CHAP 8 CNA – CAN

S2

GEII Toulouse III

2020- 2021

A. Nketsa

1- Convertisseur numérique – Analogique (CNA)

1- Introduction

Dans la nature, les tensions sont analogiques.
En informatique embarquée, nous travaillons en numérique.
Pour passer du monde numérique au monde analogique, on utilise un composant qui a pour but de transformer un nombre en une tension analogique.
Dans ce cours, nous allons considérer que ce composant est disponible.
Notre travail va consister à :

- trouver la relation qui existe entre la tension analogique à générer et le nombre produit par l'ordinateur
- écrire le nombre sur le composant.

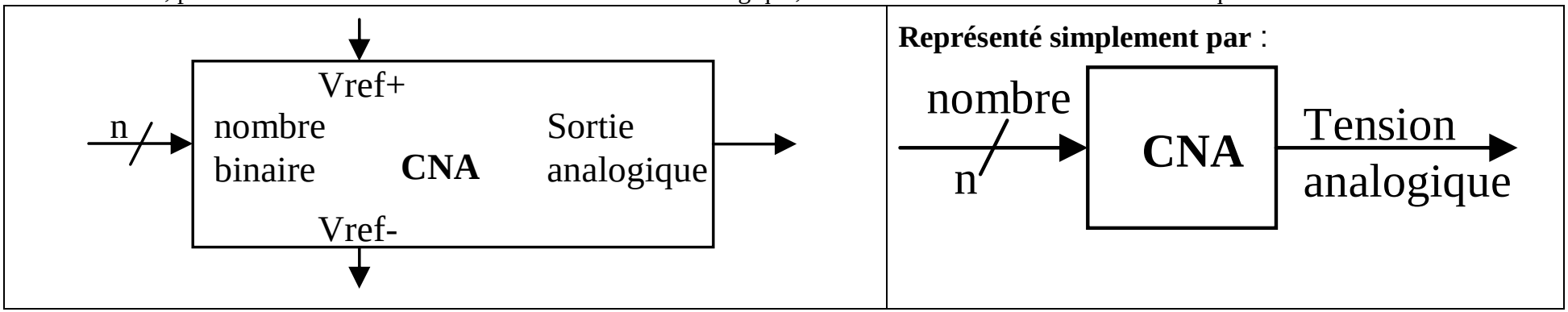
On sait que le processeur envoie des informations à l'extérieur par l'intermédiaire du port de sortie qui conserve la dernière valeur écrite

2- Définition

Un convertisseur numérique-analogique est un dispositif qui permet de transformer un nombre en une tension analogique.
Dans ce cours, nous allons traiter de l'utilisation des CNA et leurs applications. Nous ne traiterons pas de la technologie de ces composants.

3- Schéma de principe

Sur ce schéma, pour convertir un nombre binaire en une tension analogique, il faut définir les tensions de référence qui sont les bornes de la conversion



4- Caractéristiques

Un CNA est caractérisé par

- n, **le nombre de bits du nombre binaire** à convertir en une tension analogique
- **la dynamique** : c'est la plage de la tension analogique de sortie.

  Cette plage est définie par la différence entre deux tensions de référence Vref+ et Vref- $\Rightarrow$ **Dynamique = Vref+ - Vref-**

- **la résolution =** $\dfrac{\text{dynamique}}{2^n - 1}$
- **la linéarité** $\dfrac{S2 - S1}{n2 - n1}$ est une constante. S1 et S2 sont les valeurs de la sortie respectivement pour n2 et n1.

  donc la relation entre l'entrée et la sortie est une droite.

- **la fonction de transfert**

  Puisque le CNA respecte le principe de linéarité, la relation entrée-sortie est de la forme

  y = ax + b **avec y = sortie = tension analogique à générer** **x= nombre écrit sur le port**

  $2^n - 1$ converti en Vref+ $\Rightarrow$ Vref+ = a * $(2^n - 1)$ + b

  0x0000 converti en Vref- $\Rightarrow$ Vref- = a * (0) + b

  tension_analogique = a * nombre + b **a=** $\dfrac{Vref+ - Vref-}{2^n - 1}$ b = Vref-

  tension_analogique = $\dfrac{Vref+ - Vref-}{2^n - 1}$ *** nombre +** Vref-

- **le temps de conversion** (Ce temps n'est pas souvent pris en compte)

Application aux CNA utilisés en TD – TP

Nous disposons de 4 CNA sur la carte à base du microcontrôleur C167.

Caractéristiques de chacun des 4 CNA

- **n = 12.** CNA de 12bits

- Vref+ = +5v Vref- = -5v **Dynamique = 5v - -5v = 10v**

- **résolution = 10v/4095**

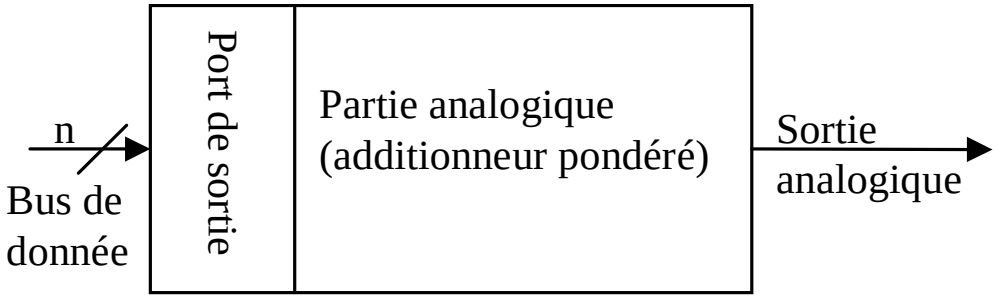
- Fonction de transfert : **tension_analogique(volts) =** $(\frac{10v}{4095} * \text{nombre}) - 5v$

5- Programmation

Nous travaillons avec le microcontrôleur C167.
Un CNA est vu comme un port de sortie, c'est_à-dire que **la dernière valeur écrite sur le port est mémorisée**

Nous disposons de 4 CNA implantés dans l'espace mémoire.
Pour accéder à ces ports, on utilise les pointeurs.

