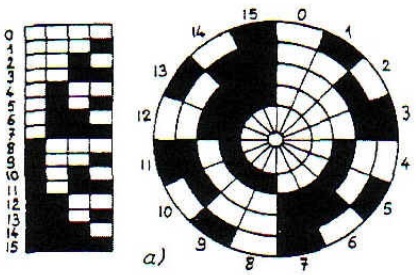
Les encodeurs numériques rotatifs

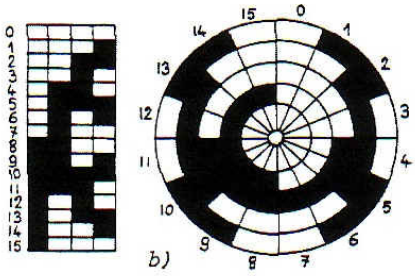
Il existe deux types de codeurs: le **codeur absolu** et le **codeur incrémental**.
Ils peuvent être de type électromécanique ou optique et peuvent se présenter en modèle **rotatif** ou **linéaire** (latte).

Dans le cas d'un **encodeur absolu**, chaque position est caractérisée par un code non volatile.
Cette position est connue dès la mise sous tension du système.

Le **codeur incrémental**, de conception plus simple, et donc moins onéreux, génère un code Gray sur deux bits.
Les modèles un peu plus sophistiqués, optiques par exemple, peuvent présenter une info supplémentaire : le « top zéro ». Il s'agit d'une impulsion qui est générée une seule fois par tour et qui détermine une position de référence permettant une réinitialisation à chaque tour

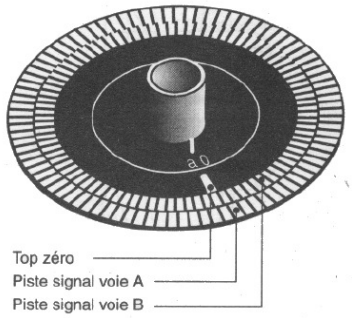


Binaire naturel



Binaire réfléchi

Codeur absolu

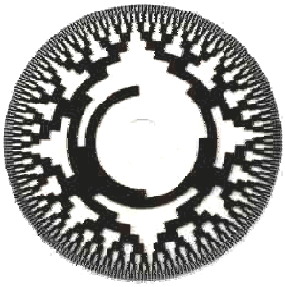


Codeur incrémental

Bref comparatif des deux systèmes :

CODEUR ABSOLU

Avantages



Il est insensible aux coupures du réseau : la position du mobile est détenue dans un code qui est envoyé en parallèle (parfois série) au système de traitement.

L'information de position est donc disponible dès la mise sous tension.

Si le système de traitement «saute» une information de position délivrée par le codeur, la position réelle du mobile ne sera pas perdue car elle restera valide à la lecture suivante.

Le positionnement peut se faire sans référence par un passage par zéro (Top zéro)

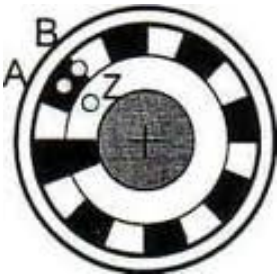
Inconvénients

Il est de conception électrique et mécanique plus complexe aussi son coût sera plus élevé qu'un codeur incrémental.

Les informations de position sont délivrées « en parallèle » ; son utilisation mobilisera donc un nombre important d'entrées du système de traitement.
A noter que les modèles plus sophistiqués transmettent leur code en mode série.

CODEUR INCREMENTAL

Avantages



Le codeur incrémental est de conception simple car son disque ne comporte que deux pistes, voire une troisième pour le « top zéro ».

Il est donc plus fiable et moins onéreux qu'un codeur absolu.

Inconvénients

Le système de gestion est sensible aux coupures du réseau : chaque coupure du courant peut faire perdre la position réelle du mobile à l'unité de traitement. Il faudra alors procéder à la réinitialisation du système automatisé.

Il est sensible aux parasites en ligne, un parasite peut être comptabilisé par le système de traitement comme une impulsion délivrée par le codeur.

Les fréquences des signaux A et B étant généralement élevées, il faudra vérifier que le système de traitement est assez rapide pour prendre en compte tous les incréments (impulsions) délivrés par le codeur.

