# Document de formation
# pour une solution complète d'automatisation
# Totally Integrated Automation (T I A)

## *MODULE B2*

## Traitement des valeurs analogiques

Nous tenons à remercier la firme Michael Dziallas Engineering, la Fachhochschule de Cologne et les enseignants d’écoles professionnelles ainsi que tous ceux qui ont participé à l’élaboration de ce document.

**PAGE :**



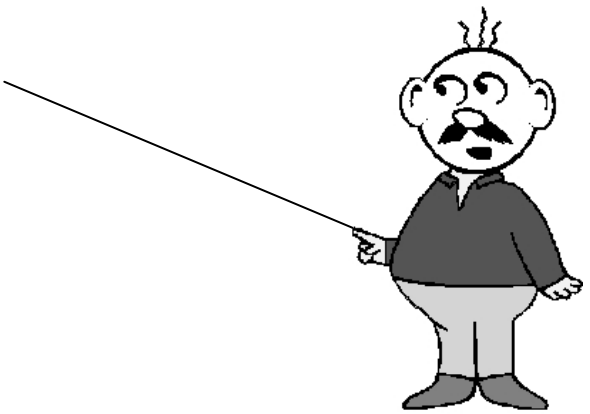
**On trouvera les symboles suivants dans ce module :**

**Information**

**Programmation**

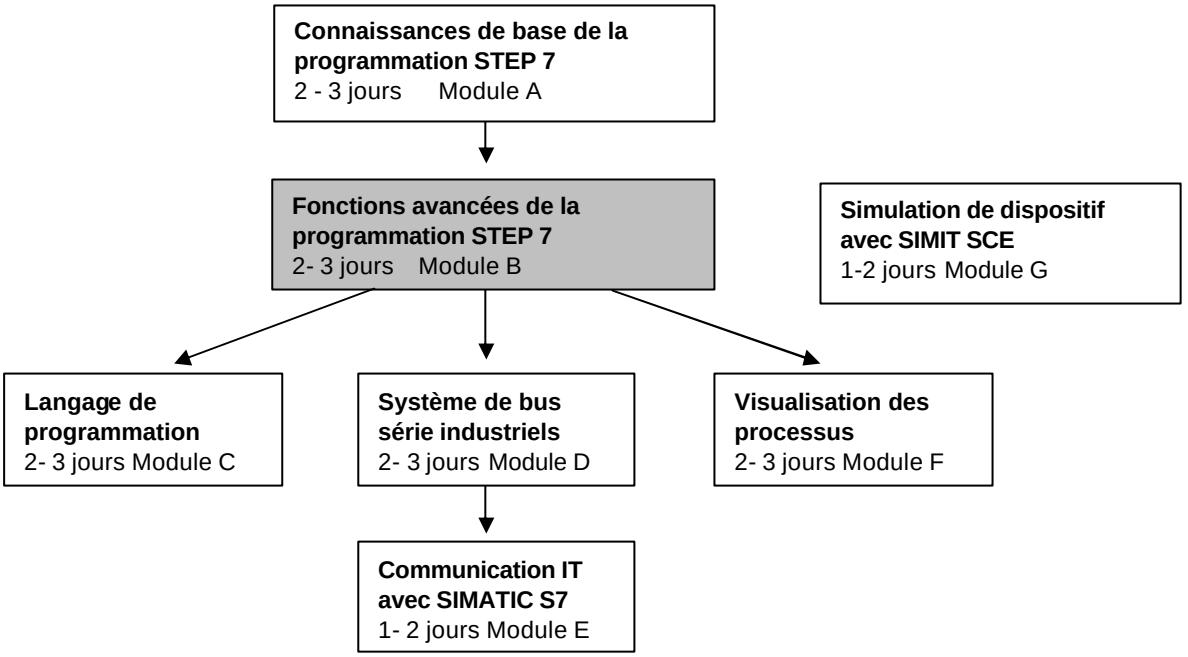
**Exemple d'application**

**Indications**



## 1. AVANT-PROPOS

Le contenu du module B2 est assigné à l'unité **‚Fonctions avancées de la programmation STEP 7'**.




**Objectif :**

Le lecteur va apprendre dans les étapes suivantes à extraire, traiter et produire des valeurs analogiques dans SIMATIC S7.

