UNIVERSITÉ DE STRASBOURG

mise en oeuvre CFC sur S7

Le langage CFC permet de gérer à la fois du tout ou rien, en séquentiel et combinatoire, du numérique, et de l'asservissement en boucle ouverte ou fermée.

Méthodologie

On commence par créer un projet, avec la description du matériel. Dans « Programmes S7 » il y a « Sources », « Blocs », « Mnémoniques ». S'il n'y est pas, y ajouter (clic droit) un « dossier Diagrammes ». Dans ce dossier on insère maintenant un (ou plusieurs) CFC. On l'ouvre. On peut voir une seule « feuille », ou six (icône ). On appelle ces 6 feuilles une « partition », rien n'empêche de créer des partitions si 6 feuilles ne suffisent pas.

Une « feuille » (je n'aime pas le nom, j'aurais préféré réseau, comme en langage CONT ou LOG) est un ensemble de composants, reliés entre eux. On trouve tous les composants existants en ouvrant, à gauche, le catalogue (). On insère les objets dans la feuille (glissé de souris depuis le catalogue), et on crée les liaisons (en cliquant d'abord sur le départ puis sur l'arrivée). En cas de liaison ToR (BOOL), on peut directement insérer un inverseur sur les entrées des composant (). Si une liaison doit être raccordée à l'extérieur de la feuille, il faut ouvrir en haut la liste des « connecteurs » (), puis on « glisse » la sortie dans IN, OUT ou IN/OUT. On peut lui donner, dans cette table, un autre nom que celui qu'il a dans le composant On peut aussi relier une E/S d'un composant ou un connecteur à une E/S physique de l'automate : clic droit, connexion à l'opérande (il propose tous les mnémoniques compatibles si on les a déjà définis, mais on peut directement mettre E0.0). Dans la pratique, je préfère faire une feuille non liée à des adresses absolues (mais avec des connecteurs) puis une seconde feuille, incluant la précédente (voir plus bas) qui elle relie ces connecteurs aux adresses absolues.

Toutes les E/S des composants ne sont pas nécessairement affichées : clic droit sur le composant -> propriétés, onglet « connecteurs » et valider (ou non) la colonne « invisibilité ». En particulier les composants standards complexes ont un grand nombre d'E/S, dont seule une partie sert dans les cas habituels. Certains composants (comme le AND) acceptent d'ajouter des entrées (clic droit -> nombre d'entrées). Une autre option possible pour ces E/S est « contrôle commande » pour qu'elle soit visible sous WinCC (après une compilation de l'OS). Si la case à cocher n'est pas proposée : dans le tableau IN ou OUT : afficher -> afficher les colonnes.

Dans une feuille, on peut utiliser un composant défini dans une autre feuille (on aura alors bien spécifié les connecteurs IN et OUT), on les trouve dans la liste des diagrammes (onglet à sélectionner en bas du catalogue). On va donc pouvoir organiser son programme de manière arborescente (SADT, décomposition en sous-programmes...). Mais attention,

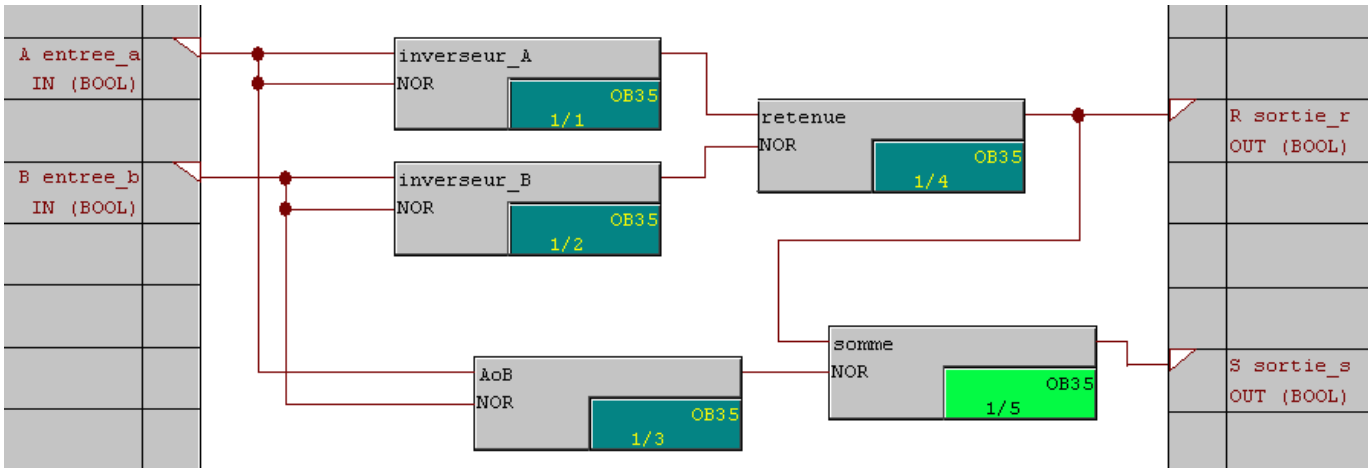
au moment où l'on insère un sous-composant, on en fait une copie dans la feuille actuelle. Ce qui veut dire qu'en modifiant un sous-composant qui a déjà été incorporé dans une ou plusieurs feuilles, toutes ces feuilles garderont l'ancien sous-composant (il y a peut-être moyen de faire autrement, en compilant une bibliothèque par exemple, mais je ne sais pas (encore) comment faire).

Ensuite, il faut le compiler (), le transférer dans l'automate () (ou PLCSIM). Si tout va bien, on peut vérifier ce qu'il se passe en ligne (mode test ), voire assigner des valeurs à des entrées de composants (en particulier celles où l'on n'avait pas mis de liaison mais laissé une valeur par défaut).

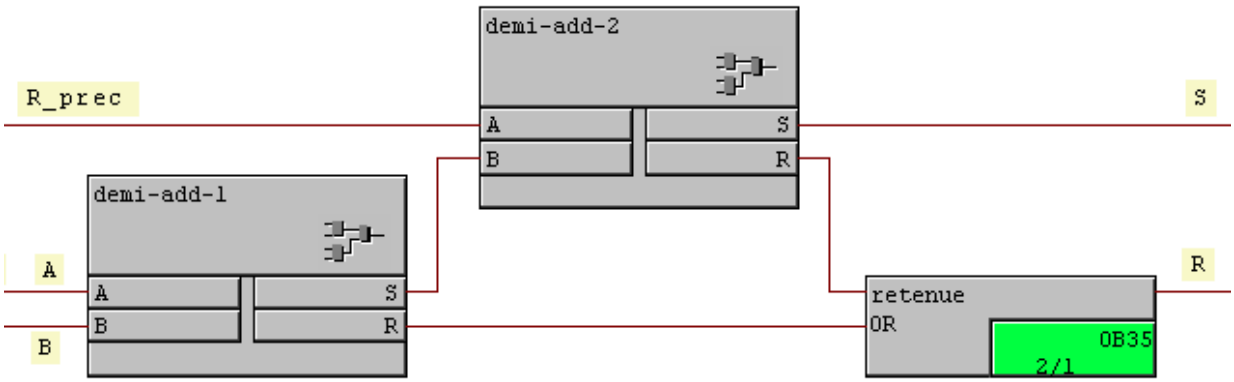
Les diagrammes sont exécutés dans un certain ordre. Cet ordre dépend de l'ordre dans lequel on a créé les composants. Mais on peut le changer soit manuellement ( puis glisser les lignes vers le haut ou le bas dans la même feuille, voir créer de nouveaux groupes d'exécution et y déplacer certains composants) soit automatiquement (outils-> optimiser l'ordre d'exécution). Par défaut, il groupe ensemble tous les composants d'une feuille (1er chiffre), puis l'ordre à l'intérieur du groupe (2éme chiffre). Une sortie (de type BOOL) d'un composant peut valider l'exécution d'un groupe ou son figeage.

