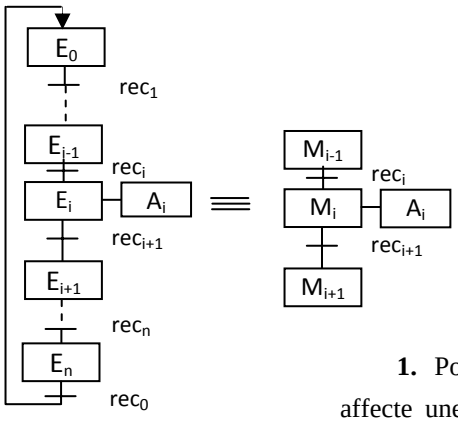
PROGRAMMATION DE GRAFCET DANS LES AUTOMATES MOELLER

En plus du langage SFC (qui est pris en charge chez Moeller de manière assez complexe et éloignée du concept même de grafcet, contrairement aux automates Schneider et Siemens), il existe deux autres techniques de programmation des grafcets dans l'automate Moeller. L'algorithme 1 utilise une technique de séquencement (activation et désactivation des étapes) très proche de celle utilisée dans le mode « list » ou langage booléen de Siemens. Elle associe à chaque étape un bit de mémoire. Dans l'algorithme 2, en plus du bit, la mémoire d'étape est réalisée par une bascule SR.

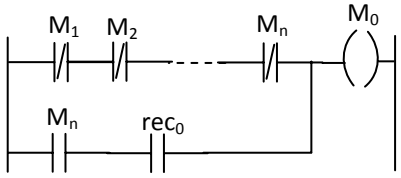
Considérons un grafcet bouclé avec n+1 étapes (numérotées de 0 à n). Si E_i a pour réceptivité d'action rec_i et pour réceptivité de désactivation rec_{i+1}, on peut alors expliquer les deux algorithmes comme suit.



ALGORITHME 1

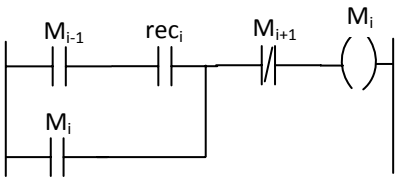
1. Pour chaque étape E_i (i variant de 0 à n), affecter une variable interne M_i (1 bit interne avec une adresse).

2. Initialisation ou activation de l'étape initiale



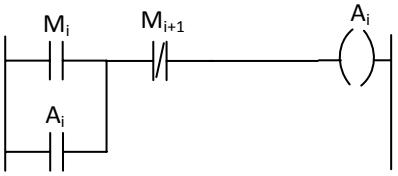
3. Activation des étapes

Pour chaque étape E_i (M_i), i variant de 1 à n, faire :



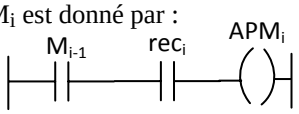
4. Actions

Pour chaque sortie Ai, i variant de 1 à n, faire :



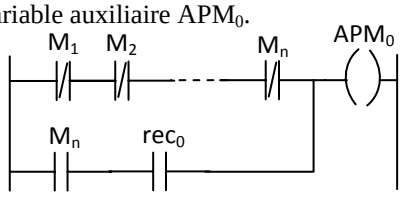
ALGORITHME 2

1. Pour chaque étape E_i (i variant de 0 à n), on affecte une variable interne M_i. De plus on crée pour chaque M_i une variable auxiliaire $APM_i = M_{i-1} \times rec_i$. Son programme correspondant ou programme auxiliaire d'activation de M_i est donné par :



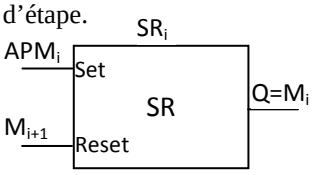
2. Initialisation ou activation de l'étape initiale

Contrairement à l'algorithme 1, on n'initialise pas M_0 mais sa variable auxiliaire APM_0.

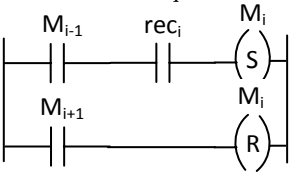


3. Activation des étapes

Pour chaque étape E_i (M_i), affecter une bascule SR_i mémoire d'étape.



