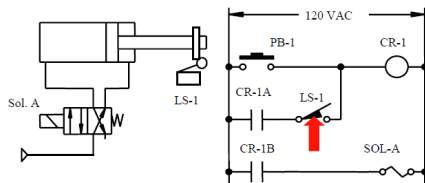


Solenoid SOL-A is no longer energized

24

## Single Cylinder Stroke #9

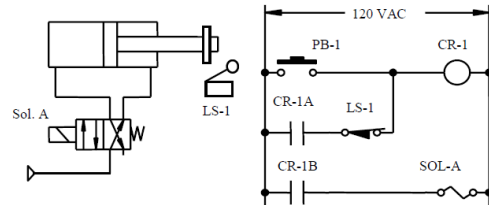
- ▶ CR-1B is now open, SOL-A is de-activated, spring returns valve to default state



25

## Single Cylinder Stroke #10

- Cylinder begins to retract, and “rolls off” of LS-1, which returns to its N.C. state



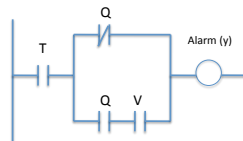
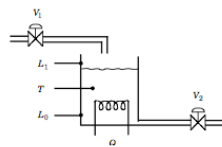
26

## Batch Reactor Example revisited

## Using ladder diagrams for simple interlocks

**Alarm for a batchreactor:**

Give an alarm if the temperature in the tank is too high,  $T$ , and the cooling is closed, not- $Q$ , or if the temperature is too high,  $T$ , and the inlet valve is open,



**Logic:**

$$\begin{aligned} \blacksquare y &= TQV + TQV \\ + TQV @ y &= TQ(V + V) \\ + TQV @ \blacksquare y &= TQ \\ + TQVy &= T(Q + QV) \end{aligned}$$

27

Logics:

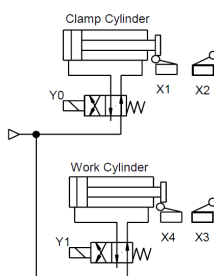
### Example (discrete production process)

A common manufacturing process:

- receive an object
- position the object
- clamp the object
- “do something” to (work on) the object
- release the object

28

### Example: Clamp/Work Layout



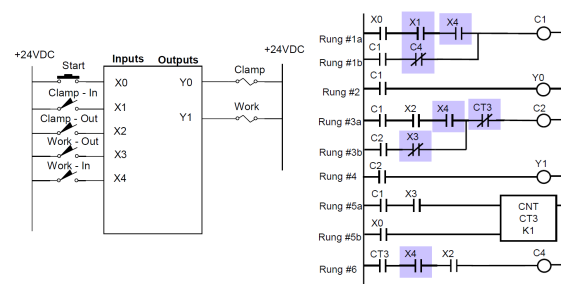
Process:

1. Press the "START" button
2. Extend the clamp cylinder
3. When the clamp cylinder is fully engaged, start extending the work cylinder
4. When the work has finished, retract the work cylinder
5. When the work cylinder is fully retracted, retract the clamp cylinder

NOTE: Example from University of Alabama ([www.me.ua.edu](http://www.me.ua.edu))

20

### Example: Clamp/Work Layout



### Relay Ladder Logic Notation with counter

Each relay represents a state variable

30

## Logic

- Larger in volume than continuous control
- Very little theoretical support
  - verification, synthesis
  - formal methods beginning to emerge
  - still not widespread in industry

31

## Relay/Ladder Logic

- Very user unfriendly way of programming logic
- Still very common in discrete production processes
  - E.g. Tetra Pak

32

## How do we control a plant?

Automation of the discrete operations (on-off) is largely a matter of a series of carefully timed on-off steps.

### Discrete signals

- Control parameters: true or false
- Actuators: on or off

### Interlocks ("föreglingar")

- Output = function(input)
- Boolean algebra

### Sequence nets:

- Dynamic systems
  - Output =  $f(\text{input}, \text{state})$
  - Next\_state =  $g(\text{input}, \text{state})$
- Finite State machines, Petri Nets, Grafcet/SFC, Grafchart

33

## Finite State Machines

Formal properties -> Analysis possible in certain cases.

Using finite state machines is usually a good way to structure code.

Using state machines is usually a good way to visualize a behaviour.

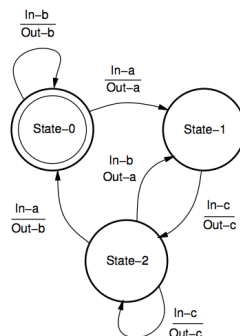
Two basic types of finite state machines

- Mealy machines
- Moore machines

34

## Mealy Machine

Output events (actions) associated with input events.