Exemple en tout-ou rien, combinatoire : l'additionneur

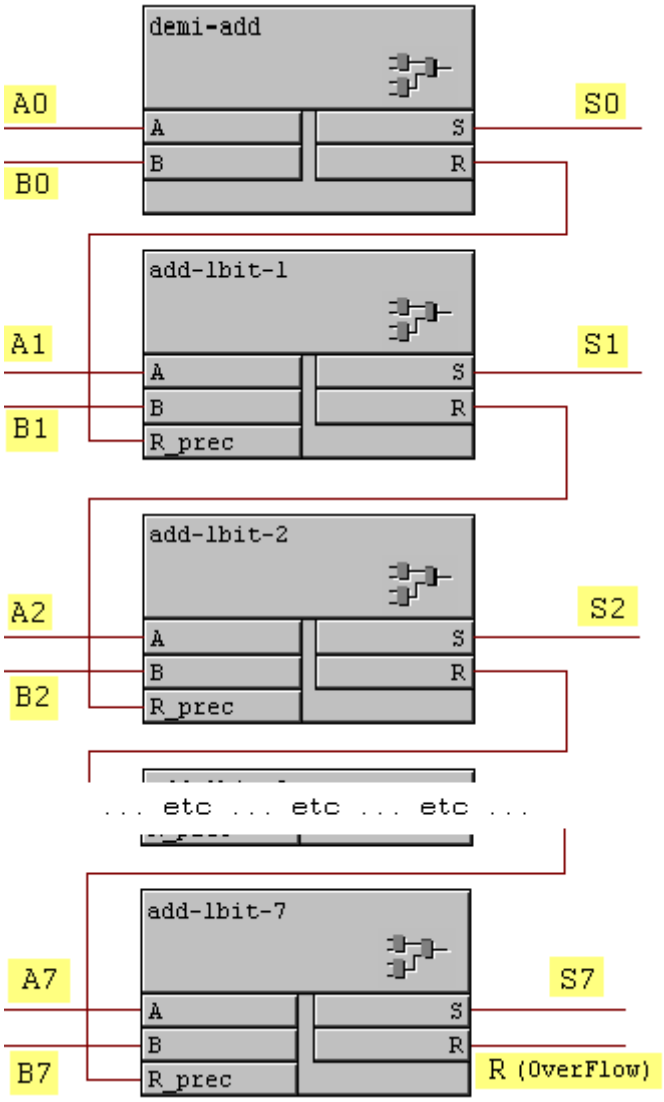
Ce problème est inspiré du transparent T3 du cours « Composants Numériques et Automatisme » dispensé en Licence Physique et Application mention Ingénierie (L2 S4), et disponible sur internet. Soit le demi additionneur : deux entrées ToR A et B, deux sorties : S (somme) et R (retenue). 0+0=0, 0+1 et 1+0=1, 1+1=0 mais retenue 1. Donc S=A xor B, R=A.B. Dans un circuit intégré, on utilise des composants identiques, donc en cours je le présente réalisé à l'aide de 5 portes NOR. :



On peut le réutiliser pour créer un additionneur 1bit plus une retenue éventuelle :

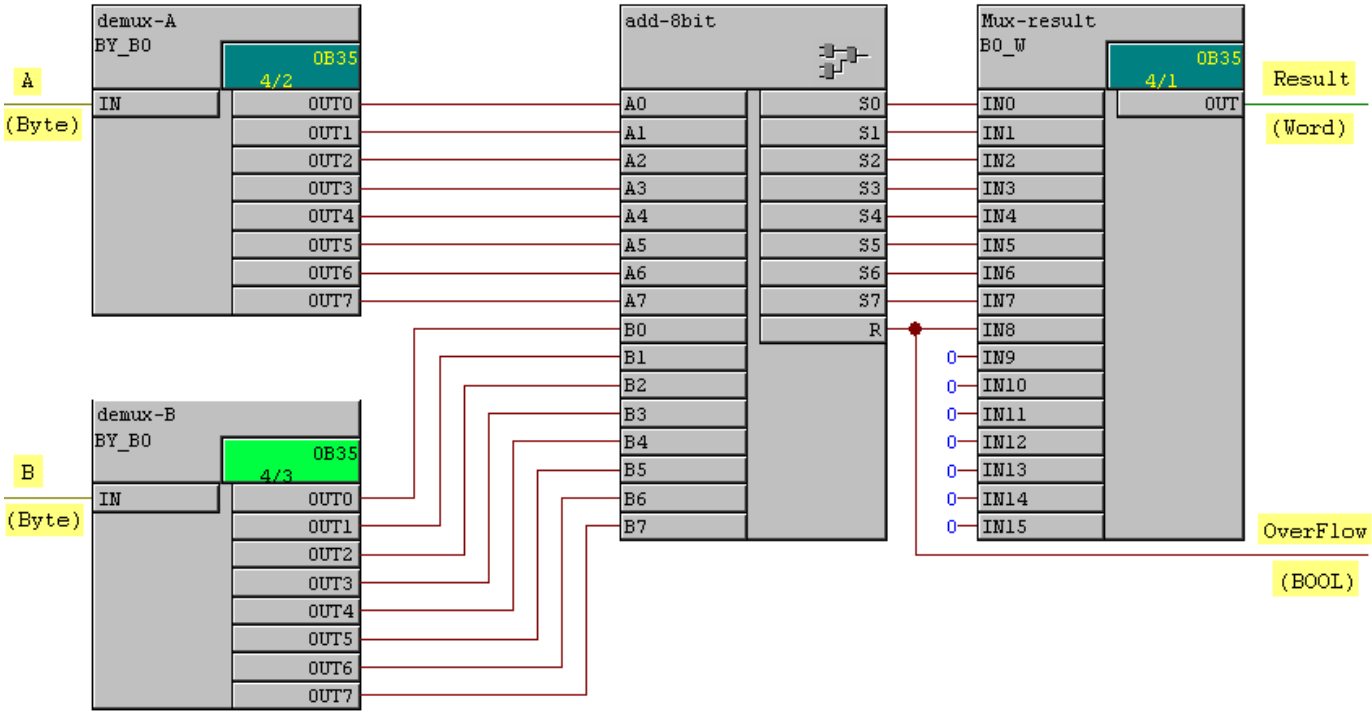


Avec un demi-additionneur (il n'y a pas de retenue à ajouter au bit de poids faible) et sept de ces additionneurs 1bit+retenue, on crée un additionneur 8bits (plus éventuelle retenue ou dépassement de capacité) :