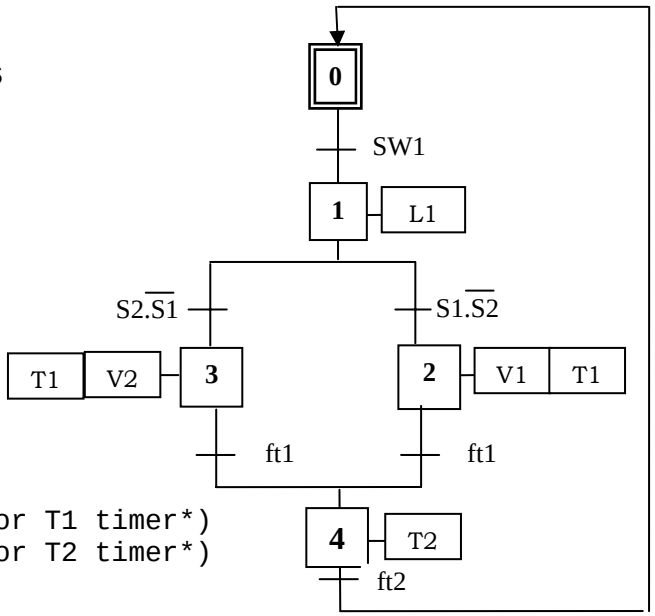
Une autre alternative à la bascule SR et sa variable d'activation APM_i, est l'utilisation de sorties (bobines) mémorisées pour les étapes. Ainsi le programme suivant est identique à celui de la bascule précédente.



4. Actions

Idem à l'algorithme 1.

```
PROGRAM ALG01
VAR
      (*programme didactique ALGO 1     17/4/12 copyright IVO & HH
      PROGRAMMATION DE GRAFCET EN LADDER PUR A L'AIDE DE BITS INTERNES
      1 ETAPE Ei = 1 BIT INTERNE Mi

      PARTIE I : DECLARATIONS DES VARIABLES
      1.Variables d'entrée*)
      SW1 AT %I0.0.0.0.0:    BOOL;
      S1  AT %I0.0.0.0.1:    BOOL;
      S2  AT %I0.0.0.0.2:    BOOL;

      (*2.Variables de sortie*)
      V1 AT %Q0.0.0.0.1:     BOOL;
      V2 AT %Q0.0.0.0.2:     BOOL;
      L1 AT %Q0.0.0.1.7:     BOOL;

      (*3.Périphériques internes*)
      T1:   TON;        (*Timer 1*)
      T2:   TON;        (*Timer 2*)
      DT1:  TIME := T#5S;    (*Delay time for T1 timer*)
      DT2:  TIME := T#5s;    (*Delay time for T2 timer*)

      (*4.Variables internes*)
      QT2 AT %M0.0.0.2.2:    BOOL; (*Output T2*)
      QT1 AT %M0.0.0.2.1:    BOOL; (*Output T1*)
      M23 AT %M0.0.0.2.3:    BOOL; (*Auxiliary variable for T1*)

      M0 AT %M0.0.0.0.0:     BOOL; (*Memory Bit for the Step 0*)
      M1 AT %M0.0.0.0.1:     BOOL; (*Step 1*)
      M2 AT %M0.0.0.0.2:     BOOL; (*Step 2*)
      M3 AT %M0.0.0.0.3:     BOOL; (*Step 3*)
      M4 AT %M0.0.0.0.4:     BOOL; (*Step 4*)
END_VAR
```

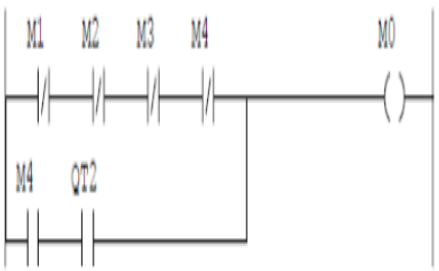


(*PARTIE II : INSTRUCTIONS

Le programme a été saisi en LADDER. La traduction en mode LIST a été effectuée par le système de l'environnement de programmation de l'API MOELLER

Etape 0: étape initiale pour la séquence*)

```
LDN   M1
ANDN  M2
ANDN  M3
ANDN  M4
OR    (     M4
      AND   QT2
      )
ST    M0
```