- Signaux analogiques
- Types de données en STEP 7
- Opérations mathématiques
- Conversion de types de données en STEP 7
- Lire et normaliser des valeurs analogiques
- Normaliser et produire des valeurs analogiques

**Pré-requis :**

Les connaissances suivantes sont requises pour l'étude de ce module :

- Connaissances pratiques de Windows 95/98/2000/ME/NT4.0/XP
- Fondements de la programmation d'automates programmables avec STEP 7 (par ex. Module A3 - ‚Startup' Programmation d'automates avec STEP 7)

**Logiciels et matériels nécessaires**

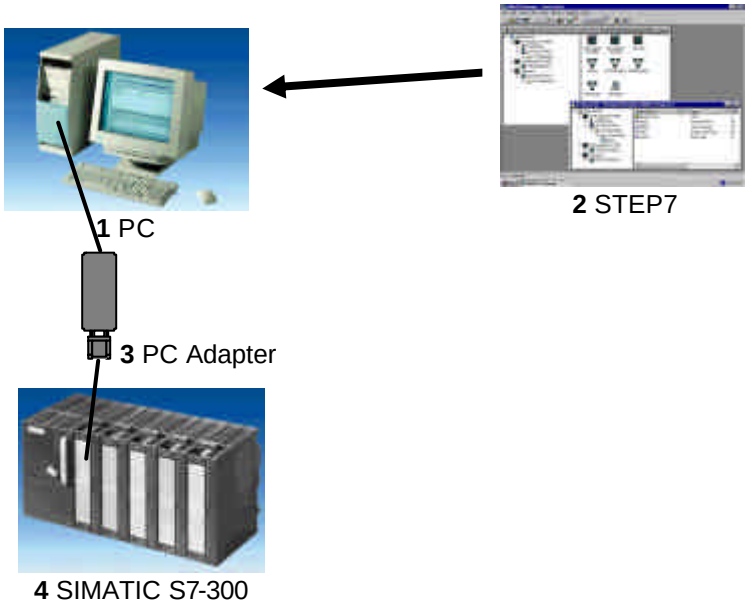
**1** PC, systèmes d'exploitation Windows 95/98/2000/ME/NT4.0/XP avec comme configuration
- Minimale : 133MHz et 64Mo RAM, espace disque env. 65 MB
- Optimale : 500MHz et 128Mo RAM, espace disque env. 65 MB

**2** Logiciel STEP 7 V 5.x

**3** Interface MPI pour le PC (par ex. PC-Adapter)

**4** Automate SIMATIC S7-300 avec au moins un module d'entrées/sorties analogiques auquel est connecté via une entrée à valeur analogique un potentiomètre ou un signaleur de signal analogique. En outre, il doit être connecté un afficheur de valeur analogique à au moins une sortie analogique.

Exemple de configuration :
- Alimentation : PS 307 2A
- CPU : CPU 314
- Entrées numériques : DI 16x DC24V
- Sorties numériques : DO 16x DC24V / 0,5 A
- Entrées/sorties analogiques : AI 4/ AO 2 x 8bits



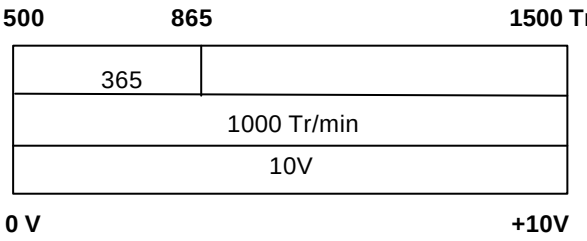
## 2. SIGNAUX ANALOGIQUES

Contrairement à un signal binaire, qui peut seulement prendre les deux états de signal „Tension disponible +24V„ et „Tension indisponible 0V„, les signaux analogiques peuvent prendre de nombreuses valeurs à l'intérieur d'un domaine donné. L'exemple typique de capteur analogique est le potentiomètre. Selon la position du curseur sur le potentiomètre, on peut lui faire prendre une résistance quelconque en-dessous de sa résistance maximale.

Exemples de grandeurs analogiques en technique des régulations :

- Température -50 ... +150°C
- Débit 0 ... 200l/min
- Nombre de tours 500 ... 1500 U/min
- etc.

Ces caractéristiques physiques sont converties en tensions électriques, courants ou résistances à l'aide d'un convertisseur de mesure. Si l'on doit déterminer par exemple un nombre de tours, on peut convertir un domaine de nombre de tours de 500 à 1500 tours/min dans un domaine de tension de 0 à +10V par un convertisseur de mesure. Si l'on mesure un nombre de tours de 865 tours/min, le convertisseur de mesure nous donne ainsi une tension de + 3,65 V.