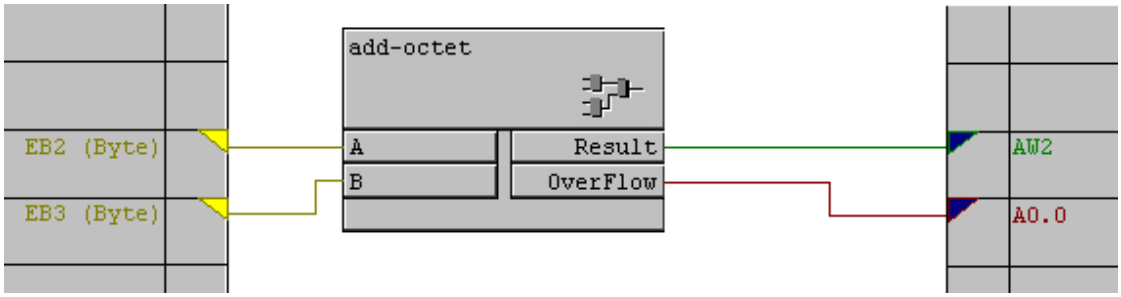


Ceci nous donne un composant à 16 entrées et 9 sorties, peut-être un peu lourd à gérer. Créons donc (encore) un nouveau composant, utilisant le précédent, permettant d'entrer

deux octets, résultat sur un octet, plus un bit de retenue (ou, être plus général, mettons la sortie sur 16 bits avec la retenue et 7 zéros de poids fort, rien n'oblige de les utiliser) :

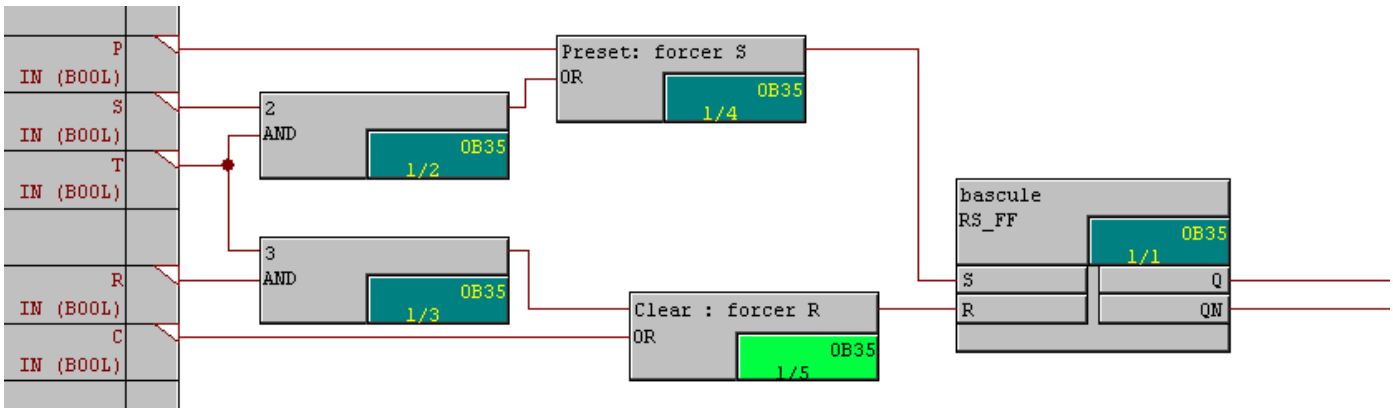


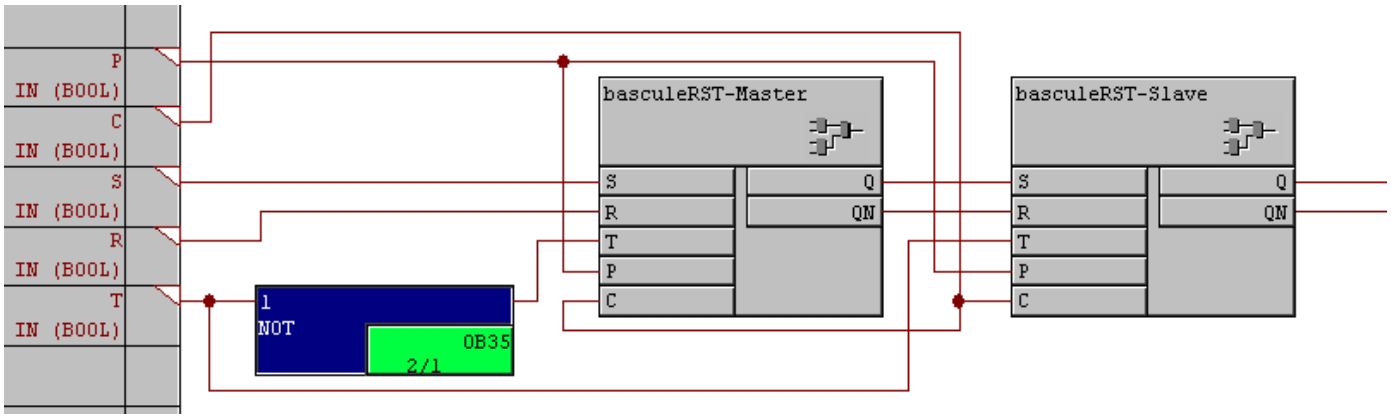
Ouf. Dernière feuille : pour tester, relions l'additionneur aux E/S de l'automate. Dans la configuration de nos consoles d'E/S, on additionne les deux chiffres de poids faible aux deux chiffres de poids fort des roues codeuses, on met le résultat sur l'afficheur et on allume A0.0 si OverFlow :