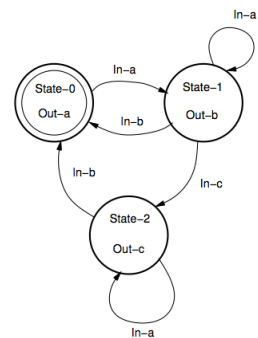


35

## Moore Machine

State transitions in response to input events

Output events (actions) associated with states

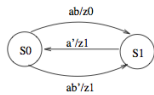


36

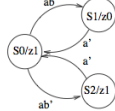
## Mealy and Moore Machines

A Mealy-machine (left) and the corresponding Moore-machine (right) are shown. The two state machines have two inputs,  $a$  and  $b$ , and two outputs,  $z0$  and  $z1$ .

Mealy:



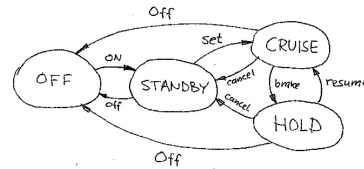
Moore:



37

## Finite State machine Cruise Control

Cruise controls are common in vehicle applications.



38

## State Machines

Ordinary state machines lack structure.

Extensions needed to make them practically useful

- hierarchy
- concurrency
- history (memory)

39

## Carl Adam Petri (1926-2010)

Carl Adam Petri (born July 12, 1926) is a German mathematician and computer scientist.

Petri Nets were invented in August 1939 by Carl Adam Petri - at the age of 13. He documented the Petri net in 1962 as part of his dissertation. It significantly helped define the modern studies of complex systems and workflow management.

*Kommunikation mit Automaten, Carl Adam Petri (1962)*



40

## Petri Nets

A mathematical and graphical modeling method. Describe systems that are:

- Concurrent  
(several computations are executing simultaneously)
- asynchronous or synchronous  
(not coordinated by a clock vs coordinated by a clock)
- Distributed  
(several computations that run autonomously but exchange information to reach a common goal)
- nondeterministic or deterministic  
(the output is not vs is completely given by the input)

41

## Petri Nets

Can be used at all stages of system development:

- modeling
- analysis
- simulation/visualization  
(“playing the token game”)
- synthesis
- implementation (Grafcet)

42

## Application Areas

- flexible manufacturing systems
- logical controller design
- communication protocols
- distributed systems
- multiprocessor memory systems
- dataflow computing systems
- fault tolerant systems
- ...

43

## Introduction

A Petri net is a directed bipartite graph consisting of places P and transitions T.

Places are represented by circles.

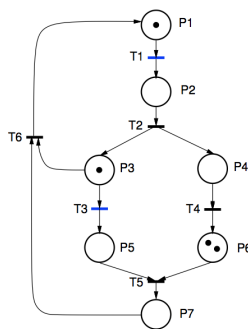
Transitions are represented by bars (or rectangles).

Places and transitions are connected by arcs.

In a marked Petri net each place contains a cardinal (zero or positive integer) number of tokens or marks.

44

## Example



45

## Firing Rules

1. A transition  $t$  is enabled if each input place contains at least one token.
2. An enabled transition may or may not fire.
3. Firing an enabled transition  $t$  means removing one token from each input place of  $t$  and adding one token to each output place of  $t$ .

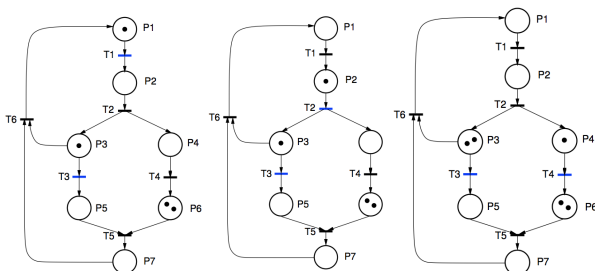
The firing of a transition has zero duration.

The firing of a sink transition (only input places) only consumes tokens.

The firing of a source transition (only output places) only produces tokens.

46

## Example



47

## Petri Net variants

Generalized Petri Nets:

Weights associated to the arcs

Times Petri Nets

Times associated with transitions or places

High-Level Petri Nets:

Tokens are structured data types (objects)

Continuous & Hybrid Petri Nets:

The markings are real numbers instead of integers. Mixed continuous/discrete systems

48



## Analysis

Properties:

- Live: No transitions can become unfireable.
- Deadlock-free: Transitions can always be fired
- Bounded: Finite number of tokens ...

