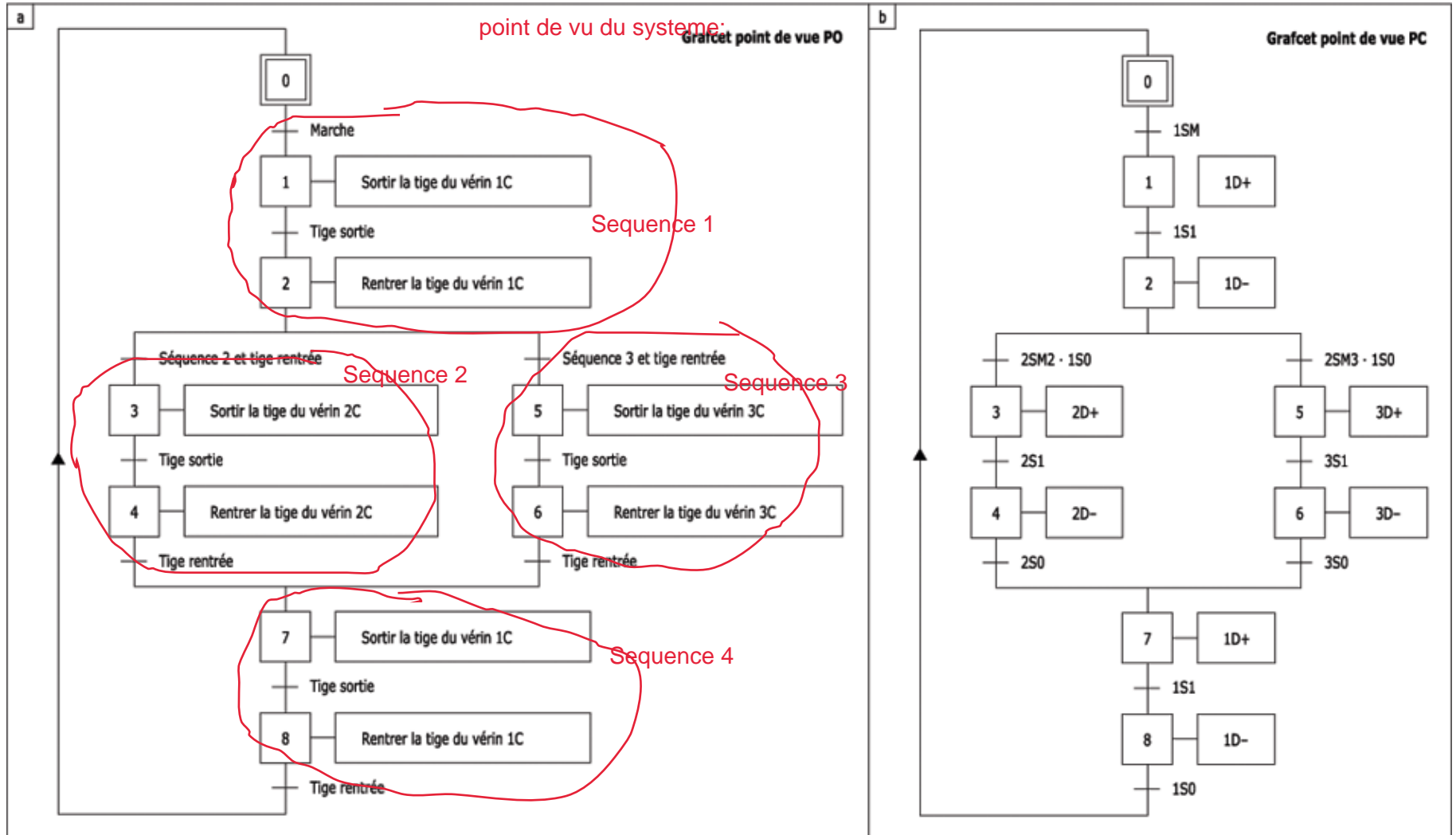


# API Automate Programmable Industriel Séquenceur

Auteur : Fabien Golay  
Version : 29.08.2016  
Sources :