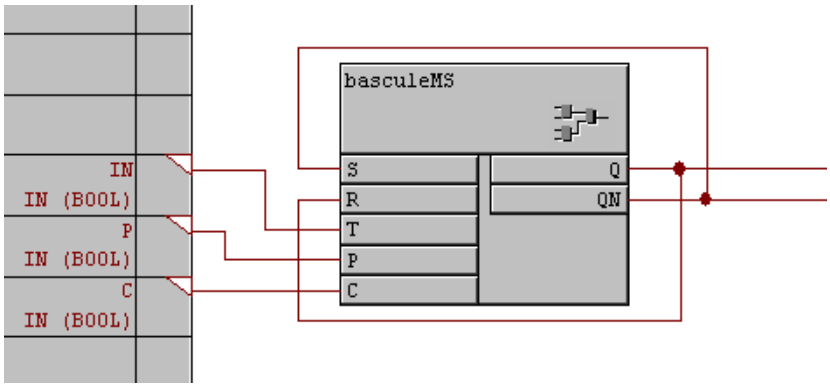


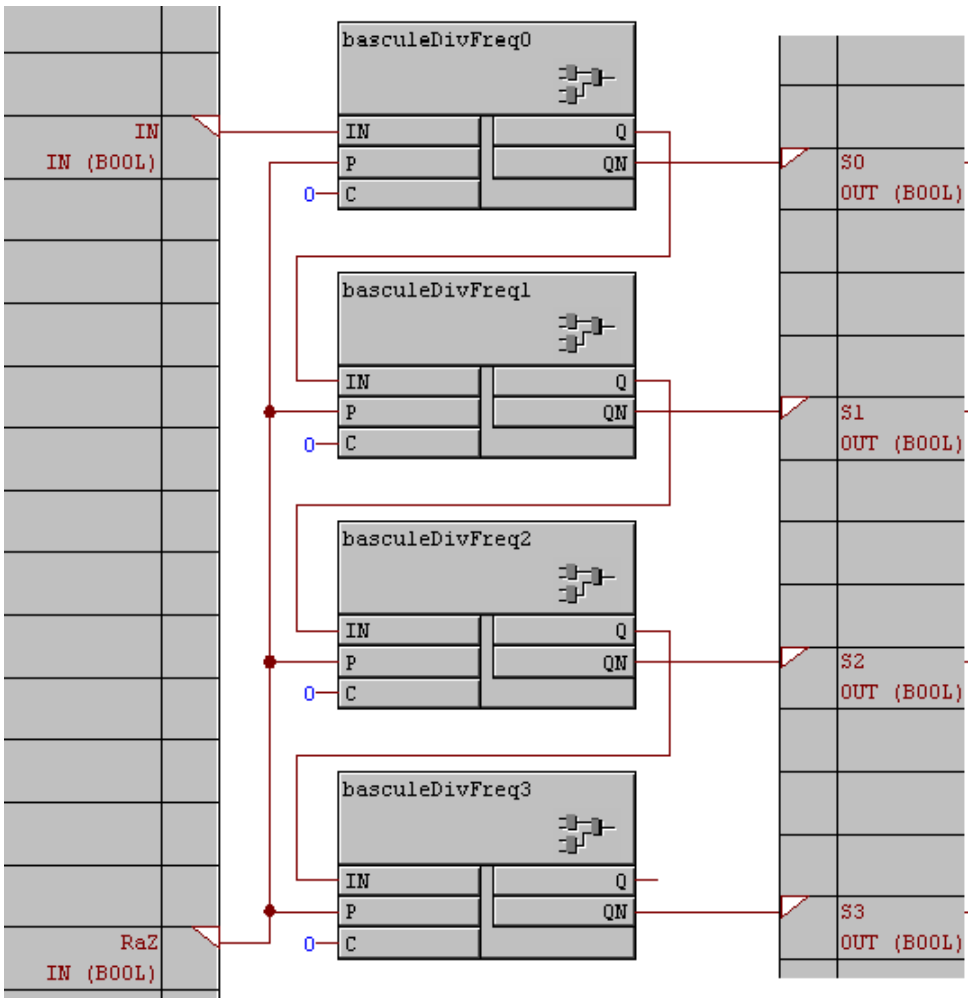
Composants séquentiels : le compteur à base de bascules

Il n'y a pas de problème particulier, on utilise simplement, en plus, des bascules (disponibles sous « flipflop » dans le catalogue). Je crée, dans l'ordre : la bascule RST utilisant la RS (existante), puis la MS (maître esclave), le diviseur de fréquence, et enfin le compteur (4bits).


P
IN (BOOL)
S
IN (BOOL)
T
IN (BOOL)
R
IN (BOOL)
C
IN (BOOL)
2
AND
OB35
1/2
3
AND
OB35
1/3
Preset: forcer S
OR
OB35
1/4
Clear : forcer R
OR
OB35
1/5
bascule
RS_FF
OB35
1/1
S
R
Q
QN

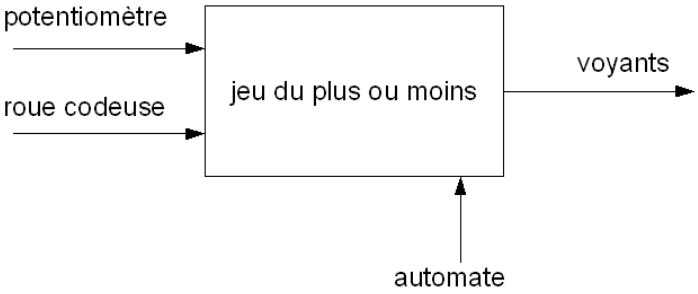

P
IN (BOOL)
C
IN (BOOL)
S
IN (BOOL)
R
IN (BOOL)
T
IN (BOOL)
1
NOT
OB35
2/1
basculeRST-Master
S
R
T
P
C
Q
QN
basculeRST-Slave
S
R
T
P
C
Q
QN