Vue entrée-sortie d'un CNA



Déclaration :

unsigned int far *CNA = adresse_cna0; // **far** pour indiquer l'adresse étendue sur 24bits au lieu de 16bits

l'adresse du premier convertisseur est adresse_cna0

l'adresse du convertisseur i est adresse_cnai = adresse_cna0 +2*i

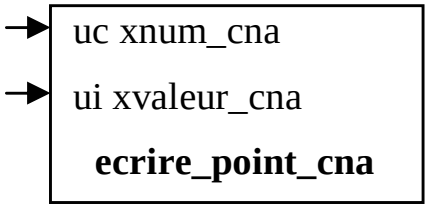
Chaque convertisseur (12bits) occupe 2 octets

Fonction d'accès aux CNA

Nous allons écrire la fonction qui permet d'accéder aux différents CNA

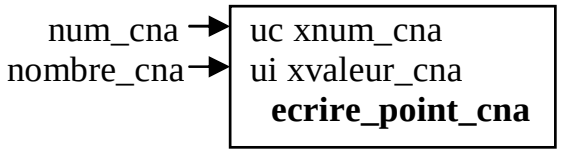
void ecrire_point_cna(unsigned char num_cna, unsigned int valeur_cna)

```
{ unsigned int far *CNA = 0xffff00;
   CNA[(num_cna & 0x0003)] = valeur_cna;
}
```



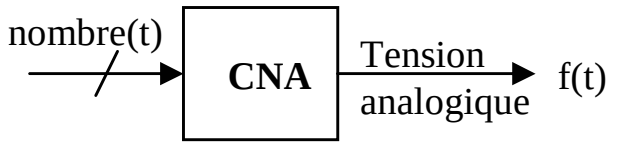
Donc par la suite, l'utilisation d'un CNA consistera à juste à appeler la fonction ecrire_point_cna avec les paramètres adéquats.

ecrire_point_cna(num_cna, nombre_cna);



6- Applications des CNA

L'application des CNA concerne la génération de signaux.
Un CNA peut être utilisé pour générer des signaux difficilement réalisables en électronique analogique seule.

Principe de génération des signaux On veut générer un signal analogique variable en amplitude dans le temps. La tension analogique souhaitée peut s'écrire **f(t).** Il suffit de calculer **nombre(t)** que l'on écrira sur le convertisseur numérique analogique qui va générer f(t).	

Cette fonction va être caractérisée par :

- ses amplitudes (maximale et minimale),

- le temps, t, qui est l'instant de génération du cette amplitude

- et la durée de génération de la fonction, tmax.

Si le signal est périodique alors tmax= periode du signal

Pour générer ce signal numériquement,

- le signal doit être généré point par point.
  - ☞ Chaque point correspondant à un instant t = k. Δt
  - ☞ Chaque point correspond à une marche d'escalier (la valeur de la fonction écrite sur le port de sortie à l'instant indiqué t)
  - ☞ Chaque point est mémorisé sur le port de sortie pendant durée Δt

  Pour cela, l'appel de la fonction ecrire_point_cna() génère la marche d'escalier.

  La dernière valeur écrite sur le CNA est mémorisée.

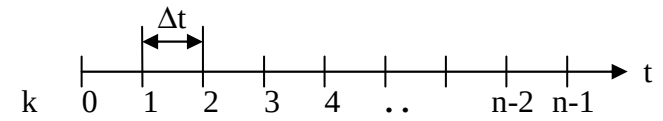
- Les points successifs seront séparés par une durée, Δt.
  Plus cette durée sera petite plus le signal généré se rapprochera du signal analogique idéal. Dans la pratique, on utilise un filtre passe bas pour lisser les marches d'escalier de manière à obtenir le signal analogique souhaité.

- l'amplitude (maximale et minimale) doit être dans la dynamique du convertisseur. C'est-à-dire que

  Vref- ≤ amplitude minimale ≤ Vref+ et Vref- ≤ amplitude maximale ≤ Vref+

  et amplitude minimale ≤ amplitude maximale

- le temps numérique (on dit aussi discrétisé (ou échantillonné)) doit être découpé en instants utiles pour des intervalles Δ**t** .
  On peut alors écrire **t = k*Δt où k est un entier qui varie de k0 à kmax avec tmax = kmax*Δt**

Discrétisation du temps ou échantillonnage du temps	

Calcul du nombre pour la génération de f(t)

De la fonction de transfert du CNA : **f(t) = (10v/4095)*nombre(t) – 5v,**

On en déduit (1) **nombre(t) =** $\dfrac{((f(t)+5)*4095)}{10}$

En considérant que t est discrétisé, on peut s'écrire **t= k*Δt**

La relation (1) devient alors (2) **nombre(k* Δt) =** $\dfrac{((f(k*\Delta t)+5)*4095)}{10}$

La génération de cette tension analogique consiste à faire une boucle dans laquelle :

