Hệ thống điều khiển phân tán

Chương 9: Chuẩn IEC 61131-3

13.09.06

Chương 9: Chuẩn IEC 61131-3

- Giới thiệu chung về IEC 61131
- Tiến trình chuẩn hóa IEC 61131
- Mô hình phần mềm
- Biến và kiểu dữ liệu
- Tổ chức chương trình
- Ngôn ngữ lập trình

IEC 61131 là gì?

- Tập chuẩn phần mềm quan trọng nhất cho các thiết bị điều khiển công nghiệp có khả năng lập trình (PLC, DCS, Soft PLC,...)
- Bao gồm nhiều phần:
  - Phần 1 (General Information)
  - Phần 2 (Equipment requirements)
  - **Phần 3 (Programming languages)**
  - Phần 4 (Guidelines for users)
  - Phần 5 (Communication)
  - Phần 7 (Fuzzy Control)
  - ...
- Hầu hết các hệ PLC và DCS hiện đại đều hỗ trợ chuẩn IEC 61131-3

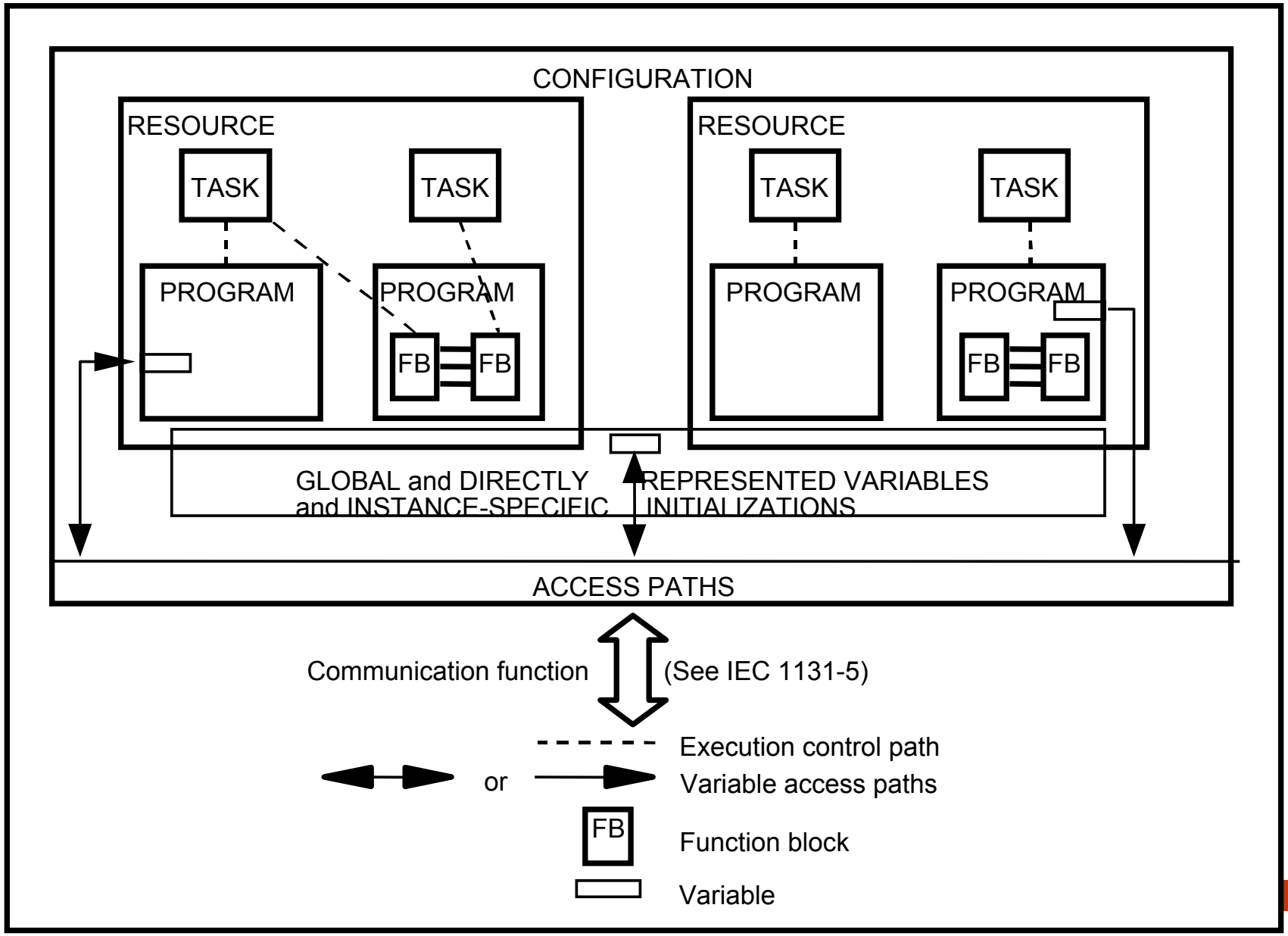
Tiến trình chuẩn hóa IEC 61131

- 1977: IEC 848
- 1979: Bắt đầu soạn bản thảo IEC 1131
- 1982: Hoàn thành bản thảo đầu tiên (5 nhóm làm việc)
- 1983: DIN 19239 PLC-Programming
- 1992: Chuẩn hóa quốc tế IEC 1131-1 và 1131-2
- 1993: Chuẩn hóa quốc tế IEC 1131-3
- 1995: Chuẩn hóa quốc tế IEC 1131-TR4
- 1994-1997: Đính chính IEC 1131-3 (Corrigendum)
- 1996-1999: Sửa đổi, bổ sung (Amendment)
- Từ 2000 -> IEC 61131-3 2nd Edition

Các tiến bộ của IEC 61131-3

- Các yếu tố cấu hình thống nhất (CONFIGURATION, TASK, RESOURCE), mô hình TASK và RESOURCE thích hợp cho nhiều hệ thống khác nhau
- Mô hình phần mềm thống nhất, hiện đại, với các khối tổ chức chương trình hợp lý (PROGRAM, FUNCTION BLOCK, FUNCTION)