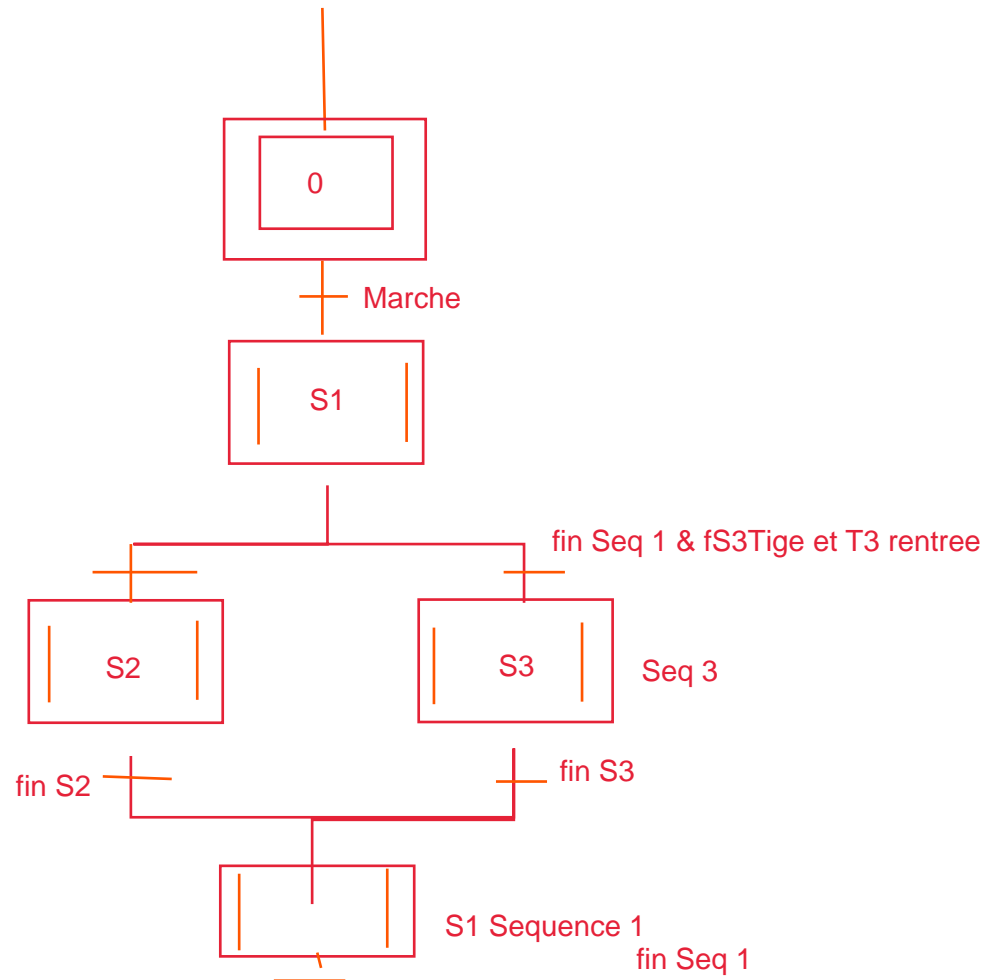
- Objectifs du séquenceur
  - Informer sur l'étape en cours
  - Autoriser le passage à l'étape suivante
  - Activer une sortie après temporisation
  - Détecter l'entrée dans l'étape
  - Mesurer le temps de l'étape