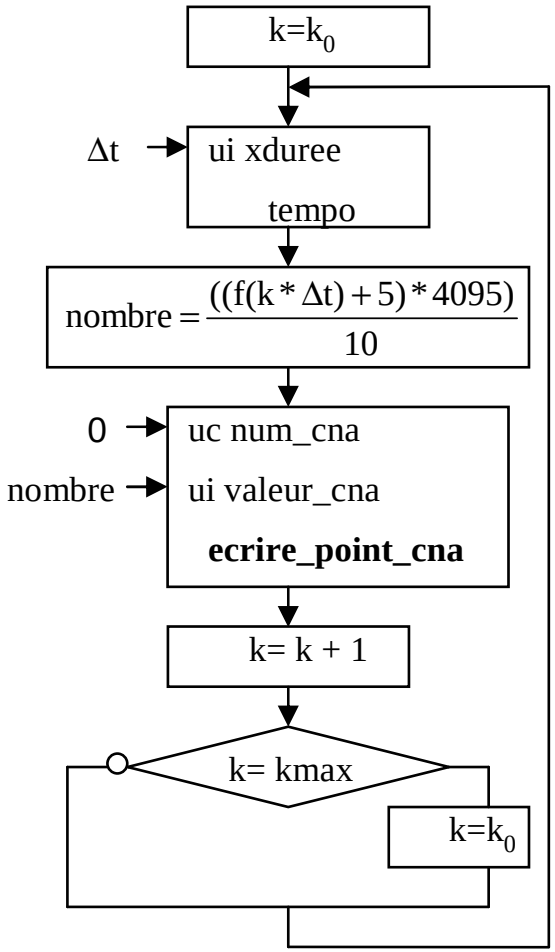
- on fait varier k de k0 à kmax
- pour chaque k, on calcule nombre(k*Δt) à partir de la relation (2),
- on écrit le résultat, nombre, sur le convertisseur numérique-analogique (CNA)
- on attend Δt pour faire le calcul pour un autre point

  Δt est une temporisation qui peut être réalisée par logiciel ou par timer (solution conseillée): on met le Timer en mode rechargeable ou générateur de signaux carrés.

On obtient les deux organigrammes suivants pour la sortie sur le CNA0 :

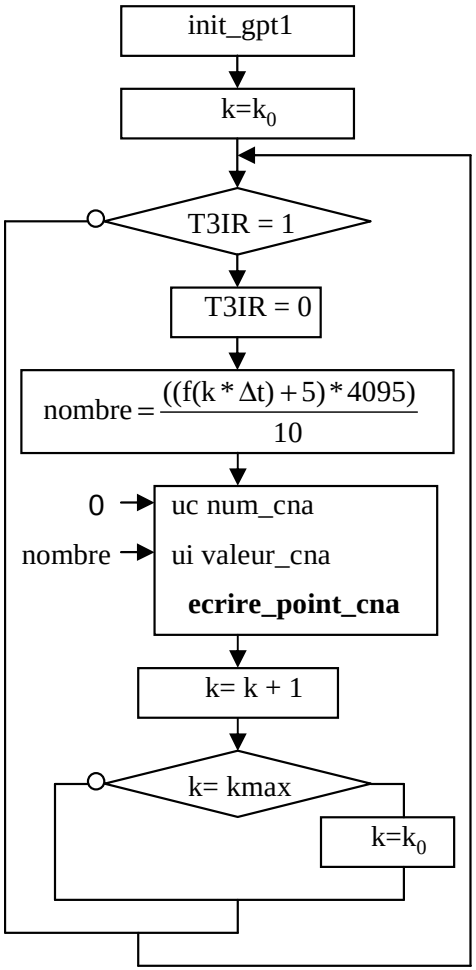
Organigramme sans synchronisation

Temporisation logicielle



Organigramme avec synchronisation

Temporisation réalisée par timer

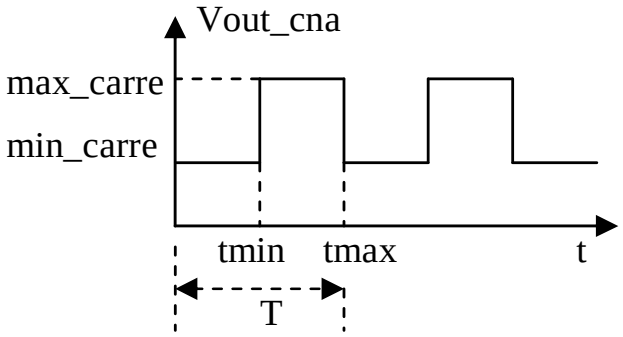


Exemples de génération des signaux

a) Génération d'un signal carré

f(t) = min_carre pour 0 ≤ t < tmin
max_carre pour tmin ≤ t ≤ tmax

La fonction est périodique de période
T= tmin + tmax

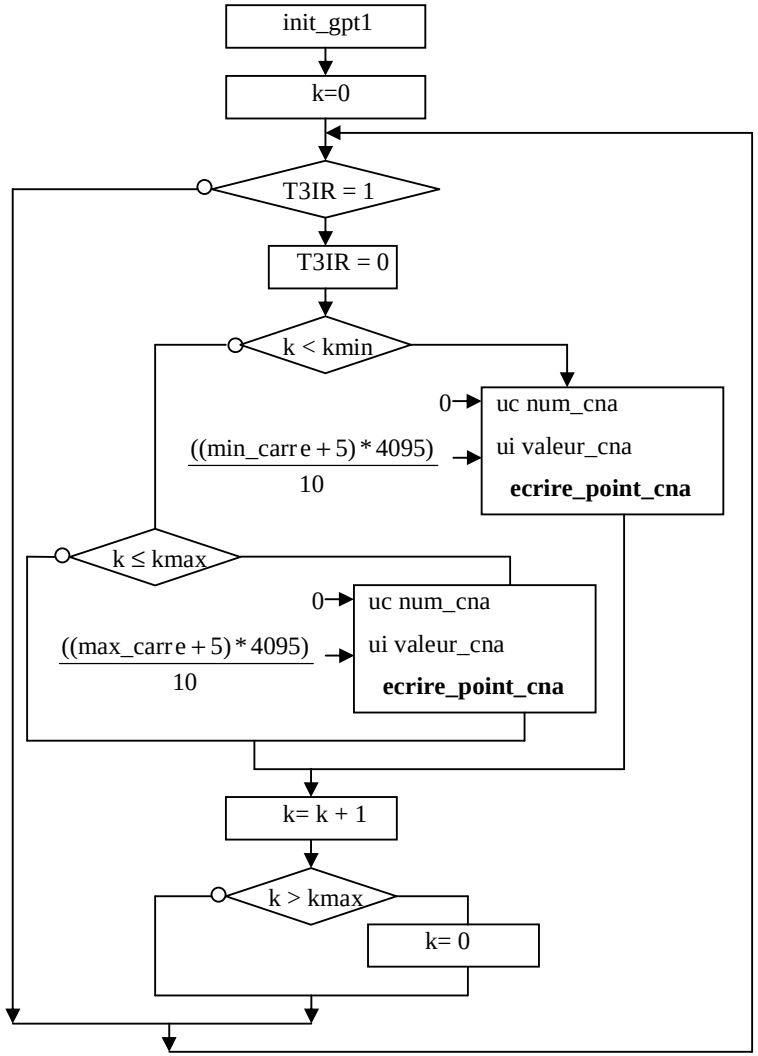


1- Conditions préliminaires à vérifier

-5v ≤ min_carre ≤ +5v
et -5v ≤ max_carre ≤ +5v
et min_carre < max_carre

2- Organigramme (nous considérons que Δt est généré par le timer T3 en mode rechargement automatique)

