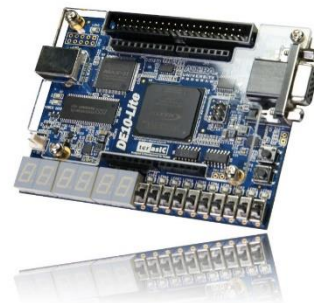


GRAFCET 電路設計

Outline

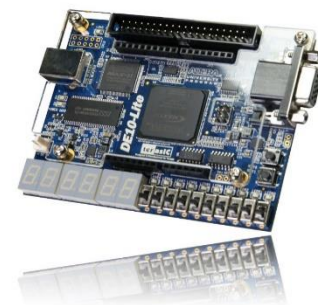
- GRAFCET
 - GRAFCET的組成
 - 基本元件
 - 離散事件模型
 - 平行和分支架構
 - Sub-GRAFCET
 - 平行和分支架構
 - State Transition
 - Concurrent States Transition
 - Branching States Transition
- GRAFCET
離散事件系統設計範例
- 隨堂練習



離散事件系統

Discrete Event System(DES)

- 包含大量的輸入輸出變數和混合sequential和concurrent events的邏輯系統
- 一個良好的離散事件建模工具必須滿足：
 1. 能描述出離散事件系統的大量且連續的狀態。
 2. 必須考慮同時發生情況，且能以簡單明瞭的方式表達。
 3. 通常一個系統的狀態只會受到幾個輸入的影響，且只有一些輸出被改變，因此只描述因輸入的改變而產生的行為。
 4. 可以清楚的瞭解輸入與輸出的行為



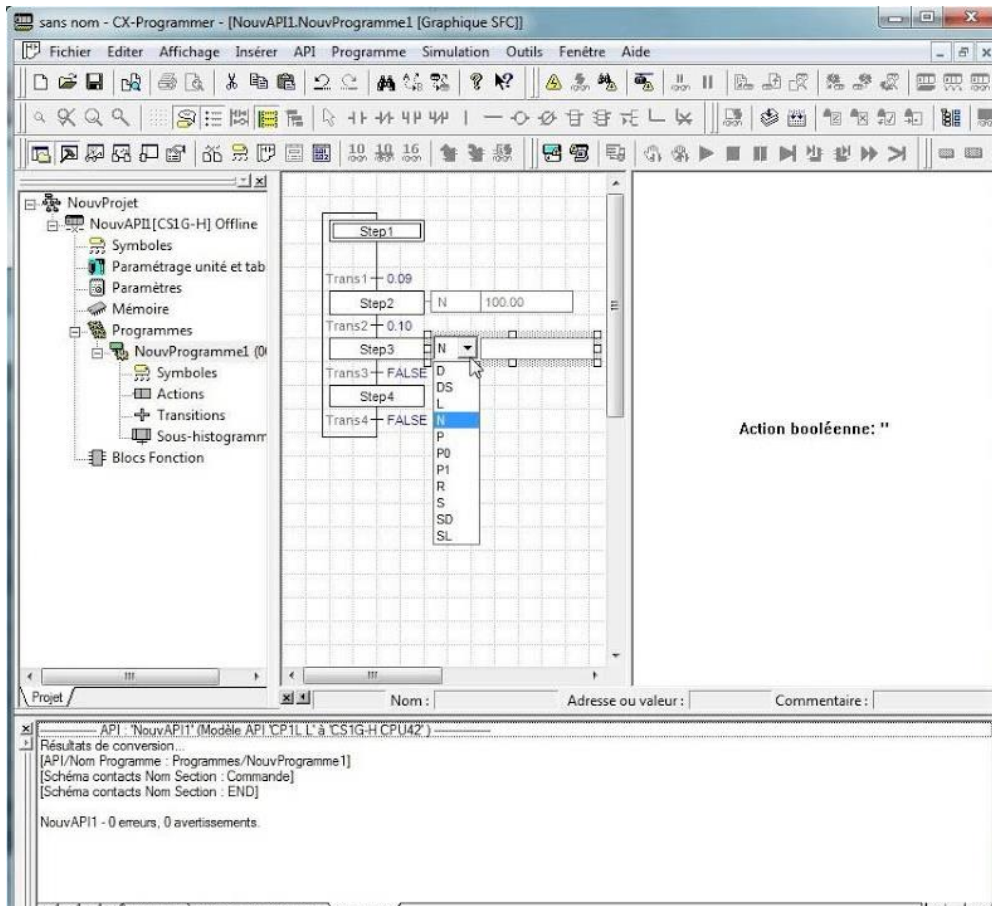
GRAFCET

- Grafcet是由Petri nets 發展而來，用來描述連續行為與同時發生情況的一種圖形化模型。1987年成為國際標準。
- Grafcet多使用在自動化系統，如工業控制器，將離散事件控制器設計在可程式控制器（PLC）上。也運用在資訊系統設計。
- Grafcet可使用在數位系統的順序控制規格描述，能夠很簡潔明瞭的表示多輸入與多輸出的同步動作，簡化離散控制邏輯使其易於處理。



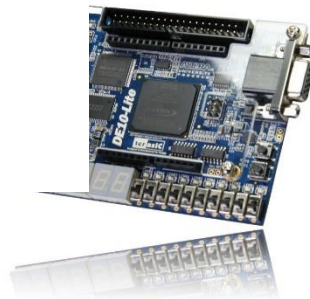
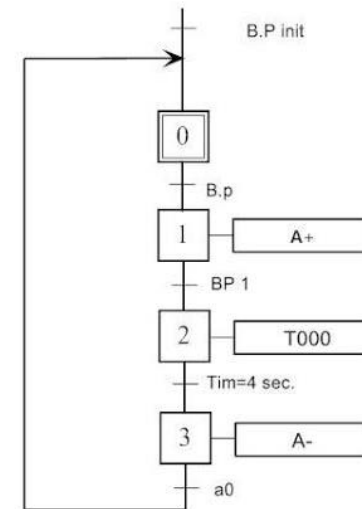
GRAFCET