(*Etape 1*)

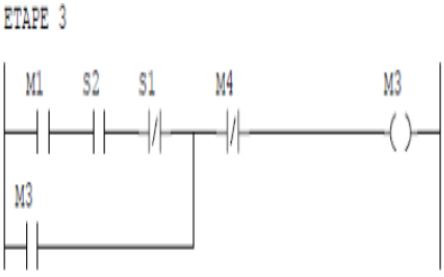
```
LD    M0
AND   SW1
OR    M1
ANDN  M2
ANDN  M3
ST    M1
```

(*Etape 2*)

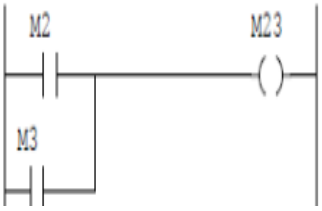
```
LD    M1
AND   S1
ANDN  S2
OR    M2
ANDN  M4
ST    M2
```

(*Etape 3*)

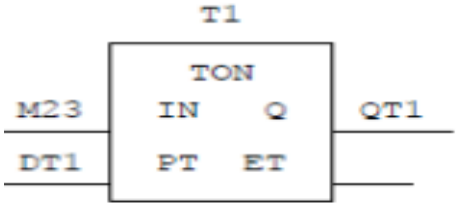
```
LD    M1
AND   S2
ANDN  S1
OR    (M3
      )
ANDN  M4
ST    M3
```



```
(*Variable auxiliaire de commande du timer T1 (activité étape 2 ou 3) *)
LD    M2
OR    M3
ST    M23
```

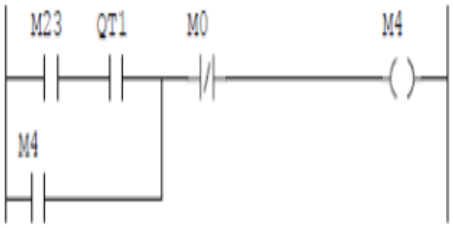



```
(*Temporisation T1*)
CAL   T1(
      IN :=       M23,
      PT :=       DT1
      |
      QT1 := Q,
       := ET
      )
```

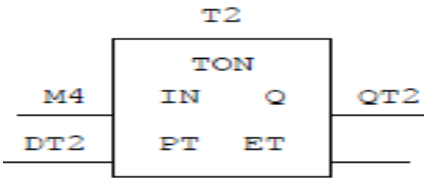



```
(*Etape 4*)
LD    M23
AND   QT1
OR    M4
ANDN  M0
ST    M4
```

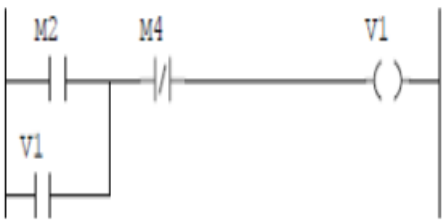
ETAPE 4



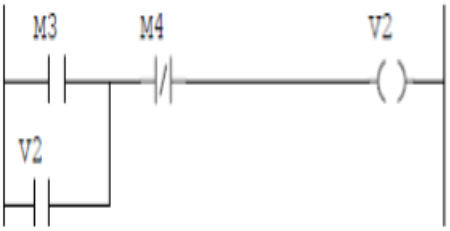

```
(*Temporisation T2*)
CAL   T2(
      IN :=       M4,
      PT :=       DT2
      |
      QT2 := Q,
       := ET
      )
```




```
(*Action vérin V1*)
LD    M2
OR    V1
ANDN  M4
ST    V1
```