tmin = kmin * Δt tmax = kmax * Δt



b) Génération d'une rampe répétitive qui va de -5v à +5v en 4095 pas en commençant par 0.

1- Conditions préliminaires à vérifier

Les bornes du signal à générer sont dans la dynamique du CNA

Δt sera la durée de la marche d'escalier qui correspond à la durée d'un pas.

Elle sera générée par le timer T3.
On peut ainsi calculer la période du signal généré
Comme la durée du signal n'est pas précisée, nous allons considérer $\Delta t = 1\mu s$.

Equation de la droite de variant de -5v à +5v en 4095;

f(t) = ax +b
x= 0 $\Rightarrow$ -5 = a*0 + b
x=4095 $\Rightarrow$ +5 = a*4095 + b

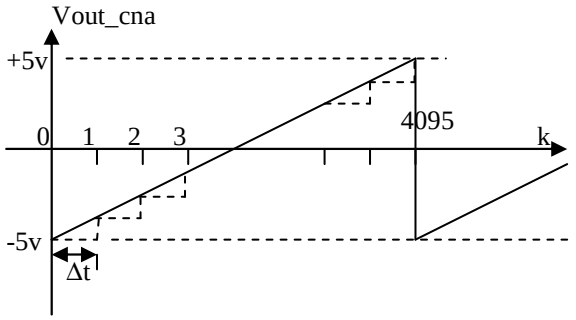
d'où f(t) = $\frac{10 * t}{4095} - 5$

Avec la forme discrétisée du temps, la relation (2) devient

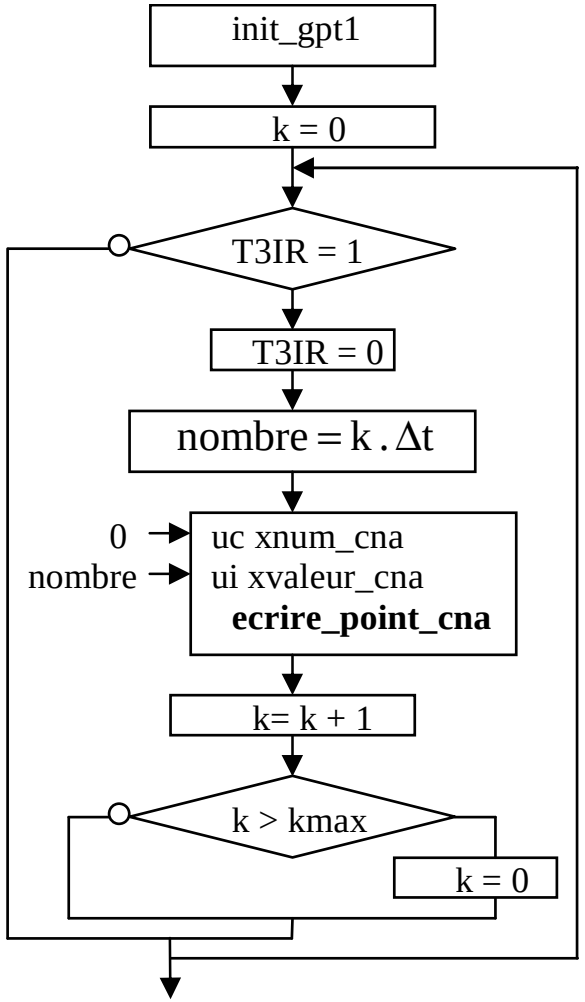
$$\text{nombre}(k^* \Delta t) = \frac{((f(k * \Delta t) + 5) * 4095)}{10} = \frac{((\frac{10 * k * \Delta t}{4095} - 5) + 5) * 4095}{10} = k^* \Delta t$$