- OMRON-CX PROGRAMMER-GRAFCET SFC



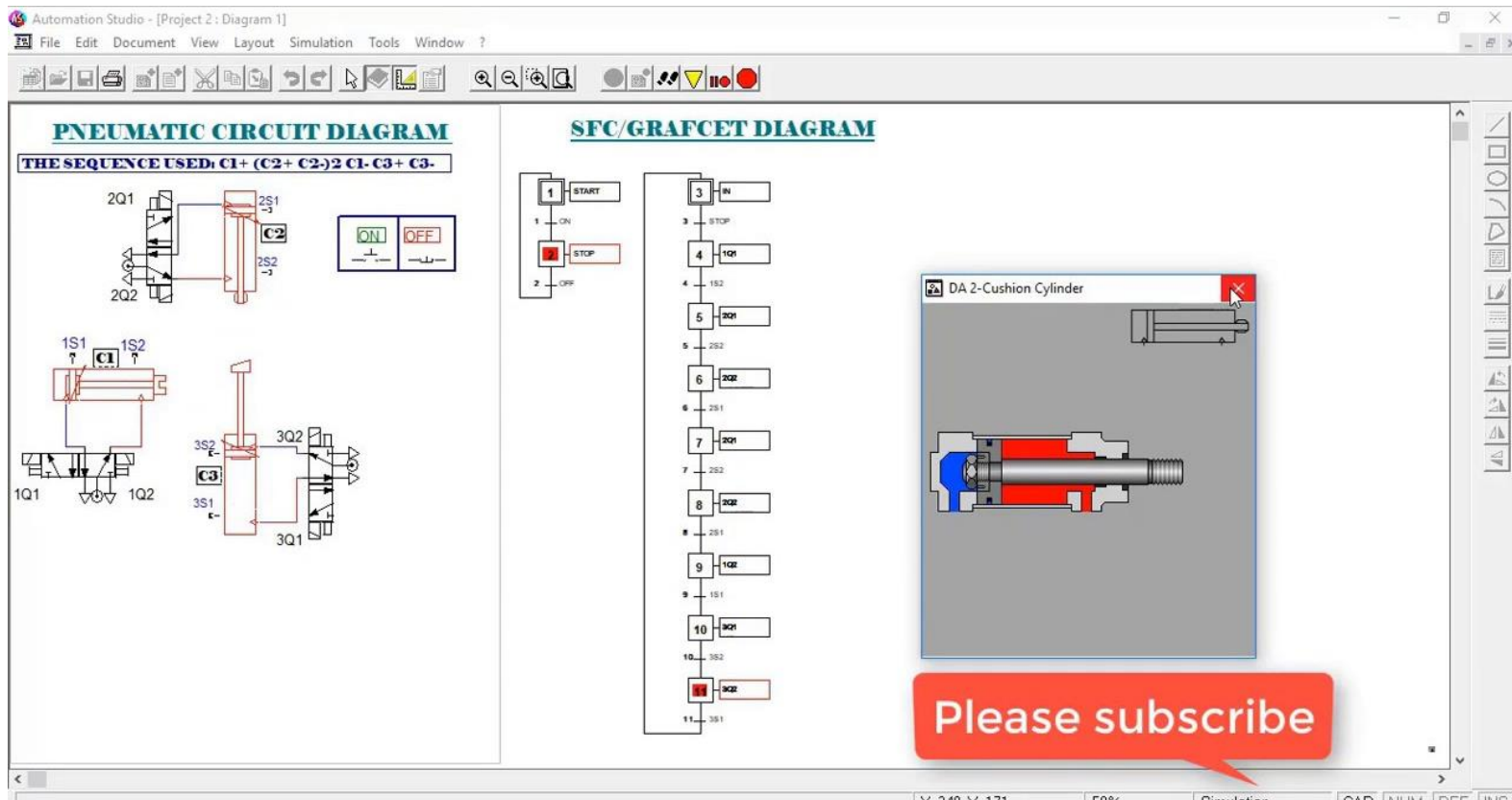
temporisateurs, compteurs, Set, Rset et plein d'autre. est l'instruction Set et Rset. L'instruction Set travaille programme utilisateur à l'exception qu'elle s'auto-ma n'est pas activé.

GRAFCET:

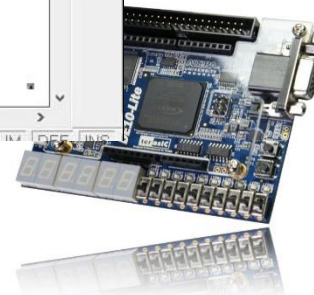


GRAFCET

- Famic Automation studio

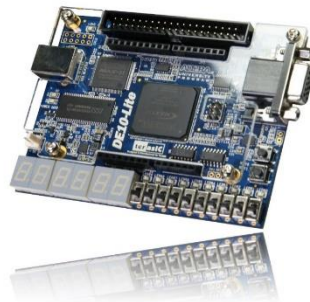
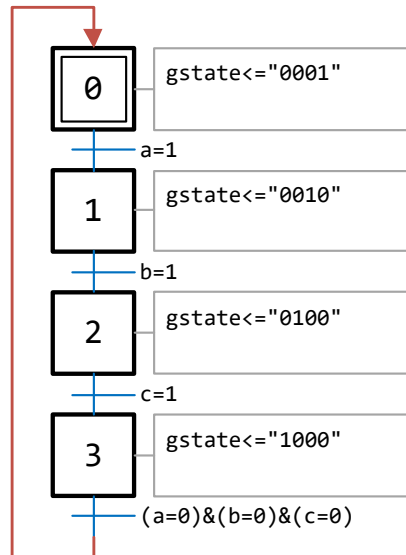


<https://www.youtube.com/watch?v=dW5YdIZP8RM>


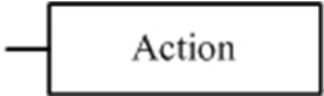

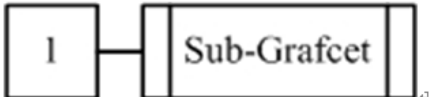
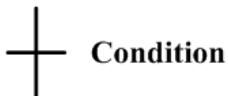
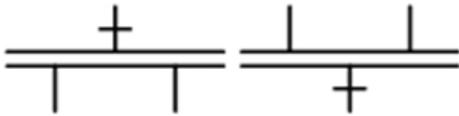
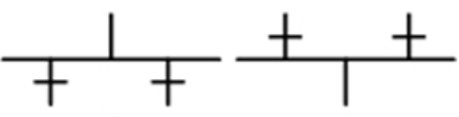


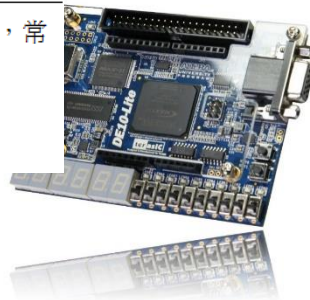
GRAFCET的組成

- 狀態(step/state)
- 轉移條件(transition/condition)
- 有向性的連結(directed connection)
- 一個Grafcet至少包含一個狀態和一個轉移條件以上，連結時由狀態連結到轉移條件或是由轉移條件連結到狀態

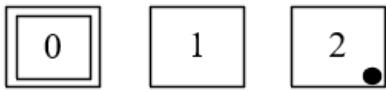

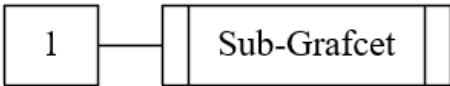
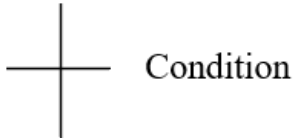
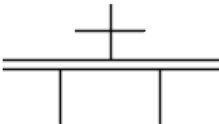
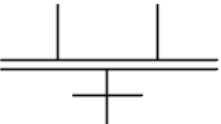
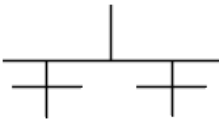
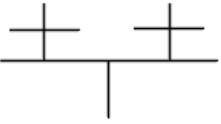


GRAFCET的基本元件

 <p>(a) (b) (c)</p>	<p>基本狀態方塊，其中雙線方塊表示初始狀態，如(a)；其中，狀態方塊中若有黑圓點表示狀態動作中，如(c)；若狀態方塊中無黑圓點表示狀態閒置中，如(b)。</p>
	<p>動作方塊，與狀態方塊連接，方塊內文字用來敘述狀態成立後所需執行之動作。</p>
	<p>狀態與動作方塊連接圖示。</p>
	<p>狀態與 Sub-Grafcet 方塊連接圖示，Sub-Grafcet 方塊表示此動作方塊代表另一個 GRAFCET 模型組成。</p>
	<p>狀態轉移條件圖示，條件以橫線標記，上下線條端點用以連接兩個狀態，橫線右側以文字描述狀態轉移所需條件。</p>
 <p>(d) (e)</p>	<p>(d) Divergence AND 與 (e) Convergence AND 狀態轉移條件圖示，常用以描述 Grafcet 模型中的平行架構。</p>
 <p>(f) (g)</p>	<p>(f) Divergence OR 與 (g) Convergence OR 狀態轉移條件圖示，常用以描述 Grafcet 模型中的分歧架構。</p>