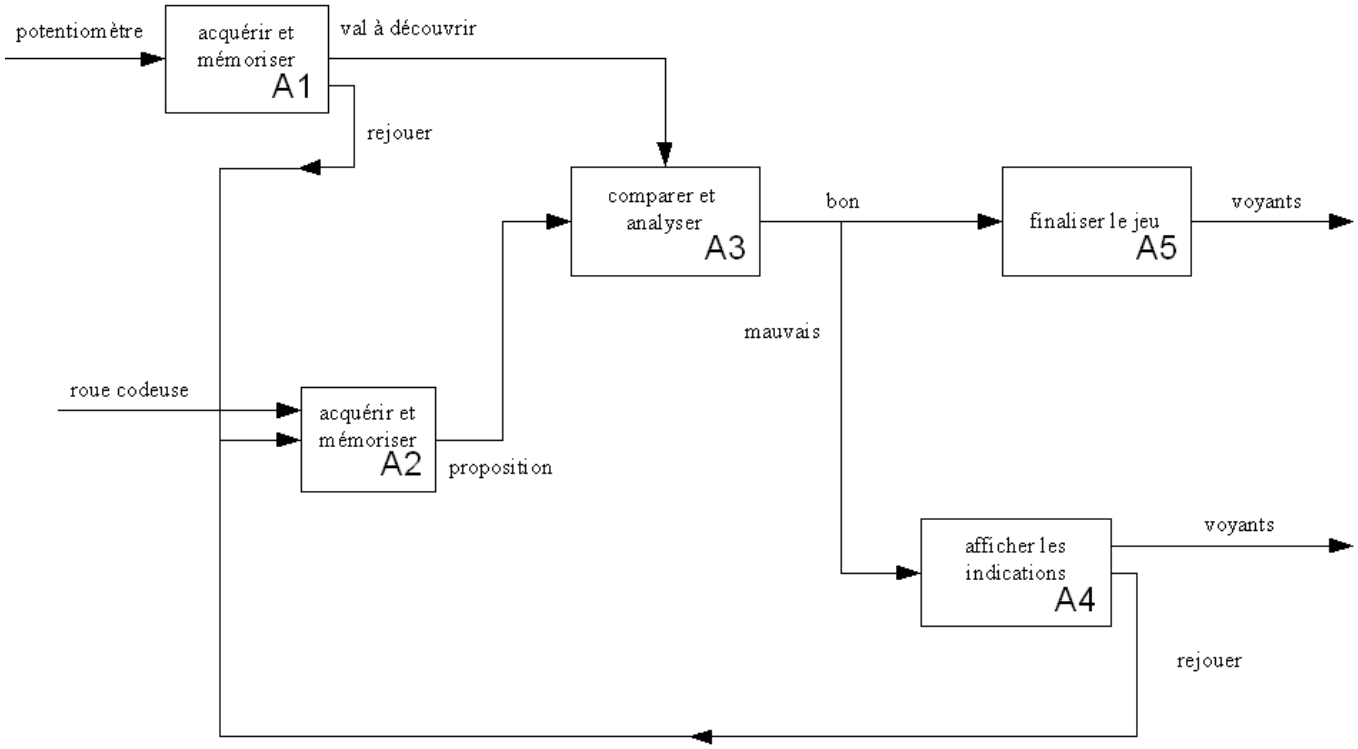

IN
IN (BOOL)
P
IN (BOOL)
C
IN (BOOL)
basculeMS
S
R
T
P
C
Q
QN



exemple numérique : jeu du plus ou moins

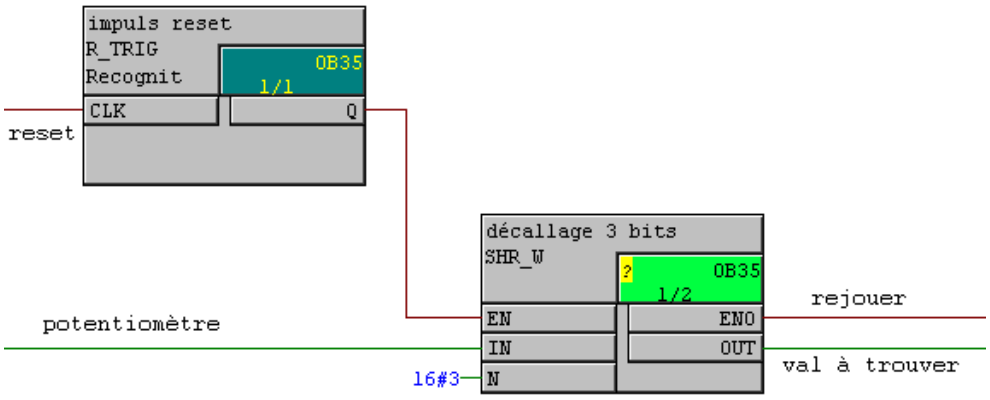
Chaque fois que je me mets à un nouveau langage, je programme ce jeu. Ici, nous allons récupérer la valeur fournie par le CAN (sur notre configuration, EW6, relié au potentiomètre de gauche). La valeur sera mémorisée (MW10) jusqu'au prochain front montant de l'entrée « reset » (E0.1) Sur le codeur 4 chiffres (EW2), nous allons proposer (en décimal) une valeur (validation par front de l'entrée E0.0), l'automate nous dira si c'est plus (A0.0 allumé) ou moins (A0.1), jusqu'à ce qu'on trouve la bonne valeur (A0.4). On peut améliorer le jeu en précisant si on chauffe (à +/-100 de la solution), si ça brûle (+/-10). Je propose d'étudier le problème sous par analyse descendante.



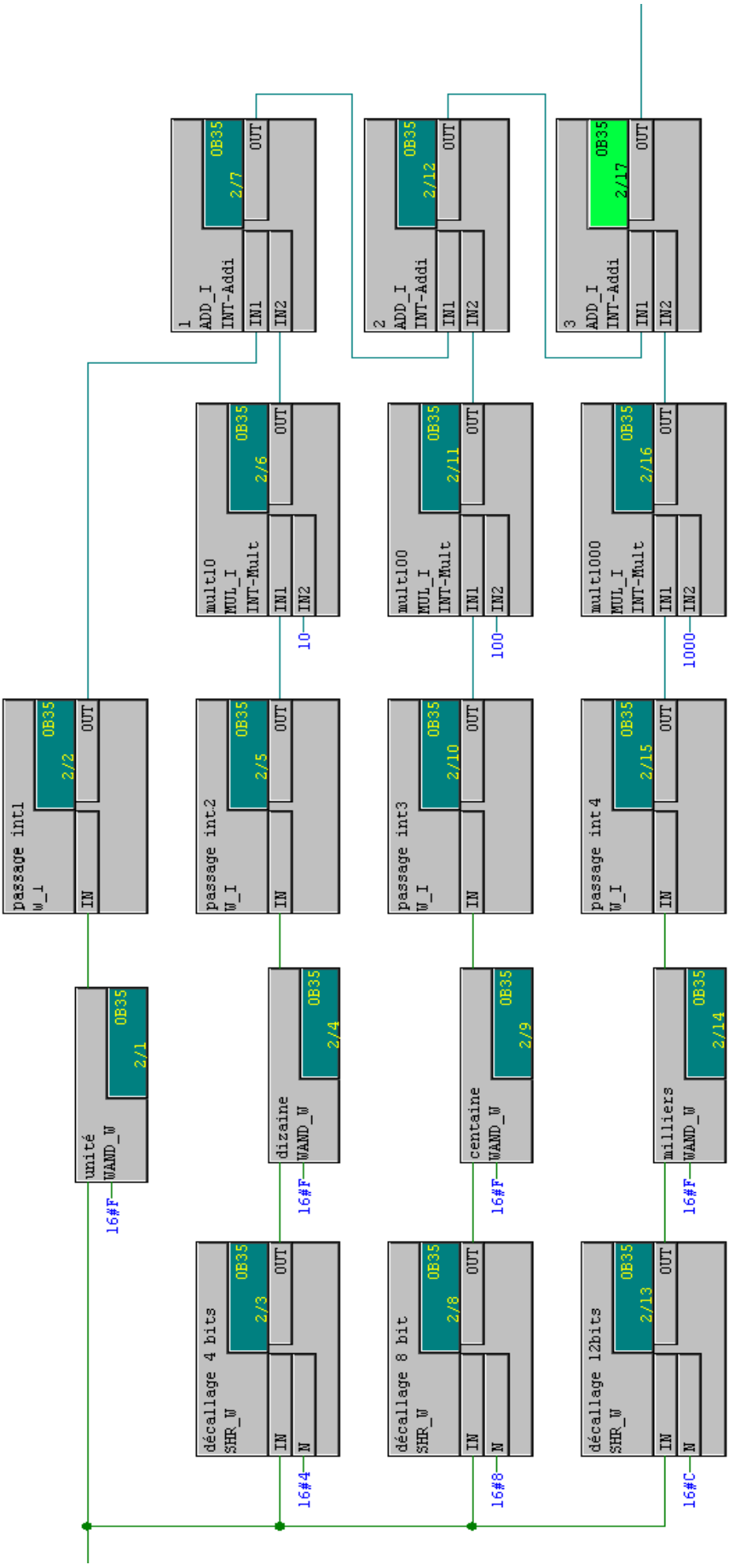


On décompose évidement le problème de la même manière. Par contre, je n'ai pas bien pu prendre en compte l'enchaînement des tâches (une tâche se déclenche dès que les précédentes ont fourni une nouvelle valeur), le front montant sur reset et sur validation va donc gérer cela. On peut néanmoins valider ou non un « groupe d'exécution », ça devrait permettre de résoudre le problème, mais je n'ai pas essayé pour l'instant.

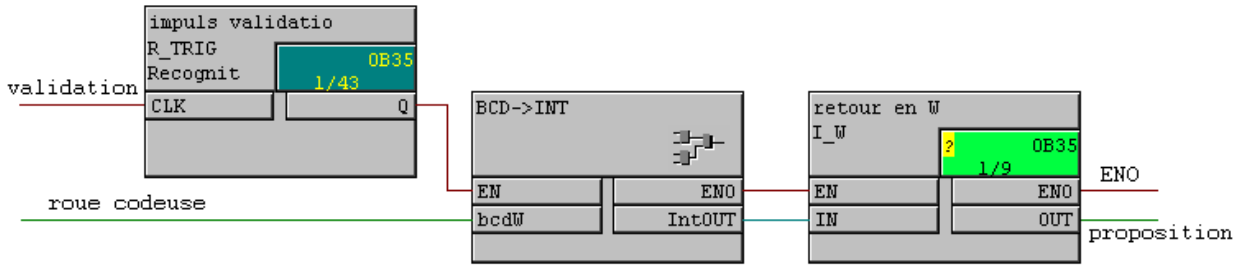
A1 doit acquérir la donnée, et la décaler de trois bits à droite (dû au câblage de notre automate). Ceci est fait uniquement au front montant du reset (même si on modifie le potentiomètre en cours du jeu).