k varie de 0 à 4095 avec kmax = 4095

Courbe obtenue



Organigramme du programme



c) Génération d'une fonction sinusoïdale

f(t) = Asin wt w = $\frac{2\Pi}{T}$

1) Conditions préliminaires

min_signal = -A

max_signal = A

avec $|A| \leq |Vref+|$

2) t = k . Δt tmax = T = kmax . Δt

nombre(k* Δt) = $\frac{((\text{Asin}(w.(k.\Delta t))) + 5) * 4095)}{10}$

En général, il faut trouver un compromis entre kmax et Δt avec T = kmax * Δt

Plus Δt est petit, plus il y a de points de génération

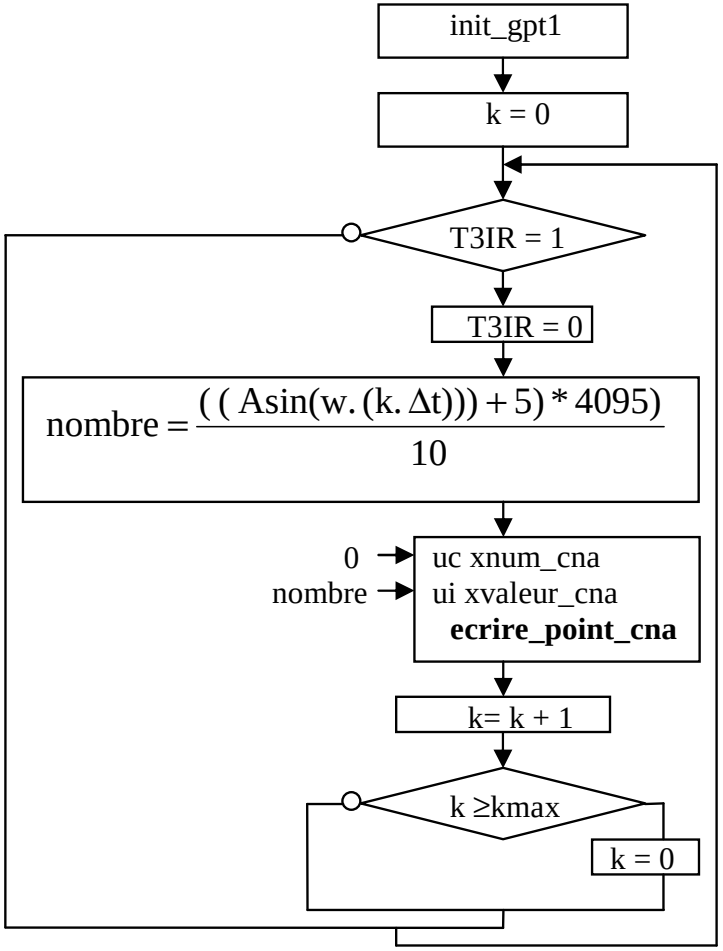
Plus Δt est grand, moins il y aura des points de génération et marches d'escaliers seront larges.

Le programme est le même que celui de la rampe, il suffit d'adapter les paramètres. : kmax et Δt

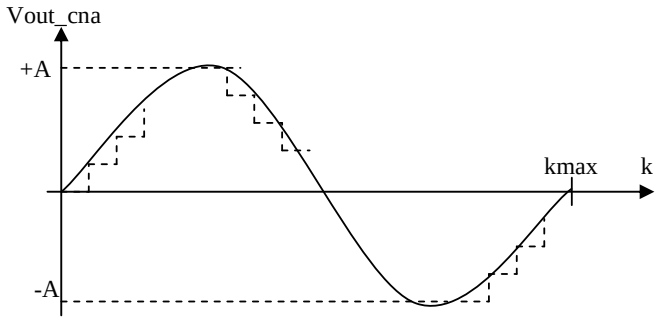
Connaissant kmax (nombre de marches d'escalier)

On en déduit Δt = $\frac{T}{kmax}$

Organigramme du programme



Courbe générée



On constate que le calcul du sinus prend du temps. Ceci va se traduire par une limitation de la fréquence du sinus à générer. Pour éviter cette perte de temps, on fait le calcul hors ligne et on stocke la valeur des échantillons dans un tableau, par exemple