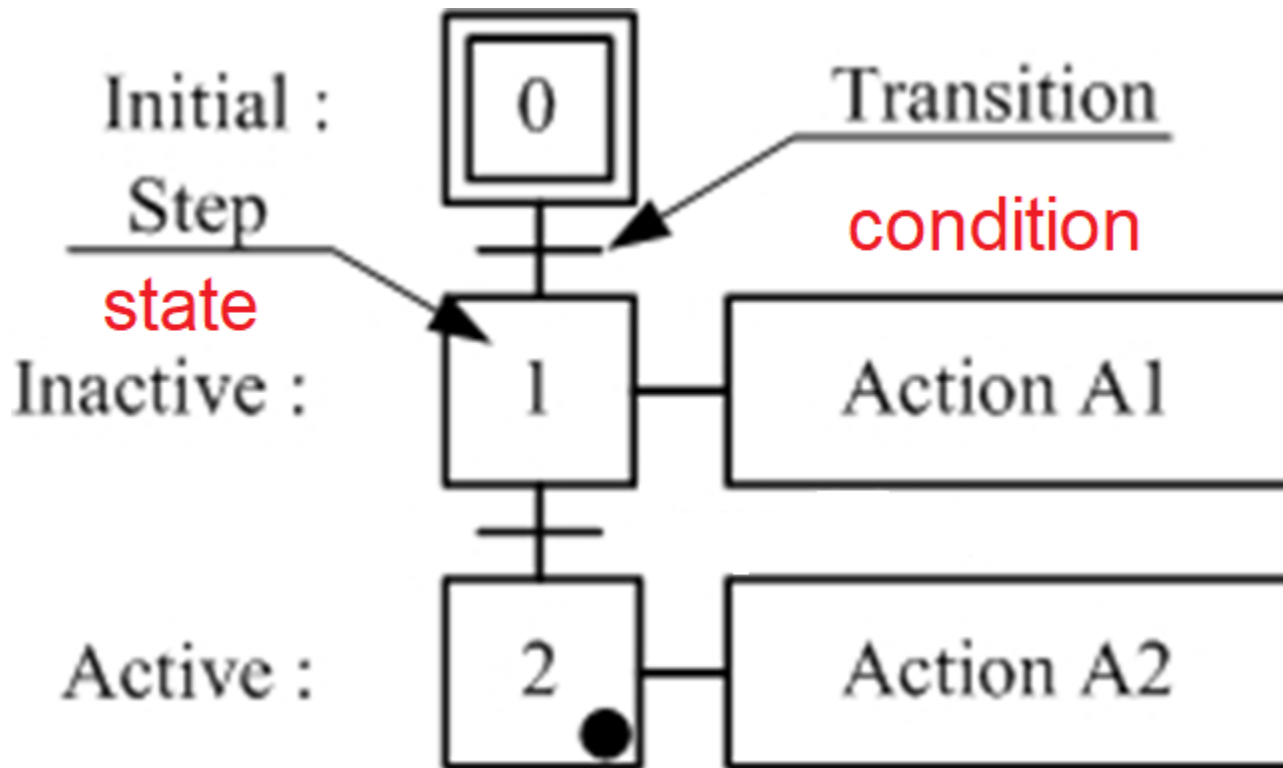


GRA

圖示	說明
 <p>Initial Inactive Active</p>	<p>Initial: 由雙線方塊表示初始狀態。</p> <p>Inactive: 表示該狀態沒有在執行。</p> <p>Active: 代表該狀態正在執行中。</p>
	<p>表示該狀態所要執行的動作。</p> <p>(Action是一個動作)</p>
	<p>表示該狀態所要執行的動作，Sub-Grafcet代表此動作還有另一個再往下切的Grafcet，通常用於該動作太過複雜時使用。</p>
	<p>上下兩端連接狀態，右側連接轉移所需條件。表示要由上面的狀態轉移至下面的狀態時，要符合什麼條件。</p>
 <p>Divergence AND</p>  <p>Convergence AND</p>	<p>Divergence AND: 上面一個狀態同時轉移至下方兩個狀態。</p> <p>Convergence AND: 上面兩個狀態同時轉移至下面一個狀態。</p>
 <p>Divergence OR</p>  <p>Convergence OR</p>	<p>Divergence OR: 上面狀態轉移至下方其中一個狀態。</p> <p>Convergence OR: 上面其中一個狀態轉移至下面狀態。</p>