Le non-comptage d'une impulsion induit une erreur de position qui ne peut être corrigée que par la lecture du « top zéro».

Nous allons nous intéresser au **codeur incrémental simple**.

CODEUR INCREMENTAL

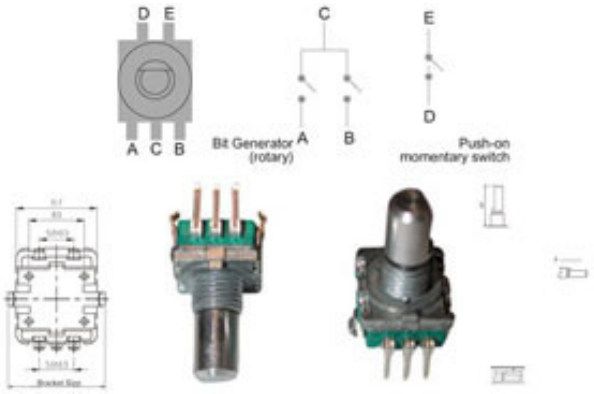
Générateur de Code Gray sur 2 bits

Pour nos applications, nous allons utiliser le **codeur incrémental simple**.

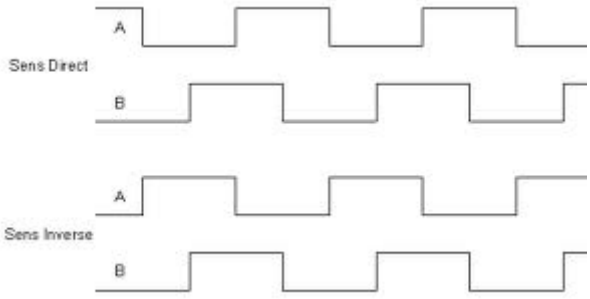
Outre une entrée «tension de référence» (C), il comporte juste deux sorties (A, B) et donc pas d'info « Top zéro ».

Certains modèles peuvent être munis d'un switch poussoir, fermé par enfoncement de l'axe, pour validation d'une commande, par exemple.

Ce type de codeur rotatif est donc un « générateur d'impulsions » qui délivre un code Gray sur ses deux sorties.
Le code Gray se caractérise par la modification d'un seul bit à la fois, à chaque changement d'état.



Codeur incrémental

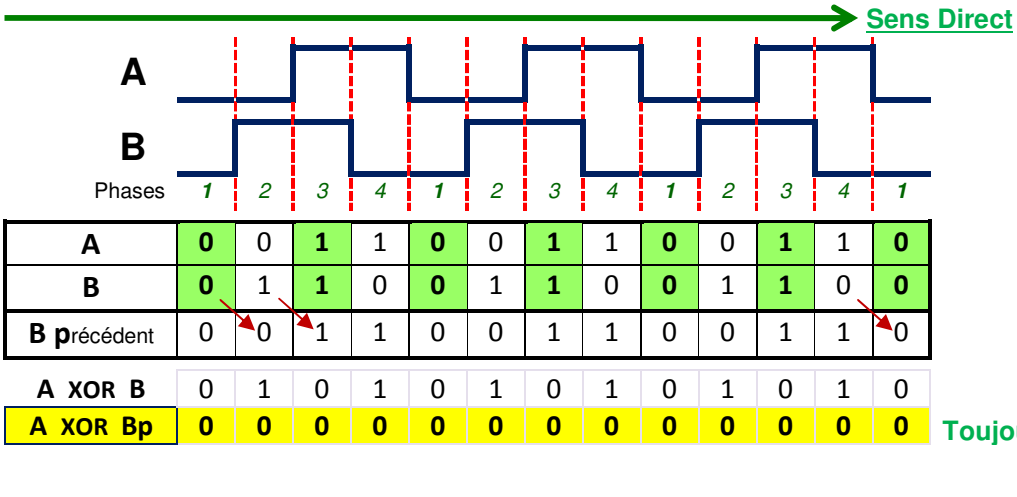


Code Gray délivré

Dans notre cas, les impulsions doivent être traitées pour en interpréter: **- le sens de rotation de l'axe**
- le nombre de crans qui ont été tournés