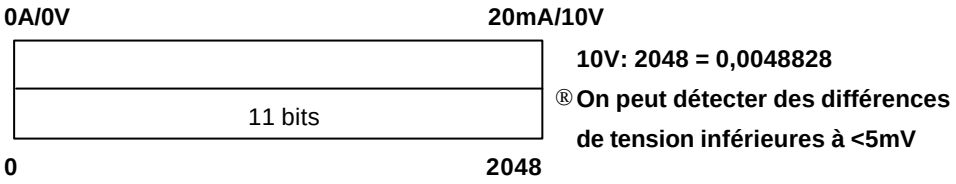



**10tr : 1000 tr/min = 0,01 V/U/min**
**365 tr/min x 0,01 tr/U/min = 3,65**

Ces tensions, courants ou résistances électriques sont ensuite connectés à un module analogique qui numérise ce signal.

**Indications :** Certains modules analogiques peuvent traiter différentes catégories de signaux. Celles-ci doivent être paramétrées dans la configuration matérielle. Veillez s'il vous plaît à respecter les instructions du manuel utilisateur associé à l'appareil.

Si des données analogiques sont traitées par un automate, la valeur de tension/courant/impédance lue doit être convertie en une information numérique. Cette conversion est connue sous le nom de conversion analogique/numérique (CAN (A/D)). Ceci signifie qu'une valeur de tension valant par exemple 3,65V doit être traduite dans une série de chiffres binaires comme information. Plus on utilise de chiffres binaires pour la représentation numérique, meilleure est la résolution. Si on avait par exemple un seul bit à disposition pour le domaine de tension 0 ... +10V, on pourrait seulement dire que la tension mesurée se trouve dans le domaine 0 .. +5V ou dans +5V ... +10V. Avec 2 bits, on pourrait déjà diviser le domaine en 4 sous domaines individuels, soit 0 ... 2,5 / 2,5 ... 5 / 5 ... 7,5 / 7,5 ... 10V. Les convertisseurs A/N (A/D) usuels convertissent avec 8 ou 11 bits en technique des régulations. Les convertisseurs 8 bits ont 256 domaines élémentaires et les 11 bits une résolution de 2048 domaines élémentaires.




**10V: 2048 = 0,0048828**

® **On peut détecter des différences de tension inférieures à <5mV**

## 3. TYPES DE DONNEES DANS STEP7

Dans SIMATIC S7, il y a un certain nombre de types de données avec lesquels on peut représenter plusieurs formats de nombres. Dans la suite, on va détailler de manière exhaustive les types de données élémentaires.

| Type et description | Taille en bits | Format-option | Domaine et représentation des nombres depuis la plus petite jusqu'à la plus haute valeur | Exemple |
|---|---|---|---|---|
| BOOL (bit) | 1 | Texte Bool | TRUE/FALSE | TRUE |
| BYTE (octet) | 8 | Hexadécimal | B#16#0 à B#16#FF | B#16#10 |
| WORD (mot) | 16 | Dual | 2#0 à 2#1111_1111_1111_1111 | 2#0001_0000_0000_0000 |
| | | Hexadécimal | W#16#0 à W#16#FFFF | W#16#1000 |
| | | BCD | C#0 à C#999 | C#998 |
| | | Décimal (o.V.) | B#(0,0) à B#(255,255) | B#(10,20) |
| DWORD (double mot) | 32 | Dual | 2#0 à 2#1111_1111_1111_1111_1111_1111_1111_1111 | 2#1000_0001_0001_1000_1011_1011_0111_1111 |
| | | Hexadécimal | DW#16#0000_0000 à DW#16#FFFF_FFFF | DW#16#00A2_1234 |
| | | Décimal (o.V.) | B#(0,0,0,0) à B#(255,255,255,255) | B#(1,14,100,120) |
| INT (entier) | 16 | Décimal | -32768 à 32767 | 1 |
| DINT (entier,32 bits) | 32 | Décimal | L#-2147483648 à L#2147483647 | L#1 |
| REAL (réel) | 32 | IEEE réel | Limite supérieure : +/-3.402823e+38<br>Limite inférieure : +/-1.175495e-38 | 1.234567e+13 |
| S5TIME (temps Simatic) | 16 | Temps S7 par pas de 10 ms | S5T#0H_0M_0S_10MS à S5T#2H_46M_30S_0MS et S5T#0H_0M_0S_0MS | S5T#0H_1M_0S_0MS<br>S5TIME#1H_1M_0S_0MS |
| TIME (temps IEC) | 32 | Temps IEC par pas de 1ms, Entiers signés | -T#24D_20H_31M_23S_648MS à T#24D_20H_31M_23S_647MS | T#0D_1H_1M_0S_0MS<br>TIME#0D_1H_1M_0S_0MS |
| DATE (Date IEC) | 16 | Date IEC par pas d'un jour | D#1990-1-1 à D#2168-12-31 | DATE#1994-3-15 |
| TIME_OF_DAY (heure) | 32 | Heure par pas de 1ms | TOD#0:0:0.0 à TOD#23:59:59.999 | TIME_OF_DAY#1:10:3.3 |
| CHAR (caractères) | 8 | Caractères ASCII | ´A´, ´B´ etc. | ´B´ |