49

## Analysis

Analysis methods:

- Reachability methods – exhaustive enumeration of all possible markings
- Linear algebra methods – describe the dynamic behaviour as matrix equations
- Reduction methods – transformation rules that reduce the net to a simpler net while preserving the properties of interest

50

## Grafcet

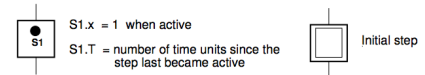
- Extended state machine formalism for implementation of sequence control
- Industrial name: Sequential Function Charts (SFC)
- Defined in France in 1977 as a formal specification and realization method for logical controllers
- Part of IEC 61131-3 (industry standard for PLC controllers)

51

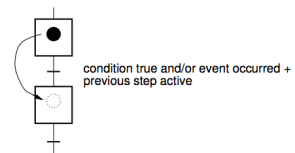
## Basic Elements

Steps:

- active or inactive



Transitions ("övergång"):

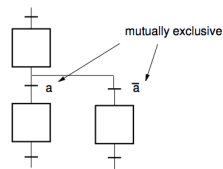


52

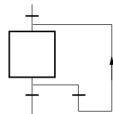
## Control Structures

Alternative paths:

- branches



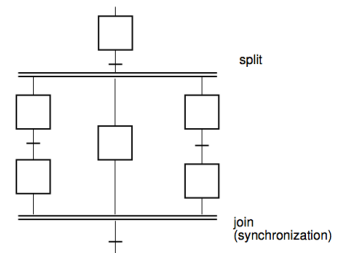
- repetition



53

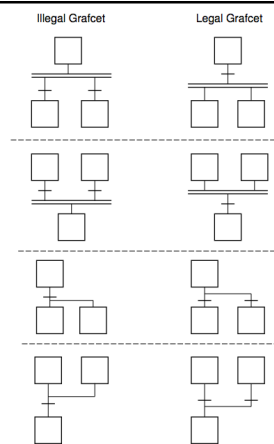
## Control Structures

Parallel paths:



54

## Legal and Illegal structures



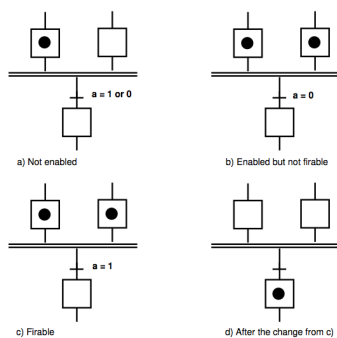
55

## Semantics

1. The initial step(s) is active when the function chart is initiated.
2. A transition is fireable if:
  - all steps preceding the the transition are active (en- abled).
  - the receptivity (transition condition and/or event) of the transition is true.
 A fireable transition must be fired.
3. All the steps preceding the transition are deactivated and all the steps following the transition are activated when a transition is fired
4. All fireable transitions are fired simultaneously
5. When a step must be both deactivated and activated it remains activated without interrupt

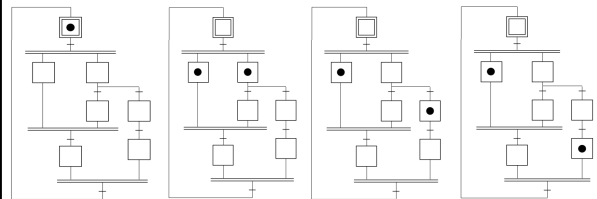
56

## Firing



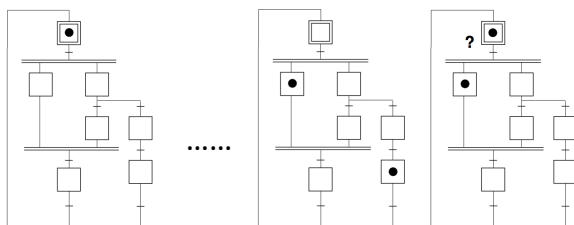
57

## Unreachable Grafcet



58

## Unsafe Grafcet



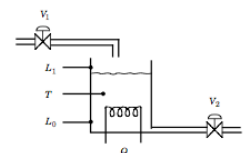
59

## Example

### Specifikationer:

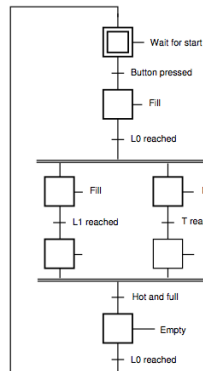
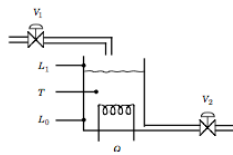
#### Verbal beskrivning

1. Starta sekvensen med att trycka påknappen B
2. Öppna ventil V1 för påfyllning av vatten till övre nivå L1
3. Värm vattnet tills temperaturen är högre än T. Värmning kan starta så fort det finns vatten över nivå L0
4. Töm tanken genom att öppna ventil V2 tills nivån når L0.
5. Stäng ventiler och gå till 1) för att vänta på en ny startorder.