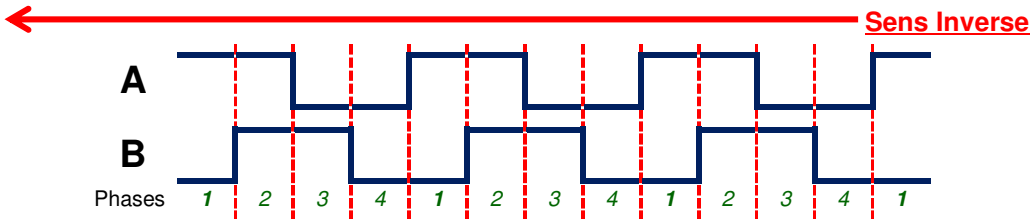
Déterminer le sens de rotation

Analysons préalablement le code généré par l'encodeur incrémental :



A	**0**	0	**1**	1	**0**	0	**1**	1	**0**	0	**1**	1	**0**
B	**0**	1	**1**	0	**0**	1	**1**	0	**0**	1	**1**	0	**0**
B précédent	0	0	1	1	0	0	1	1	0	0	1	1	0
A XOR B	0	1	0	1	0	1	0	1	0	1	0	1	0
A XOR Bp	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Toujours «0»



A	1	**1**	0	**0**	1	**1**	0	**0**	1	**1**	0	**0**	1
B	0	**1**	1	**0**	0	**1**	1	**0**	0	**1**	1	**0**	0
B précédent	0	0	1	1	0	0	1	1	0	0	1	1	0
A XOR B	1	0	1	0	1	0	1	0	1	0	1	0	1
A XOR Bp	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Toujours «1»

1) Traitement des impulsions par électronique

Sens de rotation

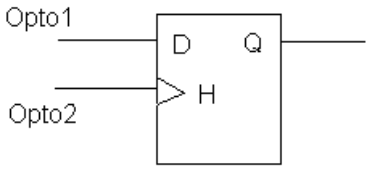
Utilisation d'une bascule D

Le déphasage de 90° électrique des signaux A et B permet de déterminer le sens de rotation :
- Dans un sens pendant le front montant du signal A, le signal B est à zéro.
- Dans l'autre sens pendant le front montant du signal A, le signal B est à un.

Si on utilise une bascule D (74HC74) en branchant le signal A sur l'entrée CK et le signal B sur la patte D, l'état de la sortie Q nous indiquera le sens de rotation de l'axe.

D	H	Q n+1
0	↑	0
1	↑	1
x	0	Qn
x	1	Qn

Bascule D



Q recopie D à chaque front montant sur H.

Les pins /CLR et /PR de la bascule 74HC74 Seront forcés à l'état haut.

2) Lecture et traitement des états stabilisés par µC

A) Sens de rotation

Suivant les tables et les graphiques édités plus haut, nous constatons qu'il est possible de déterminer le sens de rotation simplement en effectuant une opération Logique XOR sur le nouvel état de A et l'état de B précédent.

Pour différencier les nouveaux états établis des précédents, nous allons identifier les états **précédents** de A et B avec un préfixe «p» et les **nouveaux** états avec l'indice «n».

Nous allons écrire les 4 états possibles dans l'ordre où ils se présentent chronologiquement :

Sens DIRECT

	Précédent		devient		Nouveau	
	Bp	Ap	➔		nB	**nA**
4	**0**	1		1	0	**0**
1	**0**	0		2	1	**0**
2	**1**	0		3	1	**1**
3	**1**	1		4	0	**1**

Sens INVERSE

	Précédent		devient		Nouveau	
	Bp	Ap	➔		nB	**nA**
4	**0**	0		1	0	**1**
1	**0**	1		2	1	**1**
2	**1**	1		3	1	**0**
3	**1**	0		4	0	**0**

Pour déterminer le sens de rotation, nous constatons qu'il sera nécessaire de mémoriser l'état précédent de B (Bp) et effectuer une opération XOR avec la nouvelle valeur de A (nA) après modification d'un seul cran.

Si **nA XOR Bp = 0** alors **sens Direct**

Si **nA XOR Bp = 1** alors **sens Inverse**

Mais ces formules ne sont valables que pour une rotation que d'un seul cran !

En effet, **si un pas est « sauté »**, les comparaisons des anciens états et des nouveaux ne seront plus «synchronisés» et les formules ne seront donc plus valables !