- Các ngôn ngữ lập trình thống nhất, phát triển trên cơ sở chuẩn hóa các ngôn ngữ hiện có quen thuộc
- Các kiểu dữ liệu đa dạng, khả mở
- Một thư viện các hàm và khối chức năng chuẩn
- Bước đầu có ý tưởng hướng đối tượng
- Một mô hình giao tiếp thống nhất.

Mô hình phần mềm



Các yếu tố cấu hình

- Cấu hình (CONFIGURATION):
  - Tương ứng cho cả hệ PLC, có thể gồm nhiều CPU ghép nối
  - Mỗi PLC tại một thời điểm bất kỳ chỉ có một cấu hình.
  - Bao gồm một hay nhiều tài nguyên
- Tài nguyên (RESOURCE)
  - Tương ứng cho một CPU với các vào/ra và HMI (đơn giản) tương ứng
  - Bao gồm một hoặc nhiều chương trình hoạt động dưới sự điều khiển của một hoặc nhiều tác vụ
- Tác vụ (TASK)
  - Tác vụ tuần hoàn (*Periodic Task*)
  - Tác vụ sự kiện, task đơn (*Event Task, Single Task*)
  - Tác vụ rỗi (*Idle Task*)
- Biến toàn cục (Global Variables)
- Lối truy nhập (Access Path)

Các kiểu dữ liệu cơ bản

Kiểu Bool	BOOL
Kiểu nguyên có dấu	SINT, INT, DINT, LINT, INT
Kiểu nguyên dương	USINT, UINT, UDINT, ULINT
Số thực	REAL, LREAL
Khoảng thời gian	TIME
Ngày tháng	DATE
Thời gian trong ngày	TIME_OF_DAY, TOD
Ngày tháng và thời gian	DATE_AND_TIME, DT
Chuỗi ký tự	STRING, WSTRING
Chuỗi bit	BYTE, WORD, DWORD, LWORD

Các kiểu dữ liệu dẫn xuất

- Dẫn xuất trực tiếp:

  TYPE RU_REAL : REAL ; END_TYPE

- Liệt kê:

  TYPE ANALOG_SIGNAL_TYPE : (SINGLE_ENDED, DIFFERENTIAL) ;
  END_TYPE

- Dãy con:

  TYPE ANALOG_DATA : INT (-4095..4095) ; END_TYPE

- Mảng:

  TYPE ANALOG_16_INPUT_DATA : ARRAY [1..16] OF ANALOG_DATA ;
  END_TYPE

- Cấu trúc:

  TYPE ANALOG_CHANNEL_CONFIGURATION: STRUCT
  RANGE : ANALOG_SIGNAL_RANGE ;
  MIN_SCALE : ANALOG_DATA ;
  MAX_SCALE : ANALOG_DATA ;
  END_STRUCT;

Các kiểu dữ liệu tổng quát

```
ANY
  ANY_DERIVED
  ANY_ELEMENTARY
    ANY_MAGNITUDE
      ANY_NUM
        ANY_REAL
          LREAL
          REAL
        ANY_INT
            LINT, DINT, INT, SINT
            ULINT, UDINT, UINT, USINT
      TIME
    ANY_BIT
      LWORD, DWORD, WORD, BYTE, BOOL
    ANY_STRING
      STRING
      WSTRING
    ANY_DATE
      DATE_AND_TIME
      DATE, TIME_OF_DAY
```

Khai báo biến

- Kiểu của biến:
  - Kiểu cơ bản,
  - Kiểu dẫn xuất,
  - Kiểu tổng quát
  - Khối chức năng,
  - Khối chương trình
- Từ khóa
  - Bắt đầu với VAR, VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR_EXTERNAL, VAR_GLOBAL, VAR_ACCESS, VAR_TEMP hoặc VAR_CONFIG
  - Có thể kèm theo thuộc tính RETAIN, NON_RETAIN, CONSTANT, AT
  - Kết thúc với END_VAR

Ký hiệu biến trực tiếp

- Tiền tố
  - I — Biến đầu vào (Input)
  - Q — Biến đầu ra (Output)
  - M — Biến nhớ (Memory)
  - X hoặc không ghi — 1 bit, mặc định là BOOL
  - B — 8 bit, mặc định là BYTE
  - W — 16 bit, mặc định là WORD
  - D — 32 bit, mặc định là DWORD
  - L — 64 bit, mặc định là LWORD
- Ví dụ:
  - %QX75, %Q75 — Bit ra vị trí 75
  - %IW215 — Từ vào vị trí 215
  - %QB7 — Byte vào vị trí 7
  - %MD48 — Từ đúp vào tại vị trí ô nhớ 48
  - %IW2.5.7.1 — Từ vào kênh 1, slot 7, rack 5, station 2
  - %Q* — Đầu vào chưa định vị trí

Ví dụ khai báo biến

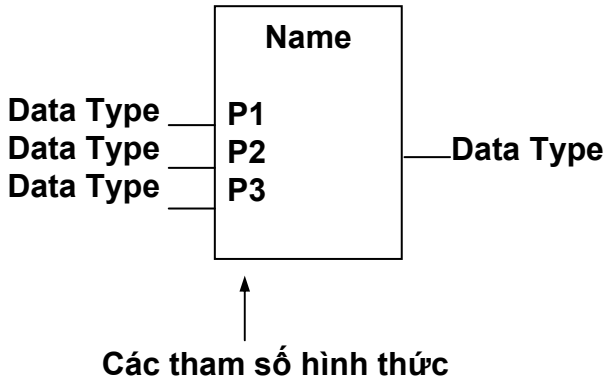
```
VAR RETAIN
   AT %IW6.2     : WORD;
   AT %MW6       :INT;
END_VAR
VAR_GLOBAL
   LIM_SW_S5 AT %IX27    :BOOL = TRUE;
   CONV_START AT %QX25 :BOOL;
   TEMPERATURE AT %IW28: INT;
   C2 AT %Q* : BYTE;
END_VAR
VAR INARY AT %IW6 :ARRAY [0..9] OF INT; END_VAR
VAR
   CONDITION_RED : BOOL = 1;
   IBOUNCE : WORD = 16#FF00;
   MYDUB :DWORD;
   AWORD, BWORD, CWORD : INT = 8;
   MYSTR: STRING[10];
END_VAR
```

Các khối tổ chức chương trình (POU)