# • Principe de base du grafcet



- Principe de base du grafcet

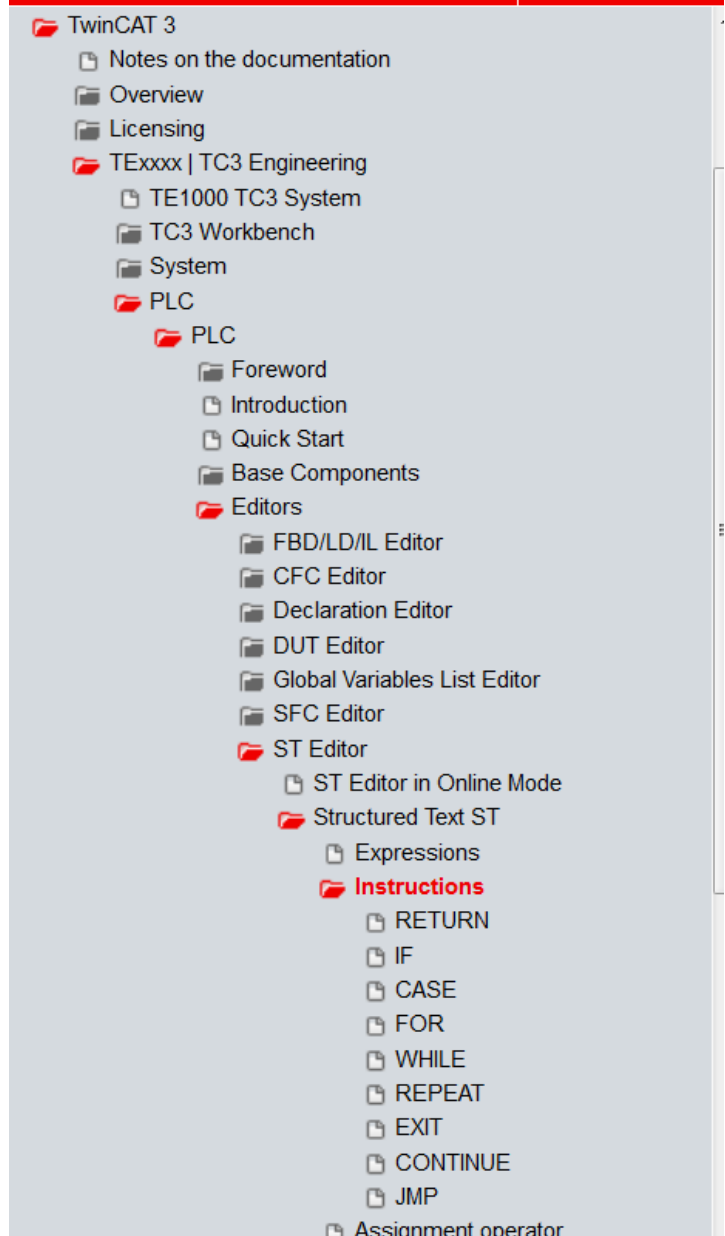
Réaliser le grafcet point de vue système (le niveau le plus haut)



- Programmation TS  
selon IEC-61131-3

Utilisation de l'instruction case avec une variable d'état ou d'étape. Dans certain cas, l'énumération permet de rendre la lecture plus facile.

L'aide des instructions dans TC3 se trouve sur <http://infosys.beckhoff.com> (navigation selon l'image).



# • Principe, grafcet vue système

Déclaration de la variable d'étape

```
iStep:int:=0;
```

Programmation de la séquence principale

```
case iStep of
```

```
0:// attend l'ordre de marche
```

```
100://Séquence 1, aller et retour de la tige 1C
```

```
150://Sélection de la séquence suivante, 2 ou 3
```

```
200://Séquence 2, aller et retour de la tige 2C
```

```
//Va à l'étape 400 à la fin de la séquence
```

```
300://Séquence 3, aller et retour de la tige 3C
```

```
//Va à l'étape 400 à la fin de la séquence
```

```
400://Séquence 1, aller et retour de la tige 1C
```

```
//Retour à l'étape 0
```

```
end_case
```

(\*Actions locales (La valeur de l'étape pouvant être exploité par d'autre P ou FB de niveau supérieur (Actions globales)\*)

```
Qx1DPlus:=iStep=? (selon valeur d'étape)
```

```
Qx1DMoins:=iStep=? (selon valeur d'étape)
```

- Option d'exécution pas à pas

Permet une exécution étape par étape.

Le flanc montant d'une variable autorise ce passage d'étape

Constatations :

- Pas nécessaire dans chaque étape

- Implique que l'étape d'avant configure l'étape d'après

- Il faut un lien vers une variable de libération de l'étape

- Il faut la valeur du mode en cours --> switch case?

Solutions :

1. Ajouter une variable d'entrée pour la configuration
2. **Utiliser une propriété (programmation orientée objet) pour intercepter l'écriture de la variable d'étape**