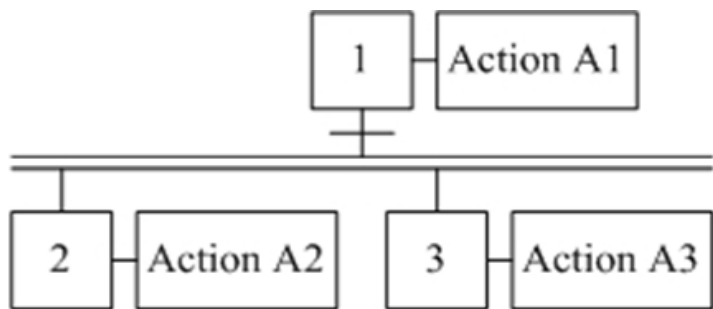


GRAFCET離散事件模型

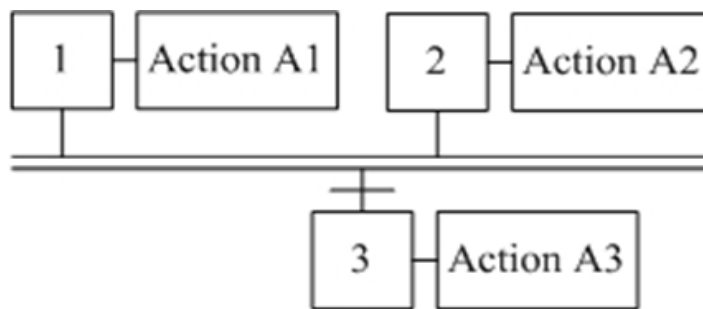


GRAFCET的平行和分支架構

Concurrent架構

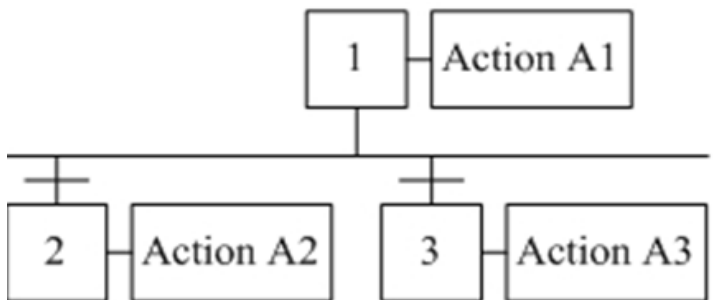


Divergence AND

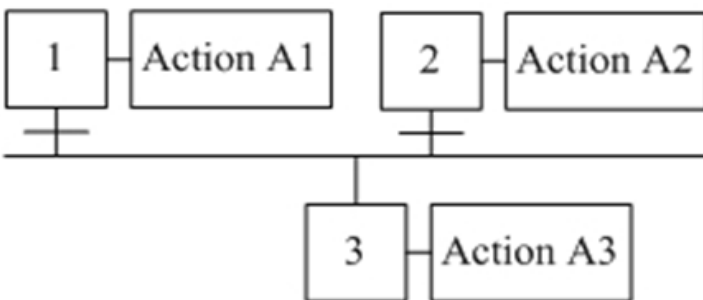


Convergence AND

Branching架構



Divergence OR



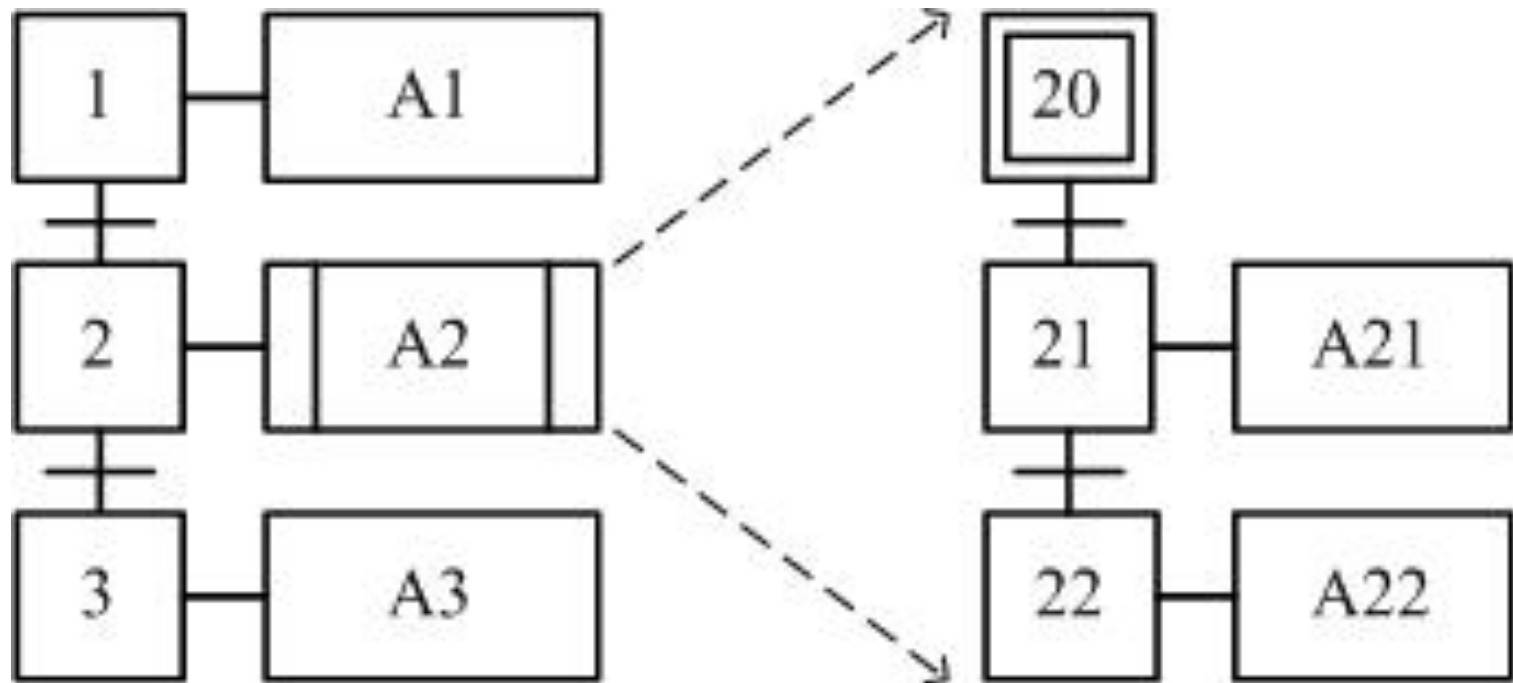
Convergence OR



GRAFCET

Sub-GRAFCET

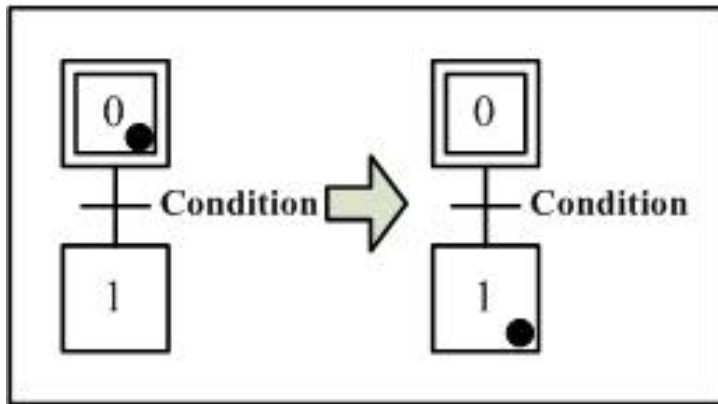
- 階層式離散事件模型：Sub-GRAFCET



GRAFCET

State Transition

- 狀態轉移(State Transition)

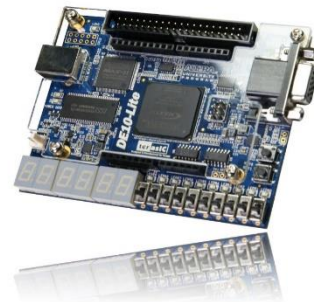
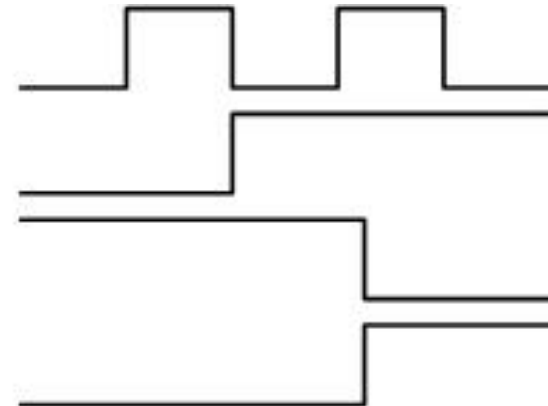


CLK

Condition

State 0

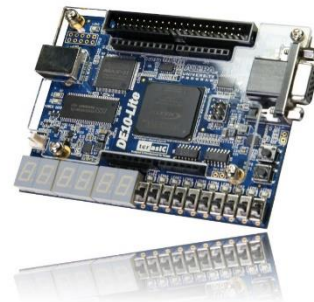
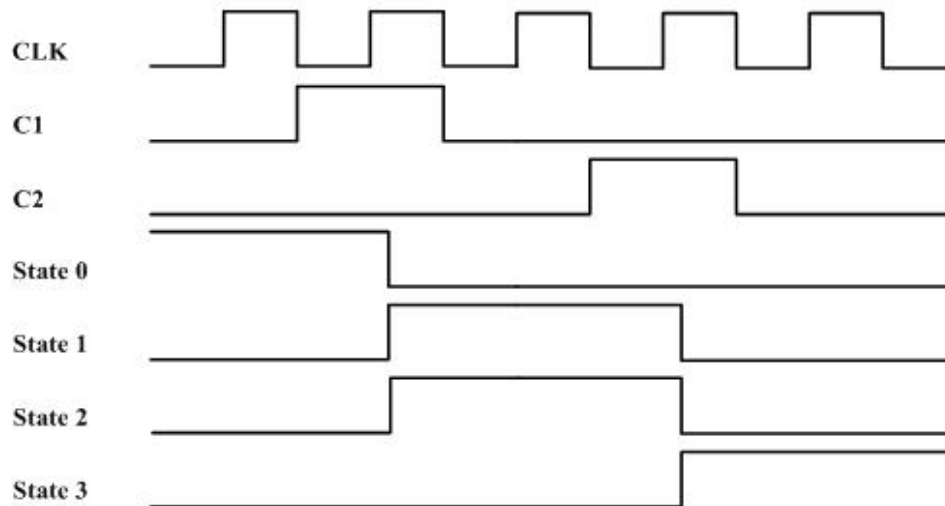
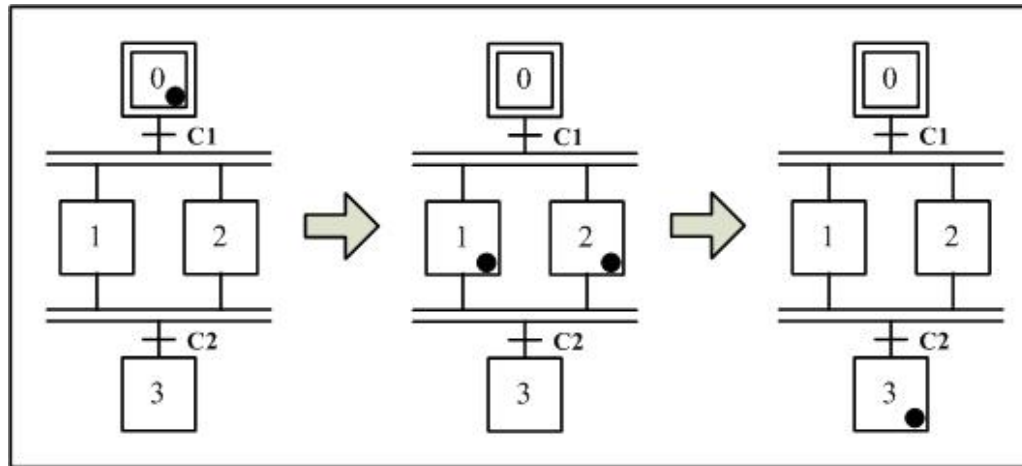
State 1