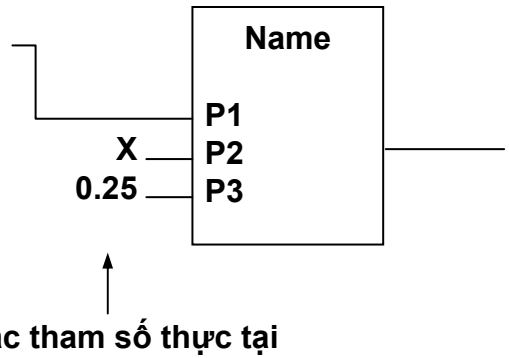
- Hàm (FUNCTION)
  - Tương tự hàm PASCAL, có thể nhiều vào, chính xác một ra
  - Như một hệ tĩnh, không có trạng thái
  - Có giá trị sử dụng lại
- Khối chức năng (FUNCTION BLOCK)
  - Tương tự lớp trong lập trình HĐT, có thể có nhiều đầu ra
  - Như một hệ động, có trạng thái
  - Phân biệt giữa kiểu và thể nghiệm theo ngữ cảnh
  - Có giá trị sử dụng lại
- Chương trình (PROGRAM)
  - Về cơ bản giống như khối chức năng
  - Truy cập được các biến trực tiếp (biến vào/ra, biến nhớ trực tiếp) và các biến toàn cục
  - Không có giá trị sử dụng lại

Khai báo và sử dụng hàm

KHAI BÁO HÀM



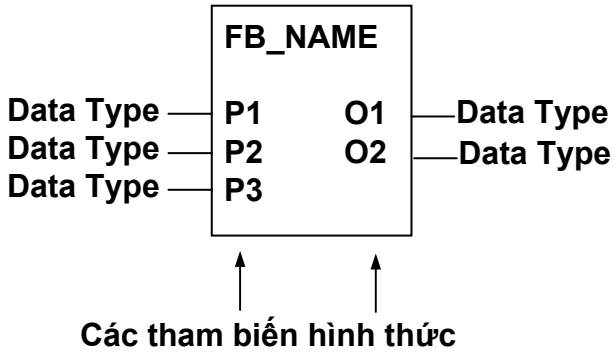
SỬ DỤNG HÀM



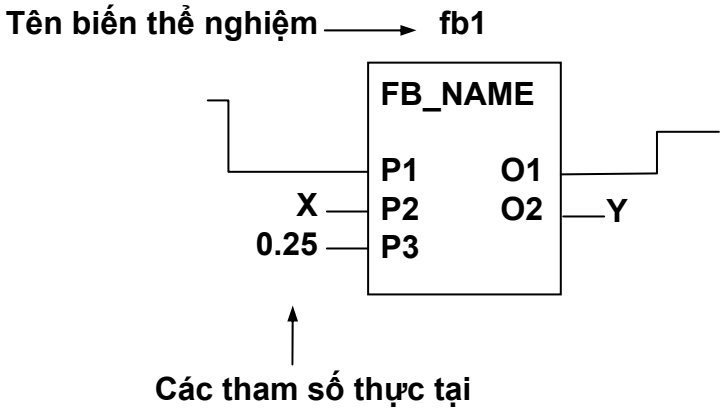

```
(* Khai báo hàm *)
FUNCTION fct1 : REAL
  VAR_INPUT
    a, b: REAL;
    c  : REAL:= 1.0;
  END_VAR
  fct1 := a*b/c;
END_FUNCTION

(* Gọi hàm *)
...
y := fct1(a:= x, b:= 2.0);
...
```

Khai báo và sử dụng khối chức năng

KHAI BÁO KHỐI CHỨC NĂMG



SỬ DỤNG KHỐI CHỨC NĂNG




```
FUNCTION_BLOCK Example
VAR_INPUT
   X :  BOOL;
   Y :  BOOL;
END_VAR
VAR_OUTPUT
   Z :  BOOL;
END_VAR
VAR
   INTERNAL_STATE: BOOL;
END_VAR
  (* statements of functionblock body *)
END_FUNCTION_BLOCK
```

Các ngôn ngữ lập trình

- Các ngôn ngữ lập trình văn bản (textual languages):
  - Instruction List (IL) : Một dạng hợp ngữ
  - Structured Text (ST): Giống PASCAL
  - Các thành phần SFC có thể sử dụng phối hợp
- Các ngôn ngữ đồ họa (graphical languages):
  - Ladder Diagram (LD): Giống mạch rơ le
  - Funtion Block Diagram (FBD): Giống mạch nguyên lý
  - Sequential Funtion Charts (SFC): Xuất xứ từ mạng Petri/Grafcet