- Réaliser le Grafcet PC de l'exo 5.1 avec l'ajout du mode pas à pas



- Création d'un objet pour la gestion des étapes

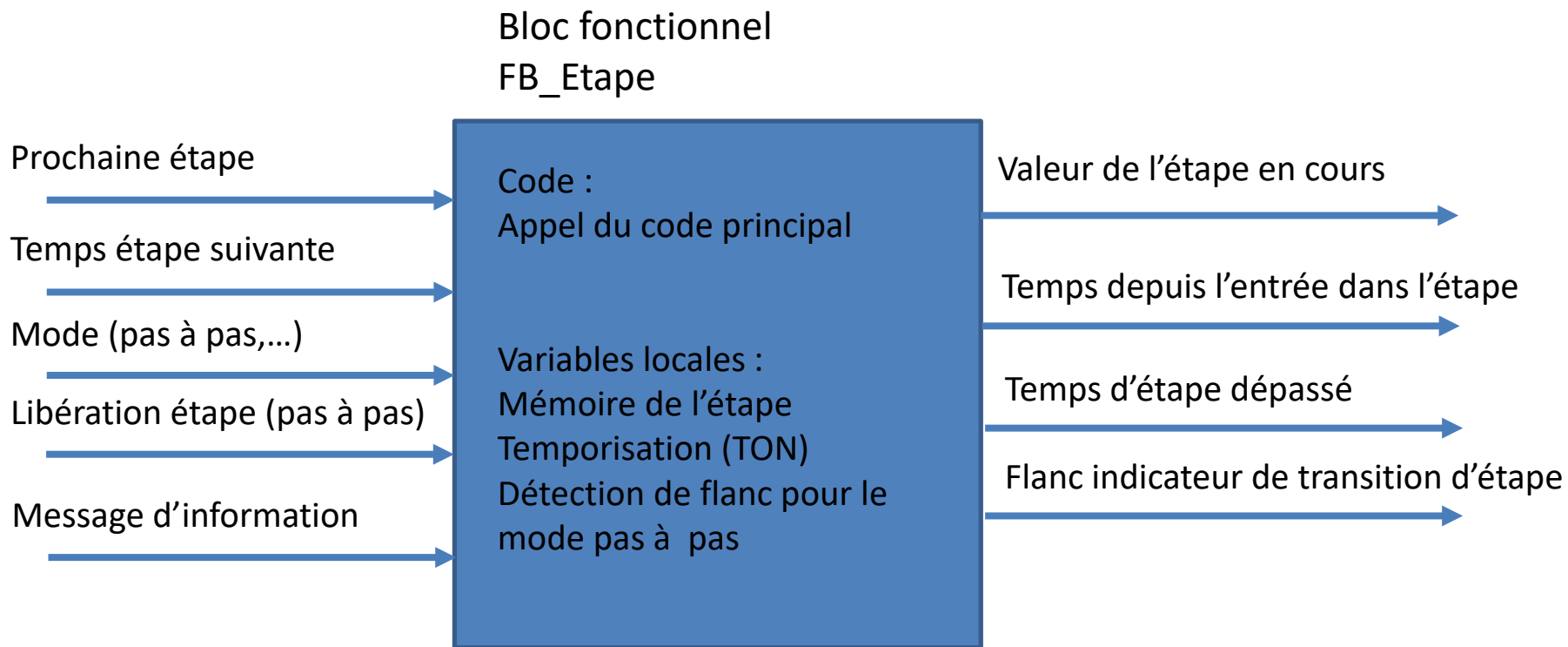
Brainstorming :

Lister les méthodes (fonctions communes)

Lister les propriétés (valeurs communes)

Autres

- Proposition



- Approche par évènement

Lors de l'écriture de la variable de l'étape, on souhaite faire le test du mode en cours. Pour cela, la propriété d'une programmation objet est adéquate (SET,GET).

2 propriétés :

pVal:int; pour la passage normal d'une étape

pSVal:int; S pour Stop, pour le passage avec gestion du mode pas à pas

1 mémoire :

iNo:int; Valeur de l'étape en cours, écriture depuis pVal et pSVal

Pour éviter l'écriture continue du message de l'étape, on va également utiliser une propriété qui réalise le transfert uniquement lors du changement d'étape.