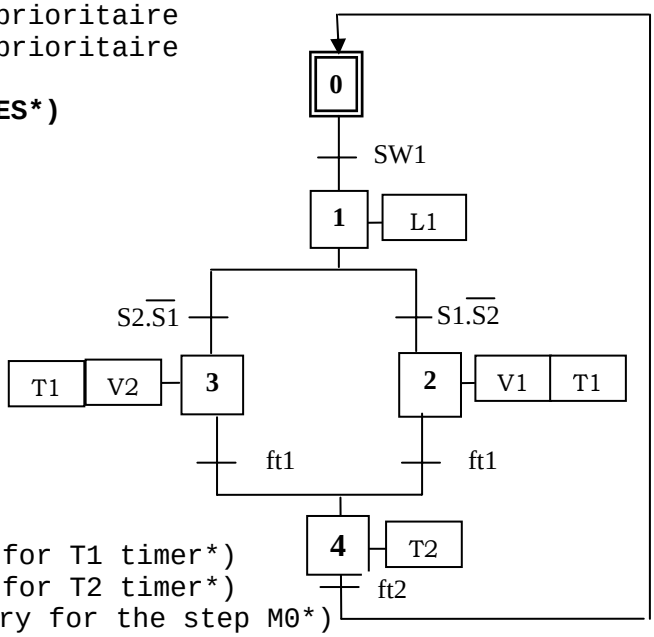

```
(*Action vérin V2*)
LD    M3
OR    (     V2
      )
ANDN  M4
ST    V2
```




```
(*Action lampe L1*)
LD    M1
OR    (     L1
      )
ANDN  M23
ST    L1
```

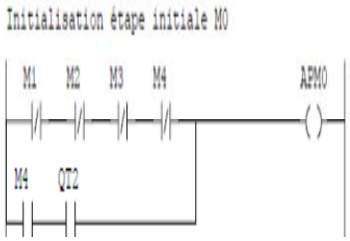

```
END_PROGRAM
```

```
VAR
      (*programme didactique ALGO 2     17/4/12 copyright IVO & HH
      PROGRAMMATION DE GRAFCET A L'AIDE DE BASCULES SR
1 ETAPE Ei = 1BIT INTERNE Mi +1BASCULE SRi MEMOIRE D'ETAPE + 1variable auxiliaire APMi
      SR flipflop: bascule avec mise à 1 prioritaire
      RS flipflop: bascule avec mise à 0 prioritaire

      PARTIE I : DECLARATIONS DES VARIABLES*)
      (*1.Variables d'entrée*)
      SW1 AT %I0.0.0.0.0:    BOOL;
      S1  AT %I0.0.0.0.1:    BOOL;
      S2  AT %I0.0.0.0.2:    BOOL;

      (*2.Variables de sortie*)
      V1 AT %Q0.0.0.0.1:     BOOL;
      V2 AT %Q0.0.0.0.2:     BOOL;
      L1 AT %Q0.0.0.1.7:     BOOL;

      (*3.Périphériques internes*)
      T1:   TON;        (*Timer 1*)
      T2:   TON;
      DT1:  TIME := T#5S;    (*Delay time for T1 timer*)
      DT2:  TIME := T#5s;    (*Delay time for T2 timer*)
      SR0:  SR;   (*SR flip flop step memory for the step M0*)
      SR1:  SR;
      SR2:  SR;
      SR3:  SR;
      SR4:  SR;

      (*4.Variables internes*)
      QT2 AT %M0.0.0.2.2:    BOOL; (*Output T2*)
      QT1 AT %M0.0.0.2.1:    BOOL; (*Output T1*)

      M0  AT %M0.0.0.0.0:    BOOL; (*Memory Bit for the Step 0*)
      M1  AT %M0.0.0.0.1:    BOOL; (*Step 1*)
      M2  AT %M0.0.0.0.2:    BOOL; (*Step 2*)
      M3  AT %M0.0.0.0.3:    BOOL; (*Step 3*)
      M4  AT %M0.0.0.0.4:    BOOL; (*Step 4*)
      M23 AT %M0.0.0.0.5:    BOOL; (*Auxiliary variable for T1*)

      APM0 AT %M0.0.0.0.6:   BOOL; (*Auxiliary program to set the step M0*)
      APM1 AT %M0.0.0.0.7:   BOOL;
      APM2 AT %M0.0.0.1.0:   BOOL;
      APM3 AT %M0.0.0.1.1:   BOOL;
      APM4 AT %M0.0.0.1.2:   BOOL;

END_VAR

      (*PARTIE II : INSTRUCTIONS
      Le programme a été saisi en LADDER. La traduction en mode LIST a été
      effectuée par le système de l'environnement de programmation de l'API MOELLER

      Pour la partie « actions » du programme, elle ne sera pas donnée puisqu'elle
      est identique à celle de l'algorithme 1.
      *)
```