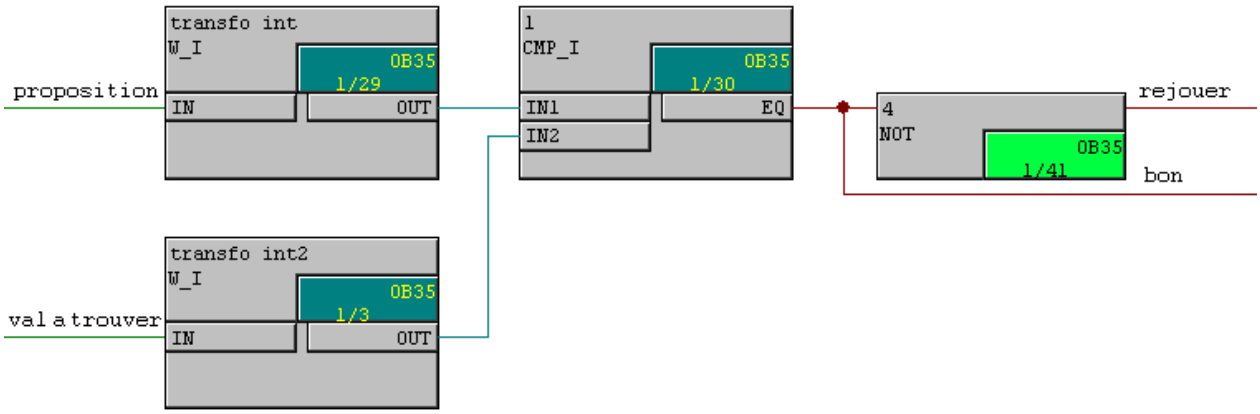
Pour A2, commençons par traduire un nombre BCD en binaire. La fonctionnalité existe peut-être dans la bibliothèque, mais je ne l'ai pas trouvée (et la mienne dépasse 999). J'isole les chiffres (décalage puis masquage) et les multiplie par 10, 100, 1000 pour ajouter le tout. Dans ma version, j'y ai également traité les EN/ENO, mais je ne le représente pas ici par manque de place.


1
ADD_I
INT-Addi
OB35
2/7
OUT
IN1
IN2
2
ADD_I
INT-Addi
OB35
2/12
OUT
IN1
IN2
3
ADD_I
INT-Addi
OB35
2/17
OUT
IN1
IN2
mult10
MUL_I
INT-Mult
OB35
2/6
OUT
IN1
IN2
10
mult100
MUL_I
INT-Mult
OB35
2/11
OUT
IN1
IN2
100
mult1000
MUL_I
INT-Mult
OB35
2/16
OUT
IN1
IN2
1000
passage int1
W_I
OB35
2/2
OUT
IN
passage int2
W_I
OB35
2/5
OUT
IN
passage int3
W_I
OB35
2/10
OUT
IN
passage int4
W_I
OB35
2/15
OUT
IN
unité
WAND_W
OB35
2/1
16#F
dizaine
WAND_W
OB35
2/4
16#F
centaine
WAND_W
OB35
2/9
16#F
milliers
WAND_W
OB35
2/14
16#F
décallage 4 bits
SHR_W
OB35
2/3
OUT
IN
N
16#4
décallage 8 bit
SHR_W
OB35
2/8
OUT
IN
N
16#8
décallage 12bits
SHR_W
OB35
2/13
OUT
IN
N
16#C

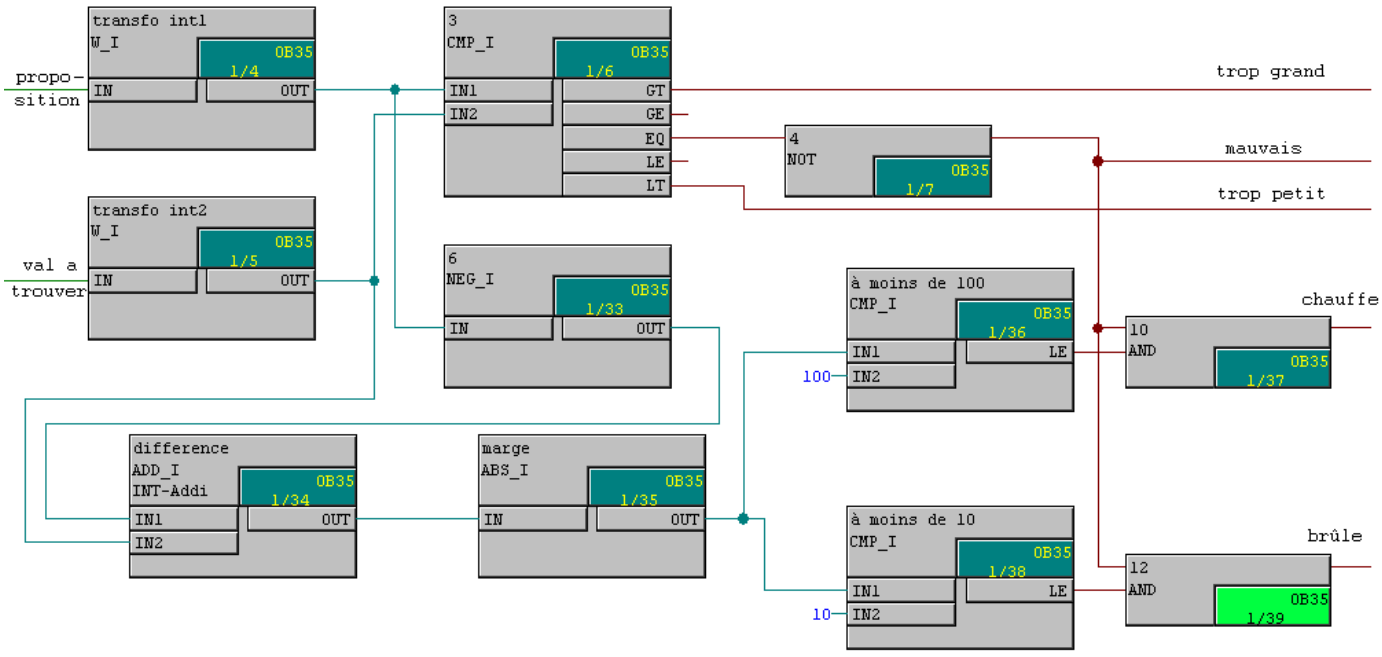
On peut maintenant créer A2, en ne prenant en compte une modification qu'au front montant du bouton de validation :



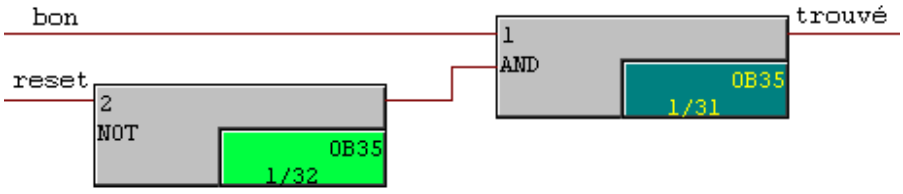
A3 compare les deux valeurs :