GRAFCET

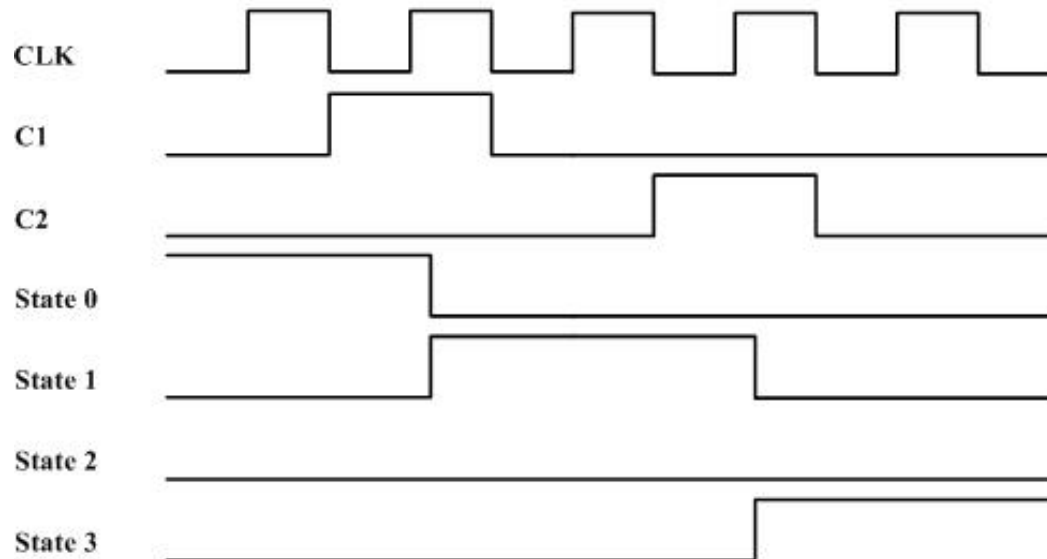
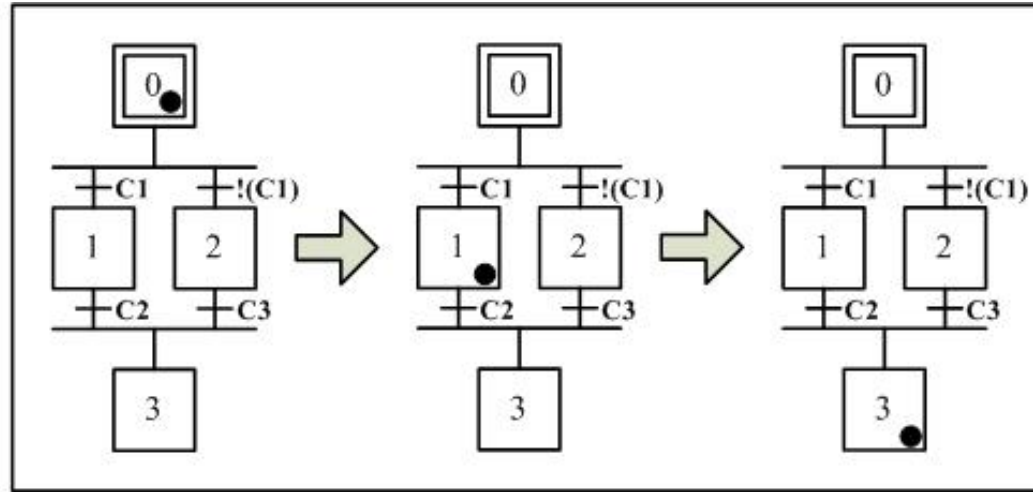
Concurrent States Transition