**Indication :** En traitement des signaux analogiques, les types de données **‚INT'** et **‚REAL'** jouent un rôle important car les valeurs analogiques extraites sont disponibles au format **‚INT'**, or, pour le traitement, il n'est question que de considérer les réels **‚REAL'** pour éviter des erreurs d'arrondis.

## 4. OPERATIONS DE CALCUL

### 4.1 CALCUL AVEC DES NOMBRES ENTIERS (INT ET DINT)

Les opérations mathématiques de base sont possibles avec des nombres entiers : addition, soustraction, multiplication et division. Les chiffres après la virgule sont toutefois ignorés ce qui mène à des erreurs d'arrondis lors de la division.

| Opération | Taille en bit | Fonction |
|---|---|---|
| +I | 16 | Additionne le contenu du mot de poids faible de l'ACCU 1 et 2 et enregistre le résultat dans le mot de poids faible de l'ACCU 1. |
| -I | 16 | Soustrait le contenu du mot de poids faible de l'ACCU 1 au contenu du mot de poids faible de l'ACCU 2 et enregistre le résultat dans le mot de poids faible de l'ACCU 1. |
| *I | 16 | Multiplie le contenu des mots de poids faible des ACCU 1 et 2 et enregistre le résultat (32 bits) dans l'ACCU 1. |
| /I | 16 | Divise le contenu du mot de poids faible d'ACCU 2 par le contenu du mot de poids faible de l'ACCU 1. Le résultat est enregistré dans le mot de poids faible d'ACCU 1. Le reste de la division est enregistré dans le mot de poids fort d'ACCU 1. |
| +D | 32 | Additionne le contenu de l'ACCU 1 et 2 et enregistre le résultat dans l'ACCU 1. |
| -D | 32 | Soustrait le contenu de l'ACCU 1 au contenu de l'ACCU 2 et enregistre le résultat dans l'ACCU 1. |
| *D | 32 | Multiplie le contenu des ACCU 1 et 2 et enregistre le résultat dans l'ACCU 1. |
| /D | 32 | Divise le contenu d'ACCU 2 par le contenu de l'ACCU 1 et enregistre le quotient dans ACCU 1. |
| MOD | 32 | Divise le contenu d'ACCU 2 par le contenu de l'ACCU 1 et enregistre le reste de la division dans ACCU 1. |

## 4.2 CALCUL AVEC DES NOMBRES REELS (REAL)

On peut calculer un grand nombre d'opérations mathématiques avec les réels. Dans ce format, les chiffres après la virgule sont toujours pris en compte.