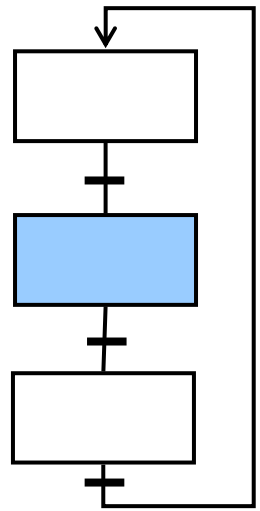
Instruction List (IL)

```
LD    A
ANDN  B
ST    C
```

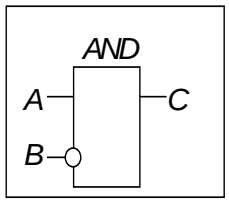
Structured Text (ST)

```
C:= A AND NOT B
```

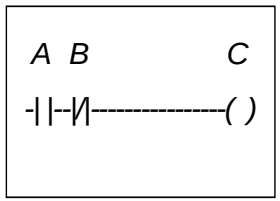
Sequential Function Charts



Function Block Diagram (FBD)



Ladder Diagram (LD)



Các ngôn ngữ văn bản: IL và ST

- Các yếu tố chung:

```
TYPE...END_TYPE
VAR...END_VAR
VAR_INPUT...END_VAR
VAR_OUTPUT...END_VAR
VAR_IN_OUT...END_VAR
VAR_EXTERNAL...END_VAR
VAR_TEMP...END_VAR
VAR_ACCESS...END_VAR
VAR_GLOBAL...END_VAR
VAR_CONFIG...END_VAR
FUNCTION ... END_FUNCTION
FUNCTION_BLOCK...END_FUNCTION_BLOCK
PROGRAM...END_PROGRAM
STEP...END_STEP
TRANSITION...END_TRANSITION
ACTION...END_ACTION
```

Instruction List (IL)

Cú pháp câu lệnh

```
NHÃN     TOÁN TỬ/HÀM     TOÁN HẠNG        CHÚ THÍCH
START:   LD              %IX1             (* PUSH BUTTON *)
         ANDN            %MX5             (* NOT INHIBITED *)
         ST              %QX2             (* FAN ON *)
         LD              2#00010001
         ST              %QB3
```

Lệnh phức hợp

```
AND(                          AND(    %IX1
LD      %IX1      hoặc        OR      %IX2
OR      %IX2                  )
)
```

Accu đa năng: chứa "giá trị tức thời"

- Thích hợp với các kiểu dữ liệu khác nhau
- Mã thực hiện cụ thể do trình biên dịch tạo ra
- Chuẩn không qui định về các cờ trạng thái accu

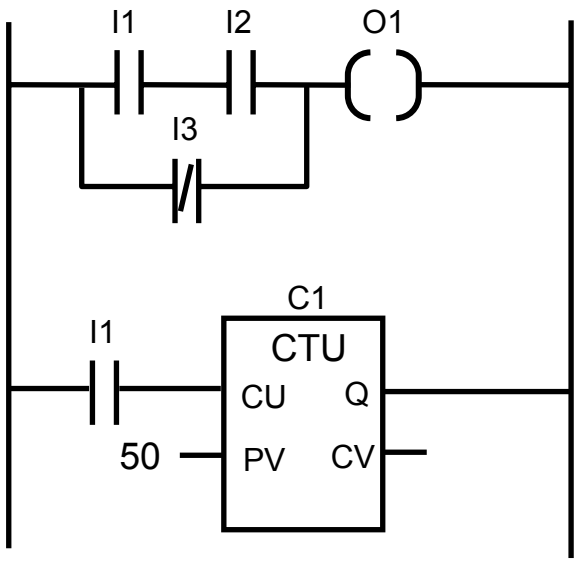
Structured Text (ST)