Il faut donc espérer que le µC bouclera sa boucle infinie assez rapidement pour savoir **traiter distinctement chaque cran, un par un,** en lui laissant le temps de bien remémoriser chaque valeur de B (Bp).

L'idéal serait d'utiliser une «Interruption» dès que l'encodeur est activé… mais le Stamp ne gère pas les interruptions.

Comptage des crans

ATTENTION:
Avec certains petits codeurs, un cran que l'on ressent en tournant la molette, provoque en fait la génération de 4 cycles consécutifs !
On peut s'en rendre compte en plaçant un multimètre entre une sortie et la broche de référence: lorsqu'on tourne d'un cran, le résultat de mesure ne change pas !
En fait, il y a bien eu passage par 4 cycles différents mais très rapides qu'un Microcontrôleur aura cependant pu lire.
Il faut en tenir compte dans la conception du programme, en divisant les mesures effectives par 4 !

Ce qui va être expliqué ci-dessous considère qu'un cran = 1 seul changement d'état (et non 4).

Nous allons devoir utiliser une variable « compteur de crans » et la mettre à jour pour chaque cran tourné dans un sens ou l'autre.

Au vu de ce que nous avons vu plus haut nous disposons des données suivantes:

Sens de rotation	nA XOR Bp=	Action Compteur
Direct	0	+1
Inverse	1	-1

Pour une rotation dans le sens direct, nous devrons incrémenter le compteur de crans et pour le sens inverse, ce compteur devra être décrémenté.

Si nous appelons «I» l'incrément du compteur et «x» le résultat de (nA xor Bp), nous avons les relations suivantes:

Pour x=0 ➔ I = +1 Ecart de 1 unité entre x et I
Pour x=1 ➔ I = -1 Ecart de 2 unités entre x et I

Nous allons raisonner sur le principe des **régressions mathématiques**.

Une régression est l'estimation de l'équation de la relation existant entre plusieurs variables, ici x et I.

Considérant ici x la mesure des causes, et I celle des effets, la liaison entre I et x s'écrit suivant la relation fonctionnelle suivante : I = fc (x).
Traduisez: à une valeur donnée de x correspond une valeur bien déterminée de I.

Cette relation peut être linéaire ou non.
Dans le cas d'un modèle linéaire, l'équation de la régression est: Modèle linéaire: **I=a+bx**.

Et oui, la fameuse équation d'une ligne droite dont les paramètres sont : a (ordonnée à l'origine) et b (pente).

Dans notre cas, nous voyons que pour une variation d'une unité de x, I peut varier de 2 unités dans le sens inverse.

Un facteur b de -2 est donc à introduire dans notre fonction avant d'évaluer le facteur «a». Ainsi :

Pour x=0 ➔ I = x * (-2) = 0 (Le résultat final doit être +1)
Pour x=1 ➔ I = x * (-2) = -2 (Le résultat final doit être -1)

Nous voyons immédiatement que le facteur de transposition «a» est alors égal à +1. En effet :

Pour x=0 ➔ I = (x * (-2)) +1 = +1 (Le résultat final de +1 est obtenu)
Pour x=1 ➔ I = (x * (-2)) +1 = -1 (Le résultat final de -1 est obtenu)

Nous pouvons donc finaliser ce raisonnement en écrivant :

Compteur = Compteur + (((nA.XOR.Bp)*(-2))+1)

Cette formule n'est à appliquer, cran après cran, **que lorsque l'encodeur a effectivement été actionné** !

Puisque le Stamp ne gère pas les Interruptions, un mouvement sera détecté en scrutant les modifications d'état des sorties A et B de l'encodeur.

Algorithme

Structuré :

```
IF (nA<>Ap.or.nB<>Bp)                                  ‘ Action détectée sur encodeur
  LET ComptOld = Compteur                              ' Mémorisation du Compteur
  LET Compteur = Compteur + (((nA.xor.Bp)*(-2))+1)     ' Incrémente/Décrémente Compteur
  LET NbreCrans = Compteur - ComptOld                  ‘ Nombre de crans
  LET Ap = nA                                          ‘ Remplacement mémo de Ap par An
  LET Bp = nB                                          ‘ remplacement mémo de Bp par Bn
Endif
```