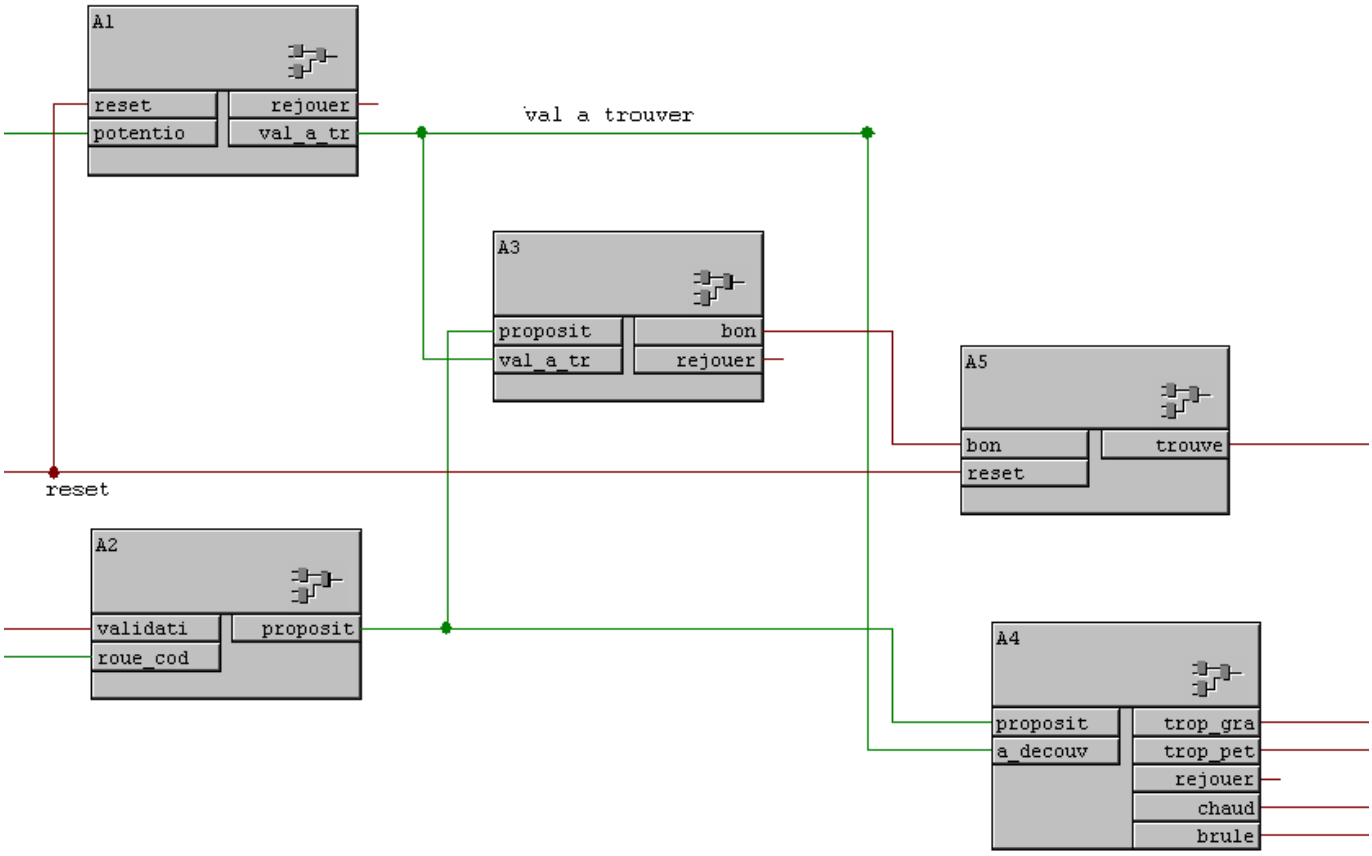
A4 compare aussi, et donne plus d'informations. Comme je n'ai pas géré le séquencement des diagrammes, les sorties revérifient si le résultat n'est pas bon pour ne donner les informations que dans ce cas :



Pour A5, pas de problème :

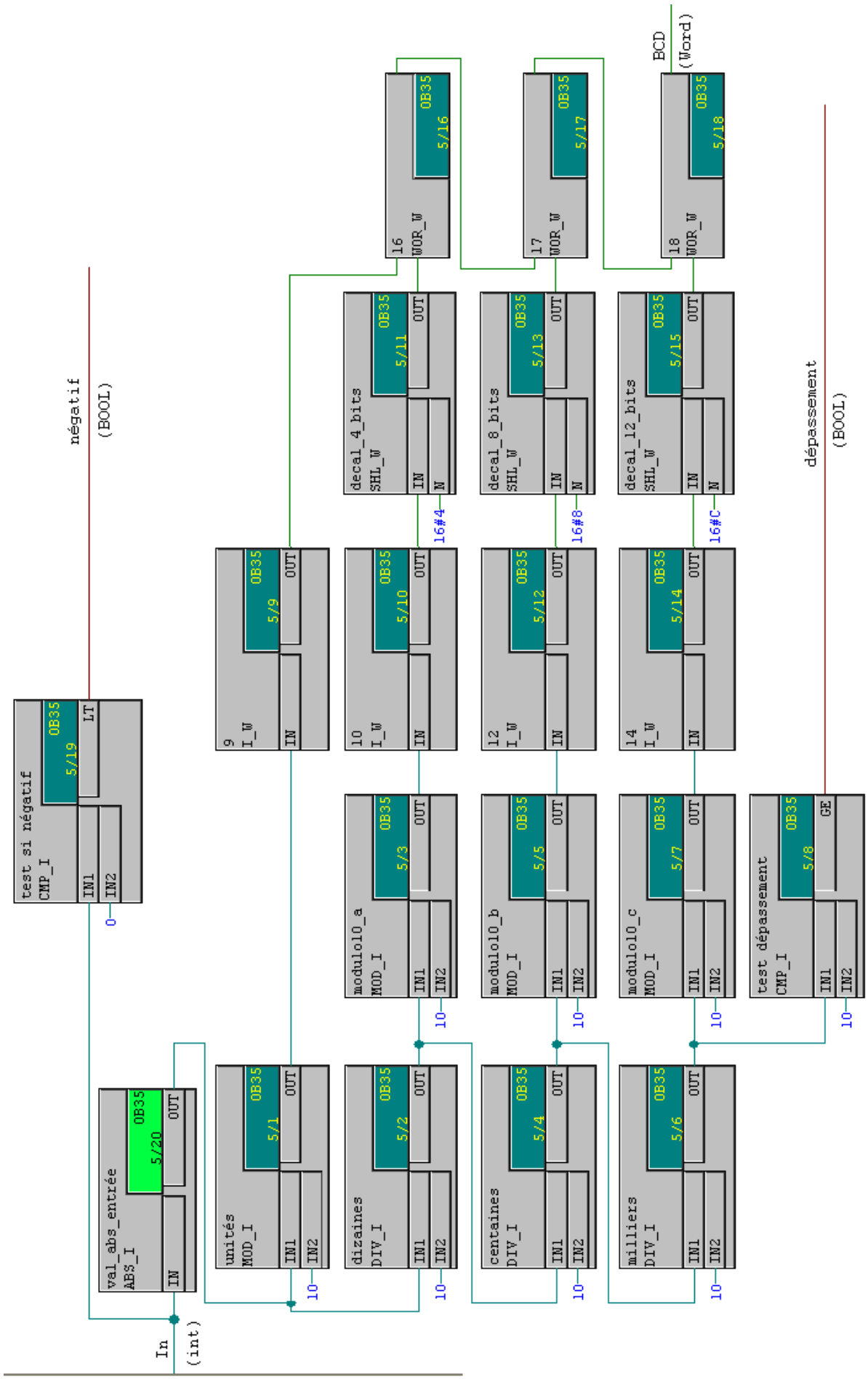


Il ne reste plus que A0. C'est dans A0 que l'on connecte les IN/OUT des différents diagrammes, mais aussi les entrées, sorties et mémos de l'automate (non représenté ici) :