unsigned char table_sinus[kmax]={v0, v1, v2, .. v_{kmax-1} **};**

Avec vk = $\frac{((\text{Asin}(w.(k.\Delta t))) + 5) * 4095)}{10}$

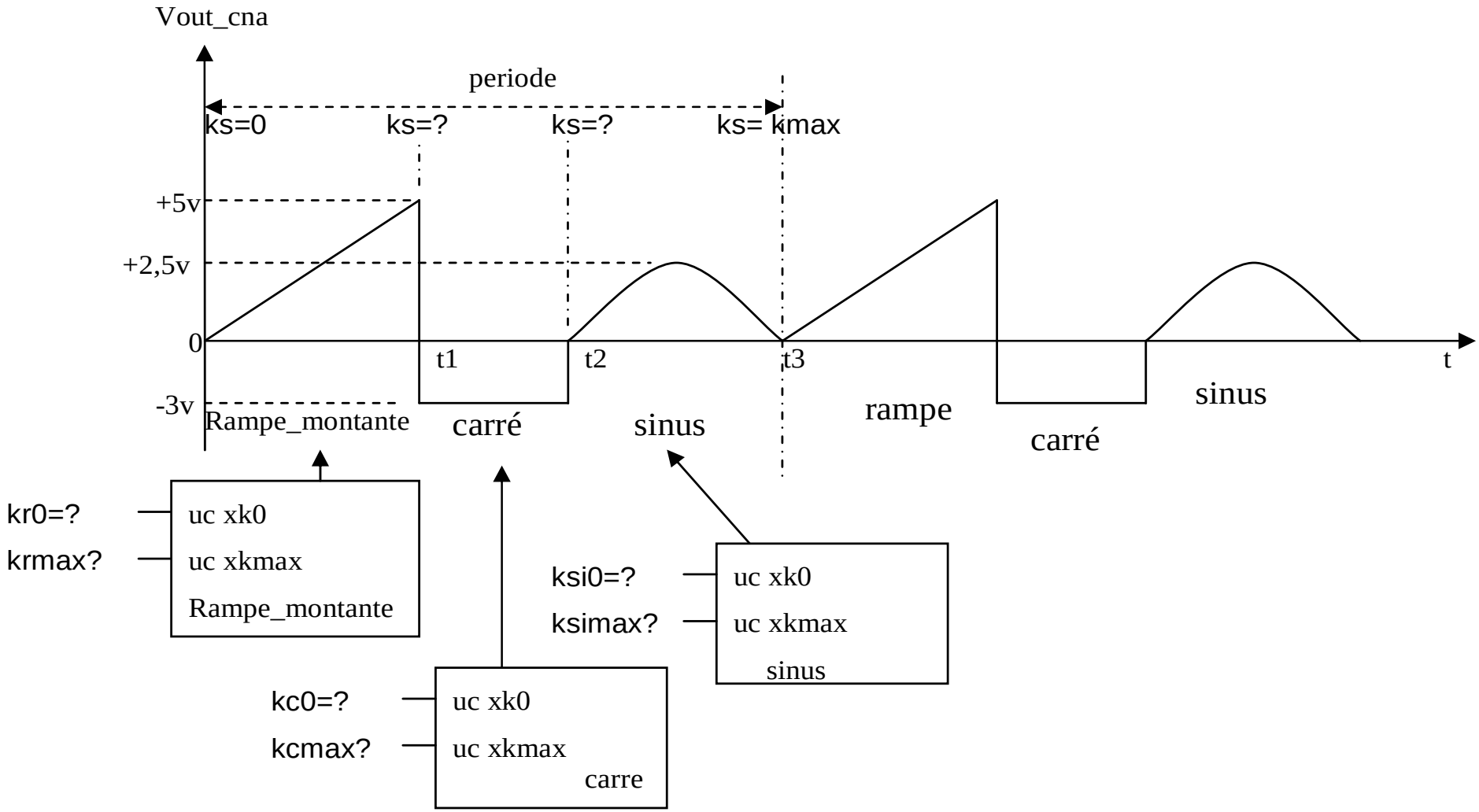
Et dans l'organigramme,

on remplace la ligne	**par**
$\text{nombre} = \frac{((\text{Asin}(w.(k.\Delta t))) + 5) * 4095)}{10}$	nombre = table_sinus[k]

d) Génération d'un signal complexe

Un signal complexe est constitué de segments de signaux élémentaires mis bout à bout.
La programmation consistera à utiliser les fonctions de génération de ces signaux élémentaires convenablement paramétrées.

Exemple



2- Convertisseur Analogique – numérique (CAN)

1- Introduction

Dans la nature, les tensions sont analogiques traduisent les grandeurs physiques qui sont le résultat des capteurs.

Par exemple :
- capteurs de température
- capteur de distance
- capteur de pression (artérielle, atmosphérique)
- capteur d'accélération
- capteur de poids (balance électronique)

En informatique embarquée, nous travaillons en numérique.

Pour passer du monde analogique au monde numérique, on utilise un composant (CAN = Convertisseur analogique – numérique))qui a pour but de transformer une tension analogique en un nombre.

Dans ce cours, nous allons considérer que ce composant est disponible.

Notre travail va consister à :

- trouver la relation qui existe entre la tension analogique à utiliser et le nombre que l'ordinateur va utiliser
- lire le nombre fourni par le composant.

On sait que le processeur lit des informations venant de l'extérieur par l'intermédiaire du port d'entrée

2- Définition

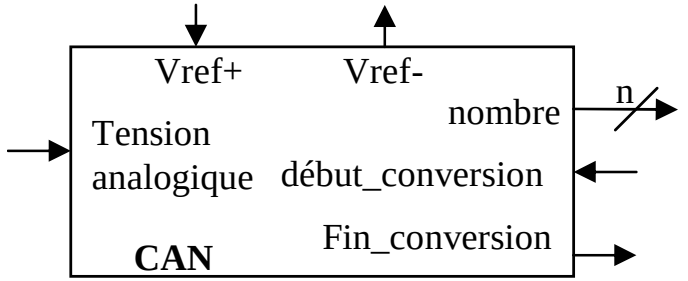
Un convertisseur analogique-numérique est un dispositif qui permet de transformer une tension analogique en un nombre binaire.

Dans ce cours, nous allons traiter de l'utilisation des CAN et leurs applications. Nous ne traiterons pas de la technologie de ces composants.

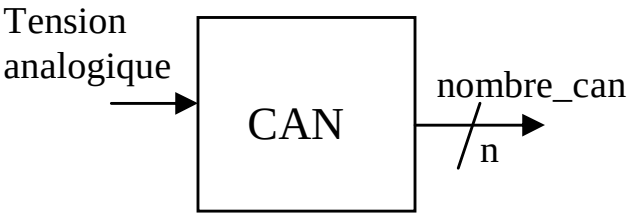
3- Schéma de principe

La tension analogique est convertie en un nombre binaire.
Cette conversion prend du temps. Il faut donc s'assurer que la conversion est terminée.
Pour cela,, on ajoute deux signaux au CAN:

- le début de la conversion
- la fin de conversion qui indique la conversion est terminée.



Ou simplement



4- Caractéristiques

Un CAN est caractérisé par

- n, le nombre de bits du nombre binaire obtenu
- la dynamique : c'est la plage de la tension analogique de sortie.

  Cette plage est définie par la différence entre deux tensions de référence Vref+ et Vref-

  Dynamique = Vref+ - Vref-
- la résolution = Dynamique/$2^n - 1$
- la linéarité

  Le CAN est linéaire donc fonction de transfert et une droite de la forme y = ax + b

  y est le nombre et x est la tension
- la fonction de transfert

  Vref- converti en 0

  Vref+ converti en $2^n - 1$

  Puisque le CAN respecte le principe de linéarité, la relation entrée-sortie est de la forme

  nombre = a * tension_analogique + b

  Vref- converti en 0 $\Rightarrow$ $0 = a^* Vref- + b$

  Vref+ converti en $2^n - 1$ $\Rightarrow$ $2^n - 1 = a^* Vref+ + b$

  on en déduit a et b avec $a = \frac{2^n - 1}{Vref+ - Vref-}$ $b = -a * Vref- = -\frac{2^n - 1}{Vref+ - Vref-} * Vref-$
- le temps de conversion

Application aux CAN utilisés en TD – TP

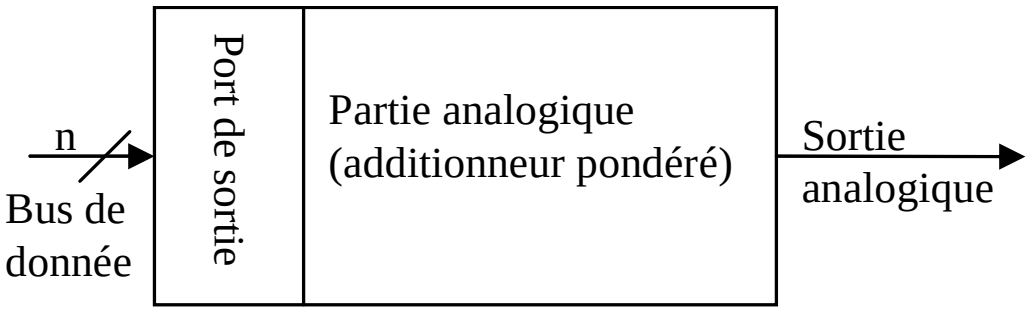
Nous disposons de 4 CAN sur la carte à base du microcontrôleur C167. Les 4 CAN sont intégrés dans le C167.
Caractéristiques de chacun des 4 CAN