1 propriété :

psMess :string; pour l'écriture du message de l'étape en cours

1 mémoire:

sMess:string; valeur actuelle du message

## FB\_Etape

### Propriétés :

piVal:int; //gestion d'étape sans fonction, uniquement SET  
piSVal:int; //gestion d'étape avec fonction de stop sur mode pas à pas uniquement SET  
psMess:string; //uniquement le Set de la variable sMess (économise la ressource)

### Variables entrées :

tTempsQ :TIME; //temps pour l'activation de la sortie xTempsQ

### Variables sorties :

tTempsEtape: TIME ; //temps écoulé dans l'étape  
xTempsQ : bool ; //est vrai si tTempsQ est atteint  
xFlancTransition:BOOL; //info de transition d'étape à l'entrée  
iStep:INT:=0; //information du no d'étape en cours  
sMess :STRING ; //message d'information de l'étape en cours

### Variables locales:

iMemStep:INT; //pour création des flancs  
fbTon:ton; //timer d'étape  
fbRtrig:R\_trig; //gestion du trig pour le mode pas à pas  
xMemPasAPas:bool; //mémoire du mode en cours (voir méthode mActualisation)

Public : mActualisation() Exécution cyclique, permet la mise à jour de l'objet

xPasAPas:BOOL:=FALSE; //activation du mode pas à pas  
xSetProchainPas:BOOL; //va au prochain pas sur flanc haut

Private : mTransition() Exécution lors d'une transition, initialise la variable sMess

- Utilisation

Déclaration

```
fbEtape:FB_Etape;
```

Appel cyclique

(\*actualisation des informations de l'objet de gestion des étapes

var 1 :gestion du blocage pour mode pas à pas

Var 2 :activation prochain pas\*)

```
fbEtape.mActualisation(xModePasAPas,xNextPas);
```

```
//gestion des étapes
```

```
CASE fbEtape.iNo OF
```

```
...
```

```
END_CASE;
```

Variables  
globales



- Utilisation

//gestion des étapes

CASE fbEtape.iNo OF

...

«no étape» :

//écriture du message

fbEtape.psMess:='Description de l'étape'

//Zone d'action à l'activation

if fbEtape.xFlancTransition then

    //action à l'activation

end\_if

//Test de la condition de transition

if xCondition then

    //Zone d'action en sortie d'étape

    //écriture de l'étape suivante avec la fonction pas à pas

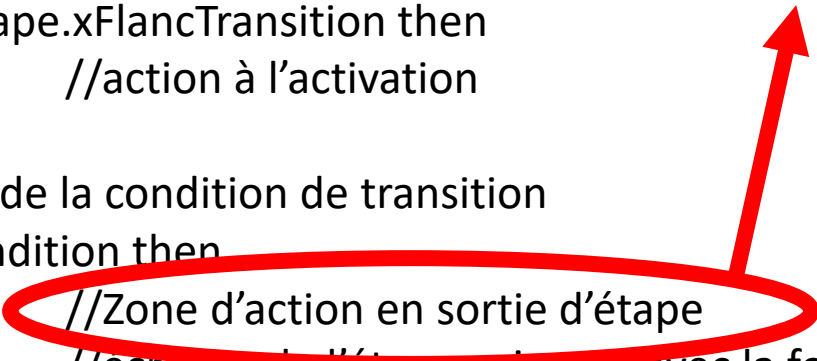
    fbEtape.piSVal:=«no étape suivante»;

end\_if

...

END\_CASE;

**N'est plus totalement  
vrai si pas à pas actif**



- Utilisation temporisation

//gestion des étapes

CASE fbEtape.iNo OF

...

«no étape précédente » :

fbEtape.psMess:='Configuration de la temporisation'

fbEtape.tTempoQ:=t#1s;

fbEtape.piVal:=«no étape»; //sans arrêt pas à pas

«no étape»:

fbEtape.psMess:='Attend la fin de la temporisation'

if fbEtape.xTempsQ then

    //passage d'étape avec arrêt pas à pas

    fbEtape.piSVal:=«no étape suivante»;

end\_if


...

END\_CASE;

- Utilisation temporisation

```
//gestion des étapes  
CASE fbEtape.iNo OF
```

**si pas à pas actif, le  
compteur tourne**



```
... no étape précédente » :
```

```
« fbEtape.psMess:='Configuration de la temporisation'
```

```
fbEtape.tTempoQ:=t#1s;
```

```
fbEtape.piVal:=«no étape»;
```

```
  «no étape»:
```

```
fbEtape.psMess:='Attend la fin de la temporisation'
```

```
if fbEtape.xTempsQ and lxInput1 then
```

```
  fbEtape.piSVal:=«no étape suivante»; elsif
```

```
  fbEtape.tTempsEtap > t#20s then
```

```
    //Temps trop long dans l'étape
```

```
    fbEtape.piSVal:=«no étape d'erreur»;
```

```
  end_if
```

```
...
```

```
END_CASE;
```