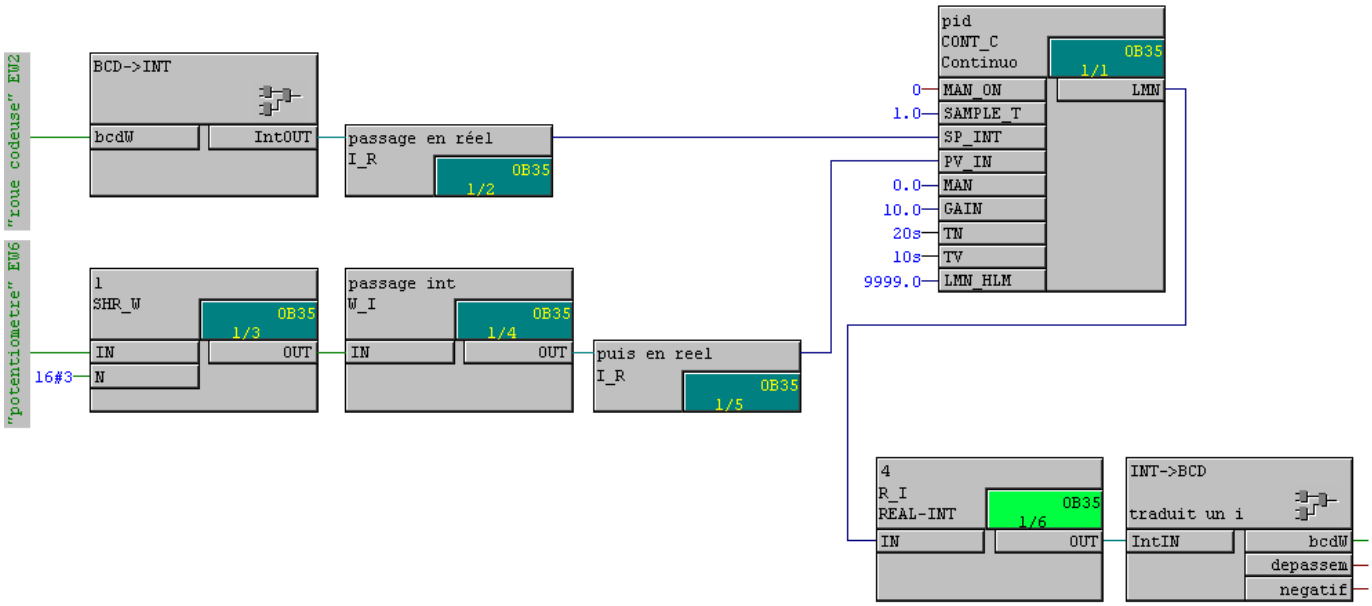


Dernier exemple : asservissement PID

J'ai testé le projet proposé dans la documentation. Il est très simple mais fonctionne bien. Avec des gains trop importants, on arrive à des instabilités. Mettons une consigne sur la roue codeuse (pour simplifier la tâche de l'utilisateur, je la considère en décimal, il faut donc convertir cette valeur BCD en un entier : je l'ai déjà fait dans l'exemple précédent, il ne me reste plus qu'à l'insérer ici).. L'automate doit fournir une valeur (malheureusement, nous n'avons pas de module de sortie analogique, je l'affiche), et nous vérifions par un capteur ce qui ce passe réellement (ici, on va jouer sur le potentiomètre, mais il faut faire un décalage de 3 bits car c'est ainsi que c'est câblé chez nous). On pourra regarder, en fonction des paramètres appliqués, la réaction aux variations du capteur (pour simplifier, plus longtemps on s'éloigne de la consigne, plus il modifie sa valeur de sortie, jusqu'à


négatif
(BOOL)
dépassement
(BOOL)
BCD
(Word)
In
(int)
test si négatif
CMP_I
OB35
5/19
LT
IN1
IN2
0
val_abs_entrée
ABS_I
OB35
5/20
OUT
IN
9
I_W
OB35
5/9
OUT
IN
16
WOR_W
OB35
5/16
decal_4_bits
SHL_W
OB35
5/11
OUT
IN
N
16#4
10
I_W
OB35
5/10
OUT
IN
modulo10_a
MOD_I
OB35
5/3
OUT
IN1
IN2
10
17
WOR_W
OB35
5/17
decal_8_bits
SHL_W
OB35
5/13
OUT
IN
N
16#8
12
I_W
OB35
5/12
OUT
IN
modulo10_b
MOD_I
OB35
5/5
OUT
IN1
IN2
10
18
WOR_W
OB35
5/18
decal_12_bits
SHL_W
OB35
5/15
OUT
IN
N
16#C
14
I_W
OB35
5/14
OUT
IN
modulo10_c
MOD_I
OB35
5/7
OUT
IN1
IN2
10
test dépassement
CMP_I
OB35
5/8
GE
IN1
IN2
10
unités
MOD_I
OB35
5/1
OUT
IN1
IN2
10
dizaines
DIV_I
OB35
5/2
OUT
IN1
IN2
10
centaines
DIV_I
OB35
5/4
OUT
IN1
IN2
10
milliers
DIV_I
OB35
5/6
OUT
IN1
IN2
10

atteindre son mini (0) ou maxi (LMN_HLM que j'ai fixé à 9999). La valeur est envoyée à l'afficheur, toujours pour simplifier je la transforme en BCD : les unités sont le reste de la division par 10; puis je divise le nombre par 10, j'en extrais les unités (qui sont les dizaines du nombre initial), je redivise par 10 et j'en extrais les unités (qui sont les centaines du nombre initial), et je recommence pour les milliers. Les 4 valeurs obtenues sont correctement décalées puis mises en un seul mot par un OU. Si le nombre de milliers est supérieur à 10, on dépasse la capacité de notre afficheur, j'allume un bit pour prévenir (on n'est pas obligé de le visualiser). J'ai aussi géré les EN/ENO mais je ne le retranscris pas ici par manque de place :



Conclusion

Ces exemples ont été traités pour découvrir le langage CFC. Dans le cas d'un automatisme simple, ce langage est plus gourmand que les langages classiques (STEP 7). Par contre c'est le langage principalement utilisé dans PCS7, utilisé pour les problèmes d'ingénierie complexe.