TER_documentation

> nc = non-connectée
>
> bits inutiles dans les cas cités

`snake.c`

Flags

les différents flags se trouvent dans la variable flag (`uint8_t`)

1111 1	1	1	1
nc	flag_stp	flag_pm	flag_init

flag_init :

> ce flag permet d'initialiser le vecteur de positions une fois au début

flag_pm :

> flag qui permet de faire apparaitre une pomme sur l'ecran

flag_sp :

> flag STOP qui permet de reset toute les variables du snake

dir

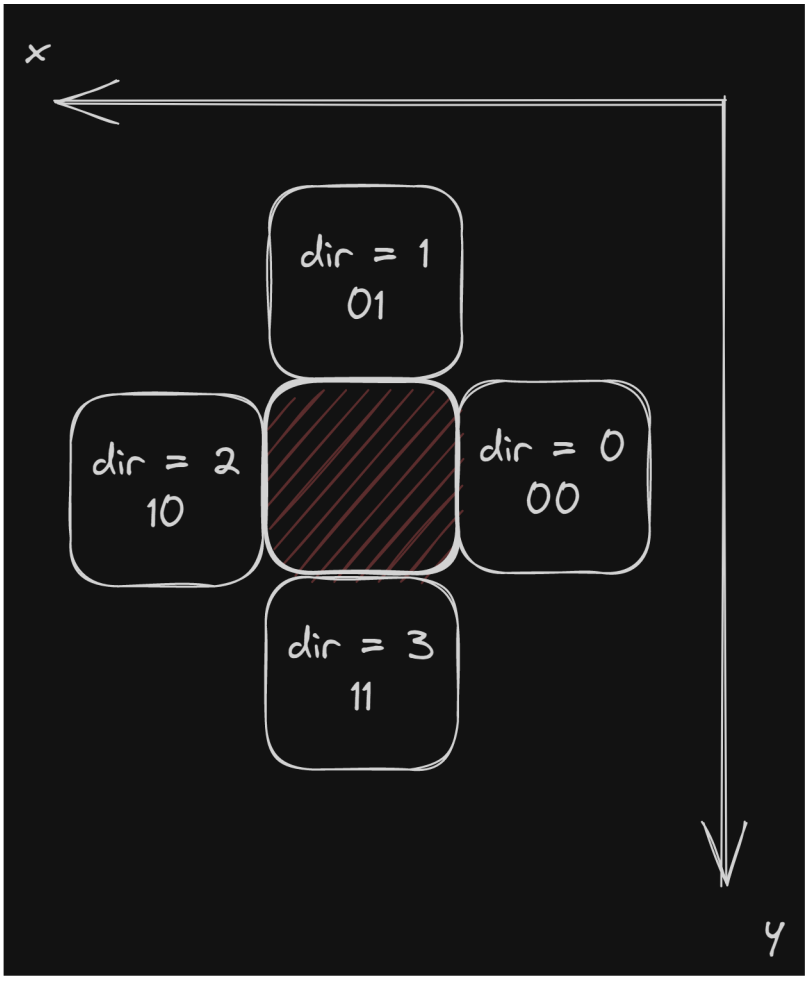
dir : sur 8 bits permet d'avoir la direction du snake
`0000 00XX` les deux derniers bits sont les bits utiles

0000 00	X	X
nc	=0 > - =1 > +	=0 > H =1 > V

Le dernier bit permet de choisir si la direction est vertical ou horizontal
L'avant dernier bit permet de savoir si on bouge de manière positive ou negative.

Exemples de valeurs :

- 0 `0000 0000` : Droite
- 1 `0000 0001` : Haut
- 2 `0000 0010` : Gauche
- 3 `0000 0011` : Bas



pos

les positions sont stokées sur un octet :

0000	0000
Ypos	Xpos

on a une matrice de 9/9
val max = `1001 1001` -> pos [8,8]

val min = `0001 0001` -> pos [0,0]

On a une bascule par vide et par plein :

- Lorsque Ypos ou Xpos= `0000` (0) => la tete depasse de l'ecran, le jeu le fait donc basculé de l'autre coté en `1001` (9)
- De meme, lorsque Ypos ou Xpos = `1010` (10) => la position est réacualisé a `0001` (1)

Incrémentation

4/4 dir gauche -> 4/5

$$\mathrm{Xpos} += (1 - \mathrm{dir}_0) \times (-1 + 2.\mathrm{dir}_1)$$
$$\mathrm{Ypos} += \mathrm{dir}_0 \times (-1 + 2.\mathrm{dir}_1)$$

test

(4,4) dir = 0b10

- $X = 4 + (1 - 0).(-1 + 2) = 5$
- $Y = 4 + (0).(-1 + 2) = 4$
  (2,2) dir =0b01
- $x = 2 + (0).(-1 + 0) = 2$
- $y = 2 + (1).(-1 + 0) = 1$

`terlib.c`

ReadInput

Lis les inputs A,B, Up, Down, Left and Right et les stock dans un entier `uint8_t` tel que

00	0	0	0	0	0	0
nc	UP	DOWN	RIGHT	LEFT	B	A

> ⚠ **Selection du jeu se fait sur un tick => tester avec des vrai plaquettes si la connectique se fait bien**
>
> - Branchement sur les pins doit etre simultané