- n = 10. CAN de 10bits
- Vref+ = +5v Vref- = -5v Dynamique = 5v - -5v = 10v
- résolution = 10v/1023
- Fonction de transfert : nombre = $\frac{1023 * \text{tension\_analogique}}{10} + \frac{1023}{2} = \frac{1023 * (\text{tension\_analogique} + 5)}{10}$
- le temps de conversion est environ de 10µs

5- Programmation

Nous travaillons avec le microcontrôleur C167.
Les CAN sont intégrés dans le C167.
Il y a 16 CAN intégrés mais nous n'en utiliserons que 4.
Les CAN sont multiplexés c'est-à-dire que, dans la pratique, l'on peut lancer la conversion sur un seul CAN à la fois.

Vue entrée-sortie d'un CNA



On dispose

- du registre ADCON pour :
  - ☞ choisir le CAN à lancer
  - ☞ lancer la conversion
  - ☞ attendre la fin de conversion
  - ☞ programmer le temps de conversion
- du bit ADCIR pour détecter la fin de la conversion
- du registre ADDAT pour lire le résultat de la conversion

Description des registres

Registre ADCON : registre de contrôle du bloc de conversion analogique-numérique

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Temps de conversion								**ADST**		**ADM**		**ADCH**			
0	0	0	0	0	0	0	0		0						

ADCON et le bit ADST sont connus par le compilateur
ADM et ADCH ne sont pas connus. Il faut donc trouver un mécanisme pour les mettre à jour.

ADCH permet de choisir le CAN à lancer parmi les 16. Pour cela les bits 3 et 2 seront fixés à 00

ADM permet de choisir le mode gestion des CAN. Nous allons le fixer à 00 pour indiquer que l'on ne fait qu'une conversion à la fois.
ADST permet de lancer une conversion.

Temps de conversion programmation du temps de conversion. 0000 permet d'avoir un temps de conversion d'environ 10µs

Tous les autres bits à 0 ne sont pas utiles pour le mode de fonctionnement choisi.

Conclusion

Nous aurons à manipuler uniquement ADCH et ADST.

Registre ADDAT : registre de résultat de la conversion

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Numéro du CAN converti						Résultat de la conversion									

ADDAT est connu par le compilateur

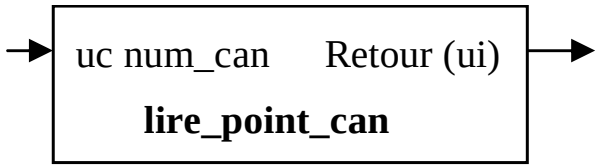
Bit ADCIR

Le bit ADCIR est connu par le compilateur.
Il est mis à 1 à la fin d'une conversion.
Il est de la responsabilité du programmeur de le mettre à 0 avant une nouvelle conversion.

Fonction d'accès aux CAN

Nous allons écrire la fonction qui permet d'accéder aux différents CNA

```
unsigned int  lire_point_can(unsigned char num_can)
 {
    ADCON = (num_can & 0x0003);  // choix voie à convertir
    ADCIR = 0;                   // préparer attente fin de conversion
    ADST   = 1;                  // lancer la conversion
    while (ADCIR==0);            // attendre la fin de conversion
    return (ADDAT & 0x03ff)      // lecture du résultat et retour
}
```



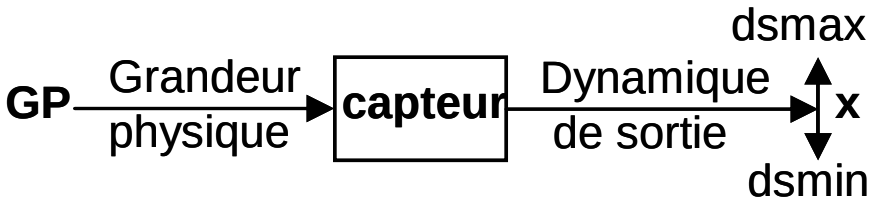
6- Applications des CAN

L'application des CAN concerne souvent le traitement du signal en particulier la gestion des capteurs.

Adaptation d'une grandeur physique issue d'un capteur

Connexion des grandeurs physiques à un CAN

Un capteur est caractérisé par une fonction de transfert x = g(GP) qui peut être linéaire ou non.
Elle décrit la relation entre la grandeur physique d'entrée, GP, et celle électrique de sortie, x (tension ou courant),



Dans la suite,

x sera une tension

GP sera en unités de la grandeur physique

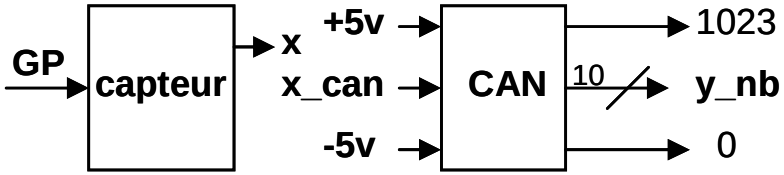
Ensuite, il suffit de calculer x= f(GP) et de remplacer dans les expressions du CAN pour faire le lien
y_nb = h(GP)

Y_nb est le nombre manipulé par le microcontrôleur

GP est la grandeur physique

Connexion d'un capteur à un CAN du TP

On admet que x est une tension



Pour obtenir une connexion cohérente,
il faut que x soit dans la dynamique du CAN
soit $-5v \leq x \leq +5v$

x = g(GP)
x_can = x
y_nb = f(x_can) = f(x)