| Opération | Fonction |
|---|---|
| +R | Additionne les réels (32 Bit, IEEE-FP) des ACCU 1 et 2 et enregistre le résultat sur 32 bits dans ACCU 1. |
| -R | Soustrait le réel (32 bits, IEEE-FP) d'ACCU 1 au réel (32 bits, IEEE-FP) dans ACCU 2 et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| *R | Multiplie le réel (32 bits, IEEE-FP) d'ACCU 1 par le réel (32 bits, IEEE-FP) dans ACCU 2 et enregistre le résultat sur 32 bits dans l' ACCU 1. |
| /R | Divise le réel (32 Bit, IEEE-FP) d'ACCU 2 par le réel (32 bits, IEEE-FP) d'ACCU 1. Le résultat sur 32 bits est enregistré dans l' ACCU 1. |
| ABS | Effectue l'opération valeur absolue d'un réel (32 bits, IEEE-FP) d'ACCU 1. Le résultat est enregistré dans ACCU 1. L'opération est effectuée sans considérer ou influencer le bit de statut. |
| SQRT | Calcule la racine carrée du réel (32 bits, IEEE-FP) d'ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| SQR | Calcule le carré du réel (32 bits, IEEE-FP) d'ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| LN | Calcule le logarithme népérien du réel (32 bits, IEEE-FP) d'ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| EXP | Calcule la valeur exponentielle (ExponenT I A l) du réel (32 bits, IEEE-FP) de base e et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| SIN | Calcule le sinus du réel (32 bits, IEEE-FP) d'ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1.. |
| COS | Calcule le cosinus du réel (32 bits, IEEE-FP) d'ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| TAN | Calcule la tangente du réel (32 bits, IEEE-FP) d'ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| ASIN | Calcule l'arcsinus du réel (32 bits, IEEE-FP) d'ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| ACOS | Calcule l'arccosinus du réel (32 bits, IEEE-FP) dans ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1. |
| ATAN | Calcule l'arctangente du réel (32 bits, IEEE-FP) dans ACCU 1 et enregistre le résultat sur 32 bits dans l'ACCU 1. |

## 4.3 OPERATIONS DE CONVERSION DE TYPES DE DONNEES

Comme les nombres ne sont pas souvent dans le format nécessaire au traitement, ceux-ci doivent être fréquemment adaptés à l'aide d'opérations de conversion.

| Opération | Fonction |
|---|---|
| BTI | Convertit un BCD en entier (16 bits). Cette opération convertit une valeur décimale codée en binaire dans ACCU 1 en un entier (16 bits). |
| BTD | Convertit un BCD en entier (32 bits). Cette opération convertit une valeur décimale codée en binaire dans ACCU 1 en un entier (32 bits). |
| ITB | Convertit un entier (16 bits) en un BCD. Cette opération convertit un entier (16 bits) stocké dans le mot de poids faible d'ACCU 1 en une valeur décimale codée en binaire. |
| ITD | Convertit un entier (16 bits) en un entier (32 bits). Cette opération convertit un entier (16 bits) stocké dans le mot de poids faible d'ACCU 1 en un entier (32 bits). |
| DTB | Convertit un entier (32 bits) en un BCD. Cette opération convertit un entier (32 bits) stocké dans le mot de poids faible d'ACCU 1 en une valeur décimale codée en binaire. |
| DTR | Convertit un entier (32 bits) en un réel (32 bits, IEEE-FP). Cette opération convertit un entier (32 bits) dans ACCU 1 en un réel (32 bits, IEEE-FP). |
| RND | Arrondit à l'entier le plus proche. Cette opération arrondit le nombre au plus proche entier.<br>Si le nombre se trouve pile entre un résultat pair et un résultat impair, l'opération choisit le résultat pair. |
| RND+ | Arrondit à l'entier supérieur. Cette opération arrondit le réel de telle sorte que l'entier obtenu soit le plus petit possible tout en étant supérieur ou égal au réel d'origine. |
| RND- | Arrondit à l'entier inférieur. Cette opération arrondit le réel de telle sorte que l'entier obtenu soit le plus grand possible tout en étant inférieur ou égal au réel d'origine. |
| TRUNC | Tronque. Cette opération utilise la partie entière du réel pour former l'entier. |

**Indication :** Dans le cas du traitement de valeurs analogiques, la valeur analogique est sous format **,INT'** et doit être convertie en réel **,REAL'** pour éviter les erreurs d'arrondis. Comme cela ne peut pas se faire directement, d'abord avec **,ITD'** pour convertir en **,DINT'** puis avec **,DTR'** pour convertir en **,REAL'**.

## 5. EXTRAIRE/PRODUIRE DES VALEURS ANALOGIQUES

Les valeurs analogiques sont lues et produites comme des mots d'informations dans l'automate. L'accès à ces mots s'effectue par les instructions :

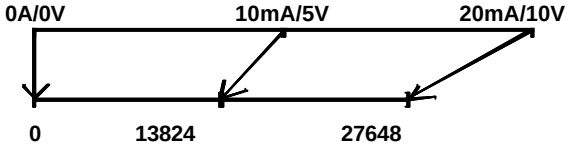
| | | |
|---|---|---|
| L | PEW x | Charger mot d'entrée analogique |
| T | PAW x | Transférer mot d'entrée analogique |

Chaque valeur analogique (« canal ») occupe un mot d'entrée et de sortie de périphérie. Le format est **‚INT'** , un nombre entier.