- Concurrent States Transition



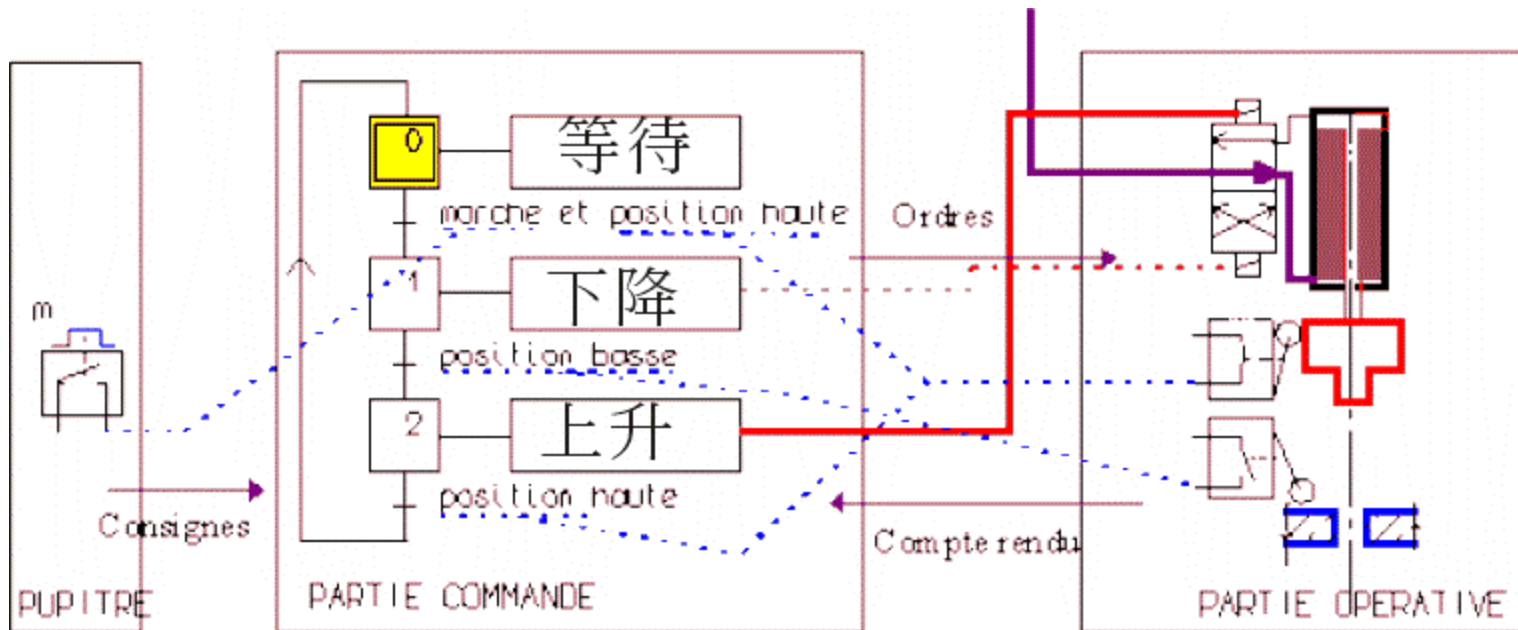
GRAFCET

Branching States Transition

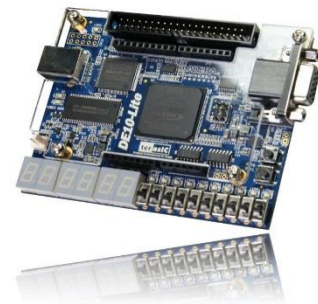


GRAFCET

離散事件系統設計範例 1



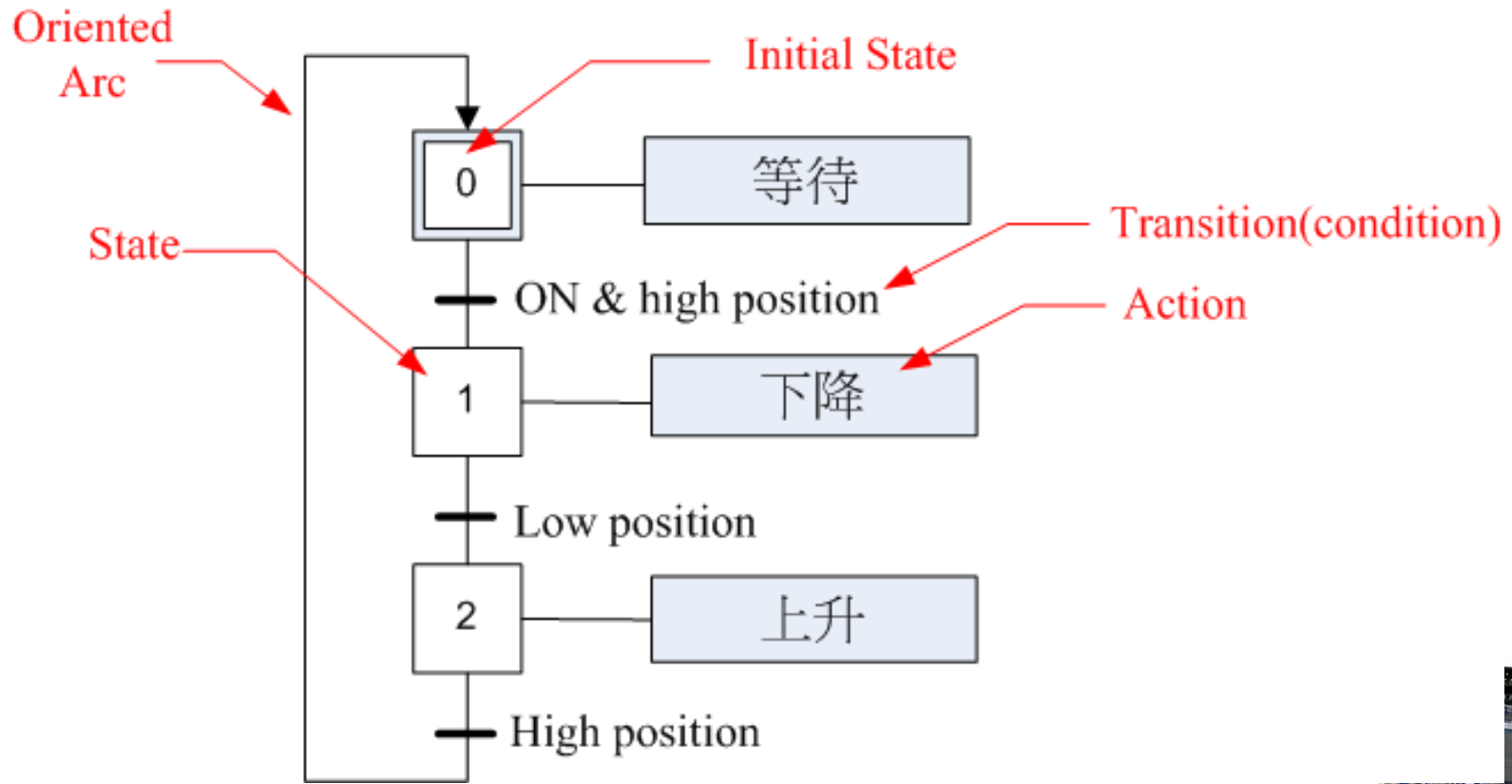
- 整理系統的轉移條件(輸入)和動作(輸出)
- 描述系統的動作流程(離散事件模型)



***沖壓床照片

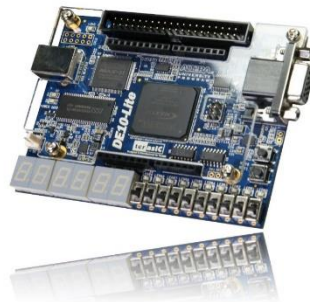
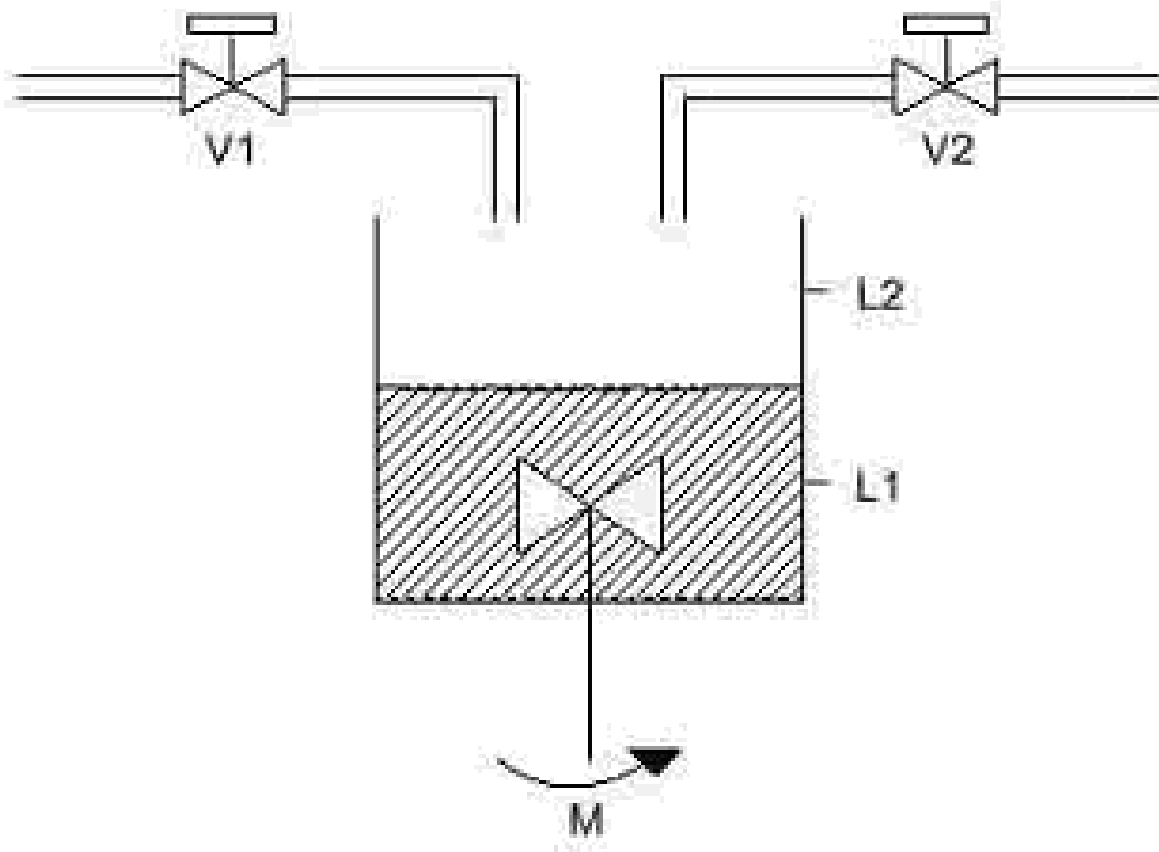
GRAFCET