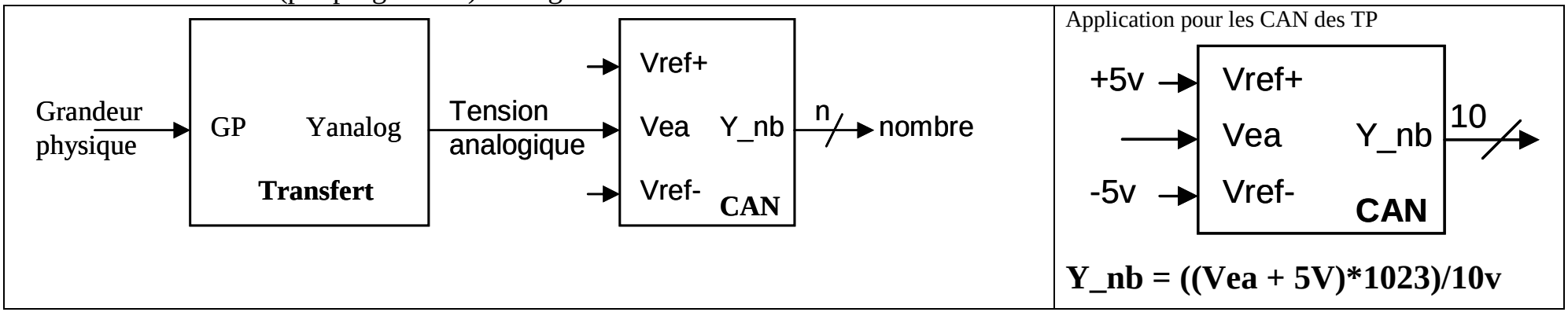
On peut alors calculer

y_nb = h(GP) = f(g(GP)) = (f o g) (GP)

Ces relations permettent d'effectuer tous les calculs sur l'adaptation d'une grandeur physique.

Remarque
Dans la pratique, un CAN est utilisé pour faire l'acquisition d'une grandeur physique.

Ensuite on doit traiter (par programme) cette grandeur. On obtient donc le schéma suivant :



On peut calculer Yanalogique = f(GP) en fonction du capteur ou du dispositif qui fournit la tension analogique.

On peut calculer Y_nb = g(Vea) on peut donc en déduire Y_nb = h(GP)

Donc si on mesure Y_nb par programme, on peut en déduire la grandeur physique avec ses unités :

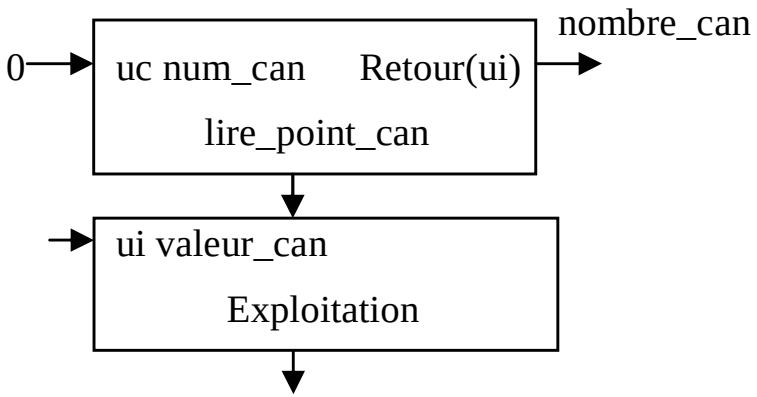
$GP = h^{-1}(Y\_nb)$

Exemples

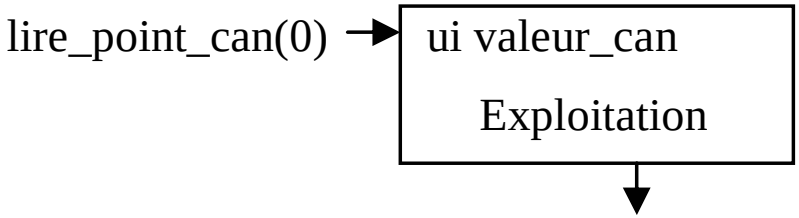
Dans les exemples qui suivent, la méthode est toujours la même. Seule la fonction f(GP) sera modifiée dans le bloc exploitation.

D'où l'organigramme ci-contre.

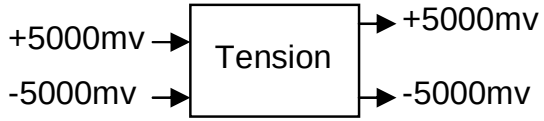
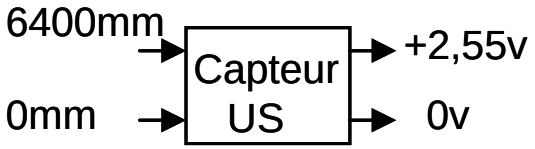
Dans cet exemple on a considéré can0.



Que l'on peut mettre sous la forme plus compacte



Calculer GP=k(y_nb) dans les cas suivants :

Voltmetre numérique **Transfert linéaire**	**Balance électronique** **Transfert linéaire**	**Télémètre Ultrason** **Transfert linéaire**
 **Dynamique :** **GP** **x** +5000mv → +5000mv -5000mv → -5000mv x= a * GP + b x= GP $y\_nb = \frac{1023*x}{10} + \frac{1023}{2}$ $= \frac{1023*GP}{10} + \frac{1023}{2}$ GP(mv)= k(y_nb) **Exploitation** printf("Tension = %4.2f mv", GP);	 **Dynamique :** 0 gramme → 0v 650 grammes → 5v x= a * GP + b $y\_nb = \frac{1023*x}{10} + \frac{1023}{2}$ GP(gr)= k(y_nb) **Exploitation** printf("Poids= %3.2f g", GP);	 **Dynamique :** 0 mm → 0v 6400 mm → 5v x= a * GP + b $y\_nb = \frac{1023*x}{10} + \frac{1023}{2}$ GP(gr)= k(y_nb) **Exploitation** printf("Distance=%4.2f mm", GP);