- Ngôn ngữ hoàn toàn mới, dựa trên PASCAL/C
- Ưu điểm: Đơn giản, mạnh
  - Lập trình ở mức cao
  - Dễ mô tả nhiệm vụ điều khiển
  - Lập trình có cấu trúc
  - Các lệnh điều khiển chương trình (IF, WHILE, FOR,..)
- Nhược điểm: Mã chậm, lớn
  - Phụ thuộc nhiều vào chất lượng của trình biên dịch
  - Không phải hệ PLC/DCS nào cũng hỗ trợ
- Lựa chọn hay không?
  - Qui mô ứng dụng
  - Tỉ lệ đầu tư phần cứng/phát triển phần mềm
  - Điều khiển đơn giản hay điều khiển cao cấp

Các ngôn ngữ đồ họa: LD, FBD và SFC

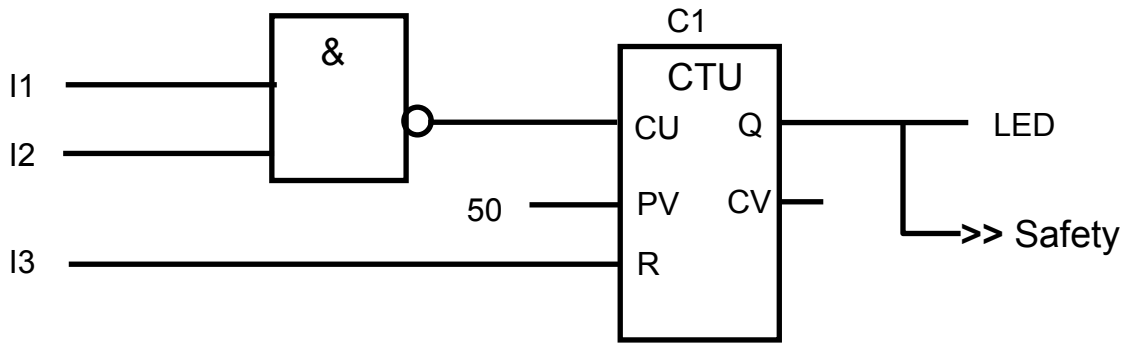
- Các yếu tố chung:
  - Ký hiệu mô tả các khối và đường nét:
  - Hướng của các dòng trong mạng
    - Power flow
    - Signal flow
    - Activity flow
  - Đánh giá mạng (network evaulation)
  - Các yếu tố điều khiển thực thi
    - Các ký hiệu nhảy
    - Các ký hiệu kết thúc
- Lựa chọn ngôn ngữ phù hợp:
  - LD cho mạch điều khiển logic
  - FBD cho điều khiển tương tự (ĐK quá trình) và điều khiển logic
  - SFC cho điều khiển trình tự, phối hợp sử dụng LD và FBD

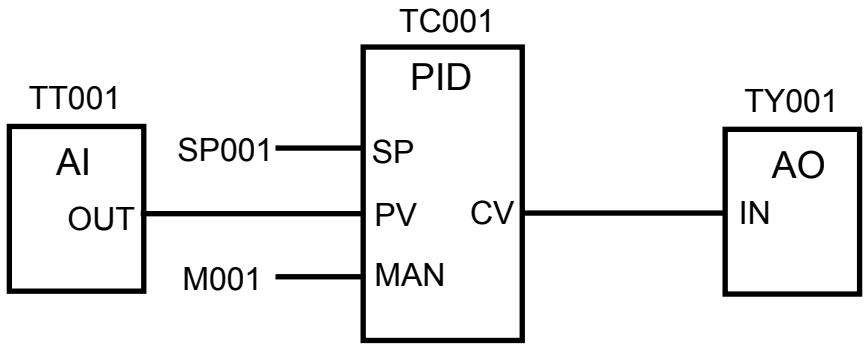
Ladder Diagram



| |	Tiếp điểm thường mở (NO)
|/|	Tiếp điểm thường đong (NC)
|P|	Tiếp điểm nhận biết sườn xung lên
|N|	Tiếp điểm nhận biết sườn xung xuống
( )	Cuộn dây (đầu ra)
(/)	Cuộn dây âm (đầu ra nghịch đảo)
(S)	Cuộn dây đặt
(R)	Cuộn dây xoá
(P)	Cuộn dây cảm nhận sườn xung lên
(N)	Cuộn dây cảm nhận sườn xung xuống