離散事件系統設計範例 1



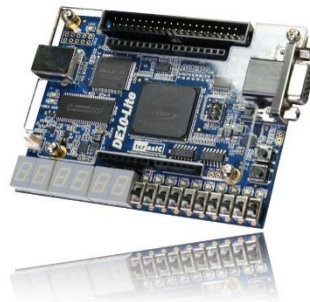
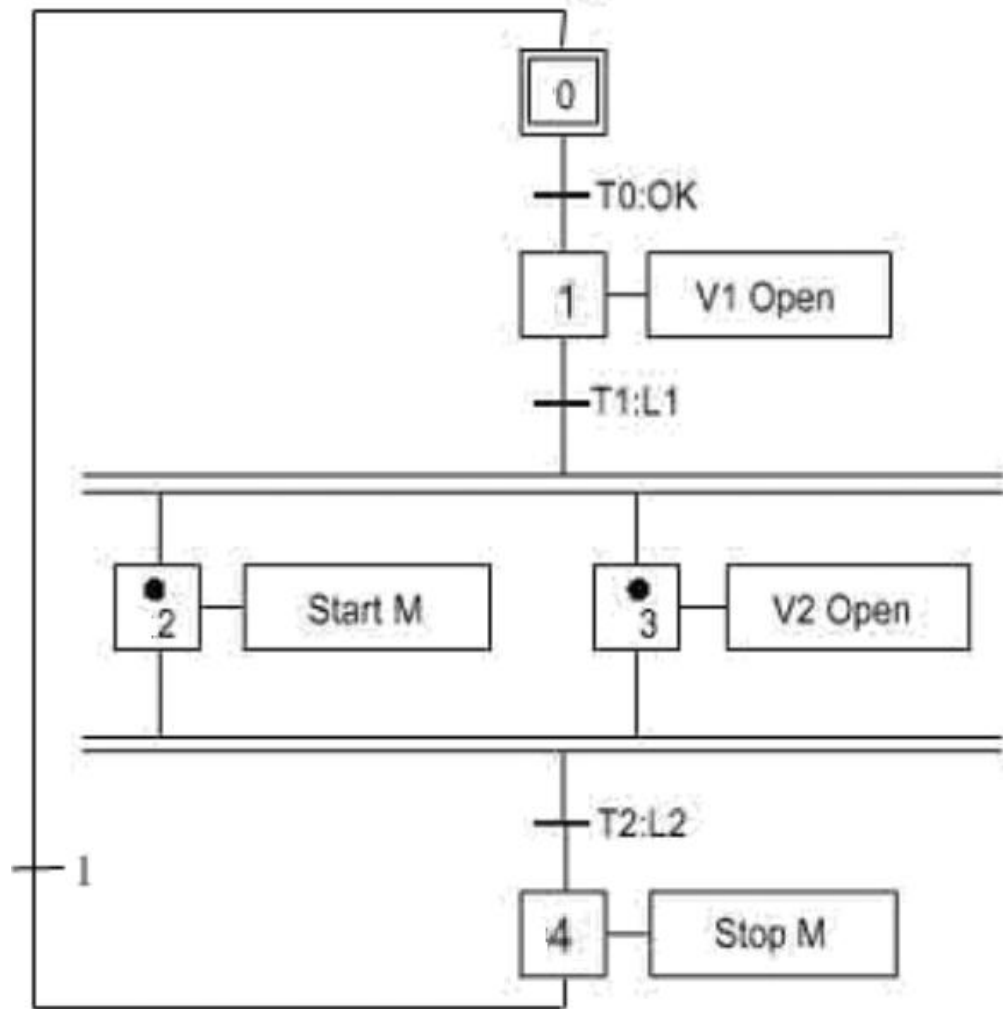
GRAFCET

離散事件系統設計範例 2



GRAFCET

離散事件系統設計範例 2

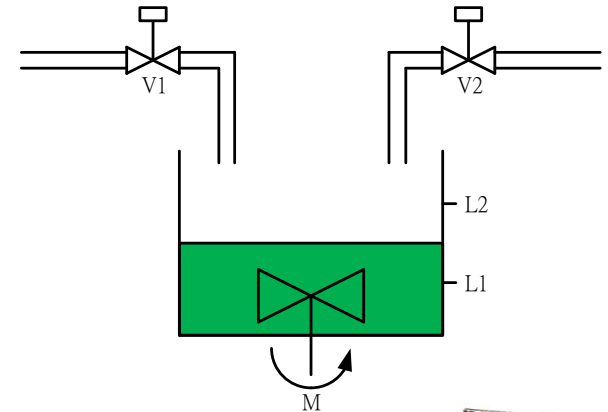


隨堂練習

- 請使用 VHDL 完成下列化學預拌槽控制器電路，並完成紀錄，包括 **GRAFCET** 離散事件建模、**VHDL Source Code**、**模擬波形圖**。

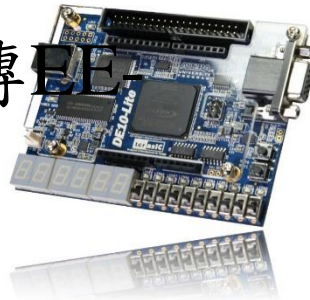
- 說明：啟動後，V1 液體閥開啟當液體加到 L1 時，同時開啟攪拌馬達 M 與 V2 液體閥，最後當液體加到 L2 時全部關閉。

2. Interface:	Sig.	Dir.	bit
	clk	In	1
	rstn	In	1
	en	In	1
	L1	In	1
	L2	In	1
	V1	out	1
	V2	Out	1
	M	out	1



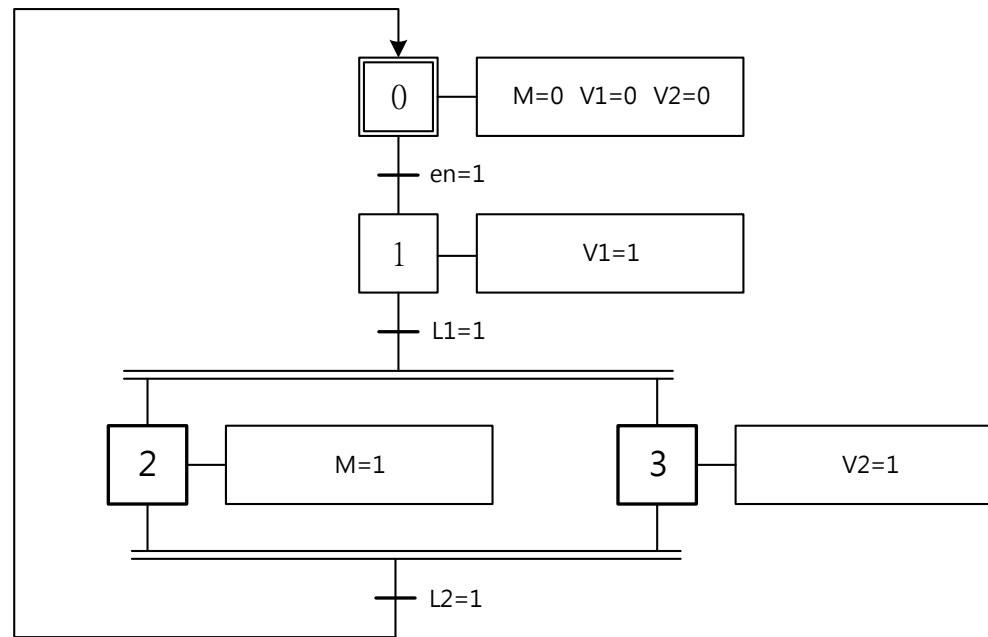
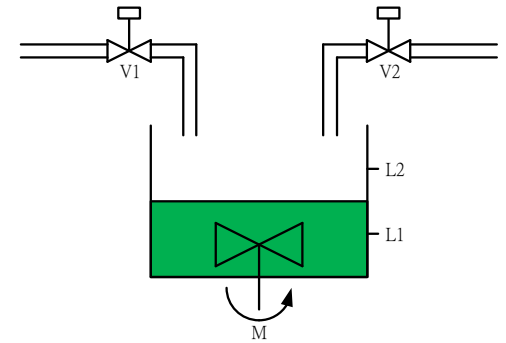
- 本次實驗完成後請，將專案與報告壓縮上傳 PE Class。