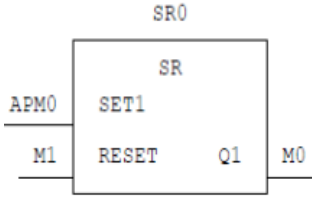
(*Initialisation étape M0*)

```
LDN   M1
ANDN  M2
ANDN  M3
ANDN  M4
OR    ( M4
      AND QT2
      )
ST    APM0
```



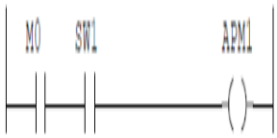
(*Bascule SR0 mémoire d'étape de M0*)

```
CAL   SR0(
      SET1 := APM0,
      RESET:= M1
      |
      M0 := Q1
      )
```



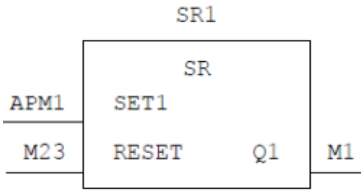
(*Programme Auxiliaire d'activation de M1*)

```
LD    M0
AND   SW1
ST    APM1
```



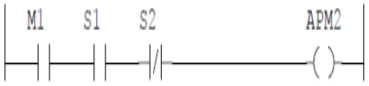
(*Bascule SR1 mémoire d'étape de M1*)

```
CAL   SR1(
      SET1 := APM1,
      RESET:= M23
      |
      M1 := Q1
      )
```



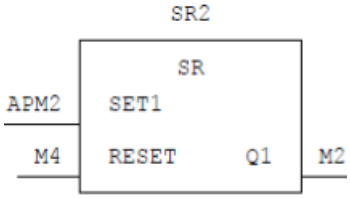
(*Programme Auxiliaire d'activation de M2*)

```
LD    M1
AND   S1
ANDN  S2
ST    APM2
```



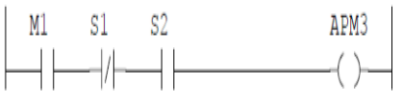
(*Bascule SR2 mémoire d'étape de M2*)

```
CAL   SR2(
      SET1 := APM2,
      RESET:= M4
      |
      M2 := Q1
      )
```



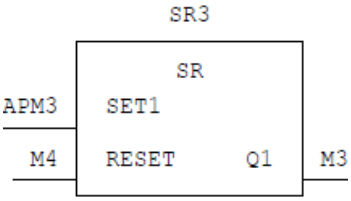
(*Programme Auxiliaire d'activation de M3*)

```
LD    M1
ANDN  S1
AND   S2
ST    APM3
```



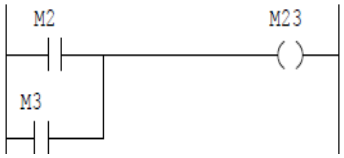
(*Bascule SR3 mémoire d'étape de M3*)

```
CAL   SR3(
      SET1 := APM3,
      RESET:= M4
      |
      M3 := Q1
      )
```



(* variable auxiliaire = activité étape2 ou étape3 *)

```
LD    M2
OR    M3
ST    M23
```

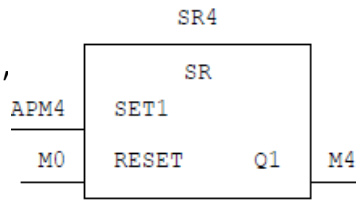


(*Programme Auxiliaire d'activation de M4*)

```
LD    M23
AND   QT1
ST    APM4
```

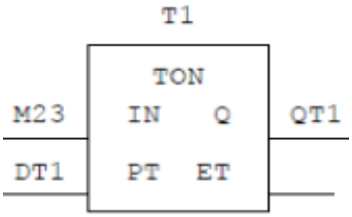
(*Bascule SR4 mémoire d'étape de M4*)

```
CAL   SR4(
      SET1 := APM4,
      RESET:= M0
      |
      M4 := Q1
      )
```



(*Temporisation T1, activée par M2 ou M3, de durée DT1, de signal de fin de temporisation QT1*)

```
CAL   T1(
      IN := M23,
      PT := DT1
      |
      QT1 := Q,
       := ET
      )
```



(*Temporisation T2 Activée par M4, de durée DT2, de signal de fin de temporisation QT2*)

```
CAL   T2(
      IN := M4,
      PT := DT2
      |
      QT2 := Q,
       := ET
      )
```