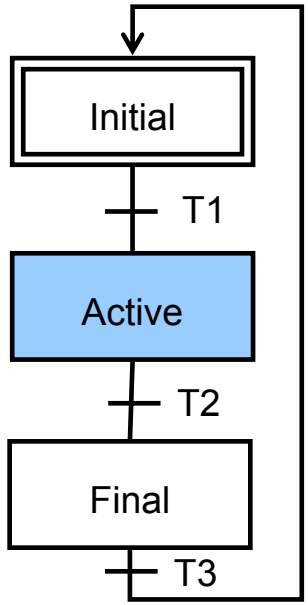
Function Block Diagram



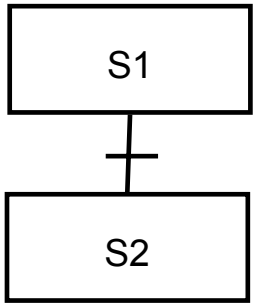


Sequential Function Chart (SFC)

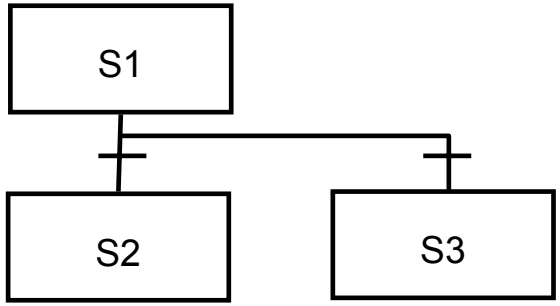
- Step: Một bước thực hiện trong điều khiển trình tự
  - Có thể bao gồm nhiều hành động đi kèm
  - Có ít nhất một bước tích cực
  - Trạng thái hệ thống được xác định qua các bước tích cực
- Transition: Chuyển tiếp, được thực hiện khi điều kiện chuyển tiếp thỏa mãn
  - Lập trình bằng ST, FBD, LD hoặc IL
- Action: Hành động đi với một bước
  - Nằm trong một "Action Block"
  - Được kiểm soát thực thi qua các "Qualifier"
  - Lập trình bằng ST, FBD, LD hoặc IL



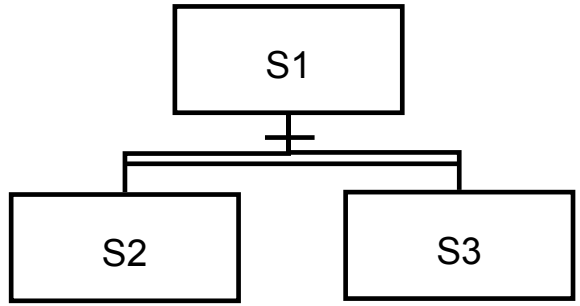
Các loại chuyển tiếp SFC



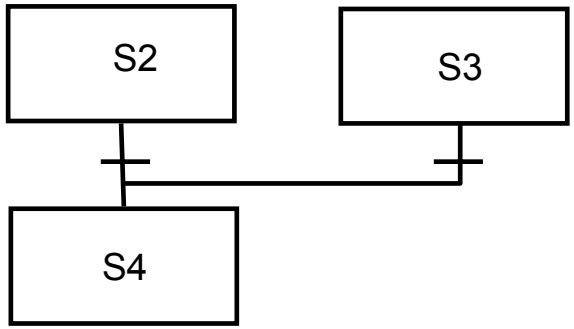
a) Đơn giản



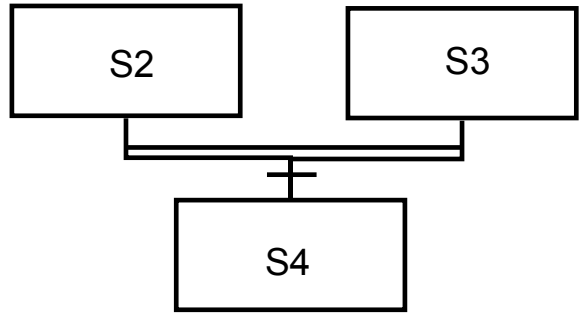
b) Phân nhánh cạnh tranh
(phân nhánh OR)



c) Phân nhánh song song
(phân nhánh AND)



d) Chuyển tiếp lựa chọn
Kết hợp kiểu OR



e) Chuyển tiếp đồng bộ
(Kết hợp kiểu AND)