- 作業 X_第X組 例如：作業10_第一組



隨堂練習

- GRAFCET離散事件建模



隨堂練習

• VHDL Source Code

```

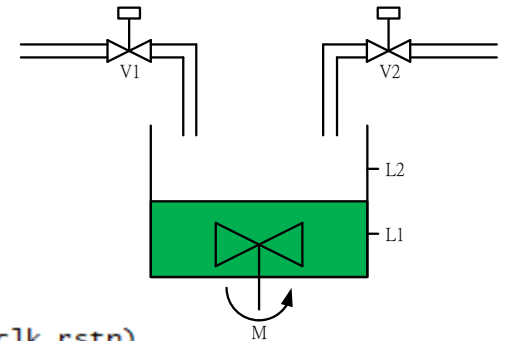
1  library ieee;
2  use IEEE.STD_LOGIC_1164.all;
3  use IEEE.STD_LOGIC_ARITH.all;
4  use IEEE.STD_LOGIC_UNSIGNED.all;
5
6  entity g0 is
7  port(
8      clk      : in std_logic;
9      rstn     : in std_logic;
10     en       : in std_logic;
11     L1       : in std_logic;
12     L2       : in std_logic;
13     V1       : out std_logic;
14     V2       : out std_logic;
15     M        : out std_logic;
16     y0,y1,y2,y3 : out std_logic;
17 end g0;
18
19 architecture rtl of g0 is
20     signal x0,x1,x2,x3 : std_logic;
21     signal w_v1        : std_logic;
22     signal w_v2        : std_logic;
23     signal w_m         : std_logic;
24 begin

```

```

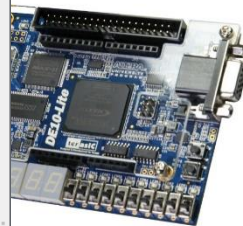
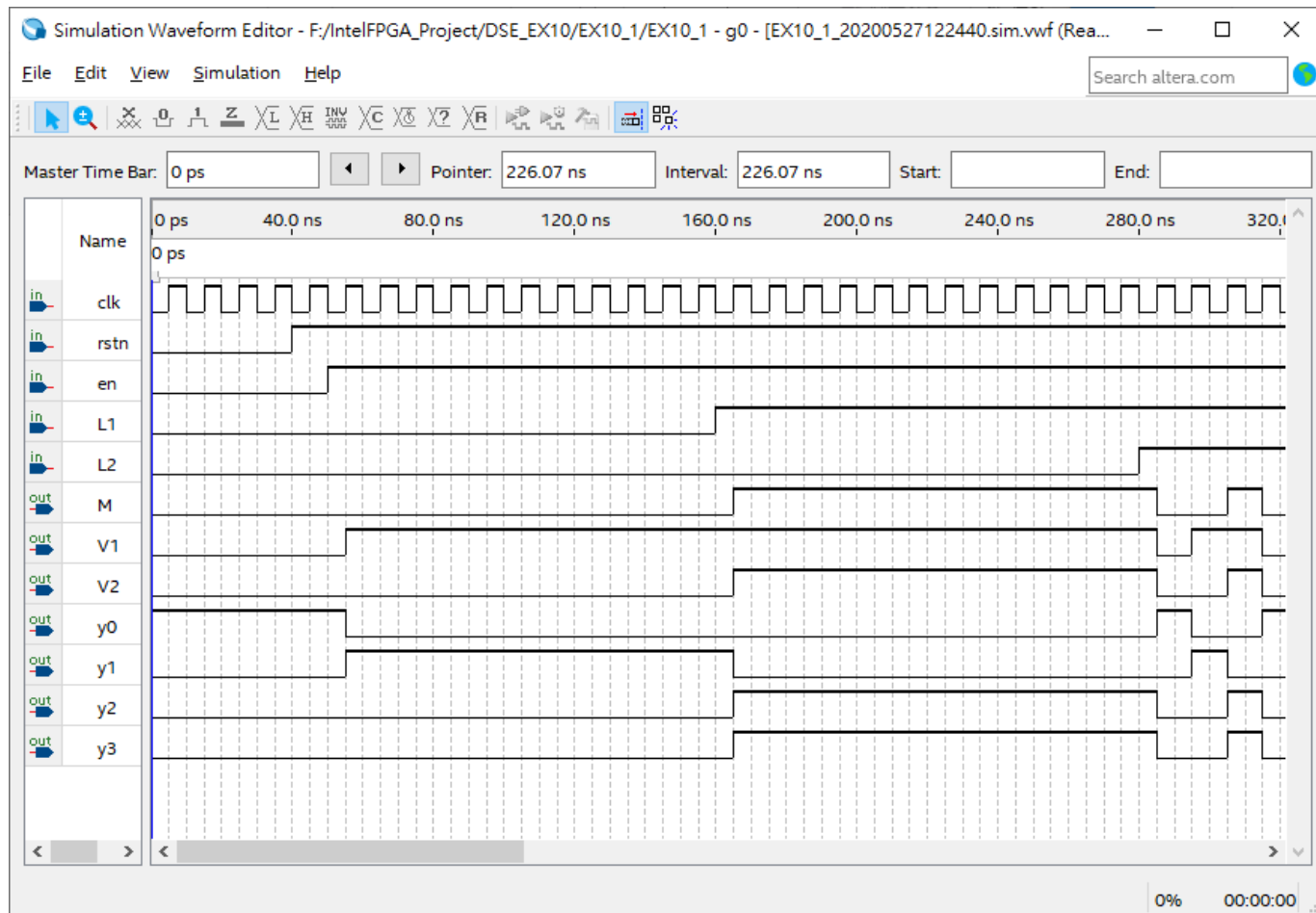
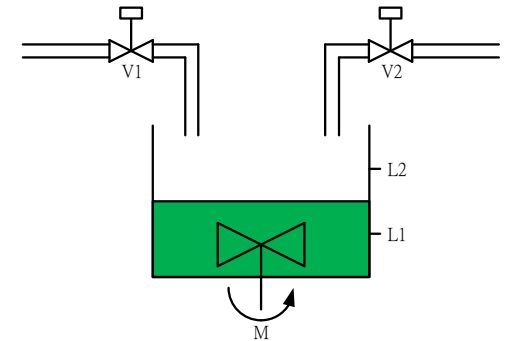
25 GRAFECT : process(clk,rstn)
26 begin
27     if rstn= '0' then
28         x0<='1'; x1<='0'; x2<='0'; x3<='0';
29     elsif clk'event and clk= '1' then
30         if x0='1' and en='1' then
31             x0<='0'; x1<='1';
32         elsif x1='1' and L1='1' then
33             x1<='0'; x2<='1'; x3<='1';
34         elsif x2='1' and x3='1' and L2='1' then
35             x2<='0'; x3<='0'; x0<='1';
36         else null;
37         end if;
38     else null;
39     end if;
40 end process;
41
42 DATA_PATH : process(x0,x1,x2,x3)
43 begin
44     y0<=x0; y1<=x1; y2<=x2; y3<=x3;
45     if x0='1' then
46         w_v1<='0'; w_v2<='0'; w_m<='0';
47     end if;
48     if x1='1' then
49         w_v1<='1';
50     end if;
51     if x2='1' then
52         w_m<='1';
53     end if;
54     if x3='1' then
55         w_v2<='1';
56     end if;
57     V1<=w_v1; V2<=w_v2; M<=w_m;
58 end process;
59
60 END rtl;

```



隨堂練習

- 模擬波形圖



隨堂練習

- RTL Schematic