L'adressage des mots d'entrée/sortie s'ajuste suivant les adresses de début du module.
Si on connecte le module analogique à l'emplacement 4, il prend alors l'adresse de début par défaut 256. L'adresse de début de chaque module analogique supplémentaire s'élève de 16 par emplacement. Ces adresses de début par défaut peuvent être consultées dans la table de configuration matérielle sous vue détaillée.

L'adresse de la première entrée analogique serait ainsi PEW 288 avec le module analogique à l'emplacement 6, celle de la deuxième PEW 290, celle de la première sortie analogique serait ensuite PAW 288 etc...

La transformation de valeurs analogiques pour la poursuite du traitement dans l'automate (numérisé) est identique que ce soit les entrées ou sorties analogiques.
Pour les modules SM334, à 4 entrées analogiques et 2 sorties analogiques, avec les domaines de valeurs analogiques 0 à 10V et 0 à 20mA, voilà l'allure du domaine de valeurs numérisées :




**Domaine nominal de la valeur analogique**

**Valeur numérisée pour traitement dans le SPS**

Les valeurs numérisées doivent souvent encore être normalisées par des traitements correspondants dans l'automate.

## 5.1. LIRE ET NORMALISER DES VALEURS ANALOGIQUES

S'il y a une valeur d'entrée analogique comme valeur numérisée, celle-ci doit tout d'abord être normalisée, avant de pouvoir être traitée dans l'automate programmable.
De même, la production analogique s'effectue habituellement sur le mot de sortie périphérique après une normalisation de la valeur de sortie.

Dans les programmes STEP7, on aura recours aux opérations de calcul pour la normalisation.
Afin qu'elle puisse s'effectuer le plus exactement possible, les valeurs doivent encore être converties en valeurs normalisées dans le type REAL pour minimiser les erreurs d'arrondis.

**Exercice :**

Dans l'exemple suivant, on lit une valeur de 0 à 10V avec un module analogique SM334 sur l'emplacement 6 (PEW288). Au début, la valeur est un ENTIER (16 bits) et doit être normalisée entre 100 et 1000 en format réel et stockée dans le double mot mémento MD10.

**Solution en LIST :**

```
L     PEW 288      // Lire Valeur analogique 0-10V correspondant aux entiers 0-27648 (sur 16 bits)
ITD                // Convertir Valeur d'entier (16 bits) en entier (32 bits)
DTR                // Convertir Valeur d'entier (32 bits) en réels
L     2.7648e+4    //
/R                 // Division par entier 27648
L     9.000e+2     //
*R                 // Multiplication par réel 900 (1000-100)
L     1.000e+2     //
+R                 // Addition à l'entier 100 (décalage (Offset) )
T     MD10         // Valeur normée 100 à 1000 en format réel
```

## 5.2. NORMALISER ET PRODUIRE DES VALEURS ANALOGIQUES

Si une valeur normalisée doit être produite par un module de sortie analogique, celle-ci doit tout d'abord être convenablement normalisée.

Dans les programmes STEP7, on aura recours aux opérations de calcul pour la normalisation. Ceci s'effectue encore avec le type de données REAL, pour minimiser les erreurs d'arrondis. C'est seulement après que cette valeur est arrondie à un nombre entier. Les chiffres après la virgule sont ainsi perdus.

**Exercice :**

Dans l'exemple suivant, on a une valeur entre 100 et 1000 stocké sous format réel sur le mot double mémento MD20 devant être normalisée et produite de 0 à 10V par un module analogique SM334 (PAW288).

**Solution en LIST :**

```
L     MD20        // Valeur 100 à 1000 en format réel
L     1.000e+2    //
-R                // Soustraction de 100 (décalage (Offset) )
L     9.000e+2    //
/R                // Division par le réel 900
L     2.7648e+4   //
*R                // Multiplication par le réel 27648
RND               // Arrondir à l'entier
T     PAW 288     // Extraire entier 0-27648 (16 bits) correspondant à la valeur analogique 0-10 V
```