60

## Example



61

## IEC 61131

- IEC standard for programmable controllers (PLCs)
- Several parts, e.g.
  - 61131-3 Programming languages
  - 61131-5 Communications
  - 61131-6 Functional safety
- Adopted by essentially all PLC vendors

62

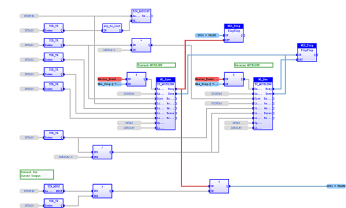
## IEC 61131-3

- Defines 5 PLC programming languages
    - Function block diagrams (FBD)
    - Ladder Diagrams (LD)
    - Structured text (ST)
    - Instruction list (IL)
    - Sequential function chart (SFC), i.e. Grafset
- + how they may interact
- Currently extended with object-orientation

63

## Function block diagrams (FBD)

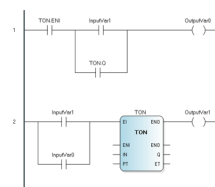
- Graphical data-flow language
- Interconnects function blocks
- Cp. Simulink



64

## Ladder Diagrams

- Ladder logic extended with function blocks



65

## Structured Text

- Block-structured high-level programming language inspired by Pascal
- Iteration loops (WHILE, REPEAT)
- Conditional branches (IF, CASE)
- Functions

[illegible]

66

Instruction List

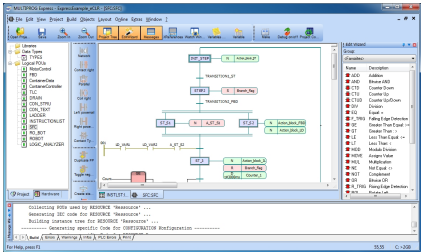
- Low-level textual assembly-like language
- Stack-machine oriented

```
LD      Speed
GT      1000
JMPCN   VOLTS_OK
LD      Volts
VOLTS_OK LD      1
ST      %Q75
```

67

Sequential Function Charts

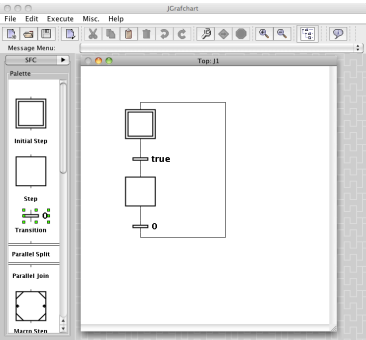
- Grafcet



68

JGrafchart

JGrafchart is a Grafcet/SFC editor and execution environment developed at the Department of Automatic Control, Lund Institute of Technology.



69

JGrafchart

JGrafchart is a Grafcet/SFC editor and execution environment developed at the Department of Automatic Control, Lund Institute of Technology.

JGrafchart supports the following language elements:

- workspace objects
- Steps
- initial steps
- transitions
- parallel splits and parallel joins
- macro steps
- exception transitions

In addition there is support for:

- digital inputs and digital outputs
- analog inputs and analog outputs
- socket inputs and socket outputs
- internal variables (real, boolean, string, and integer)
- action buttons and free text for comments.

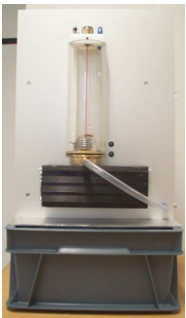
70

JGrafchart

JGrafchart is a Grafcet/SFC editor and execution environment developed at the Department of Automatic Control, Lund Institute of Technology.

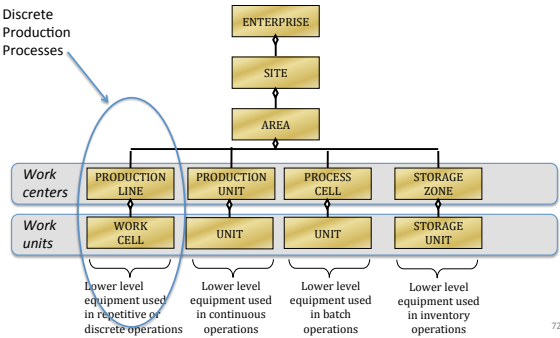
JGrafchart will be used in Laboratory Exercise 1.

A richer description of JGrafchart is contained within Laboratory Exercise 1.



71

Physical Model of an Enterprise



72

