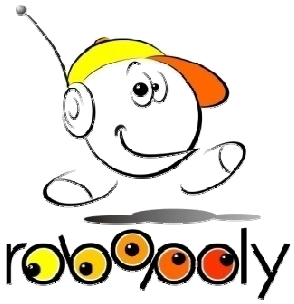
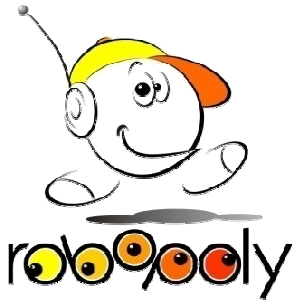


Démon Robopoly – Programmation avancée

REGISTRES ET MASQUAGE

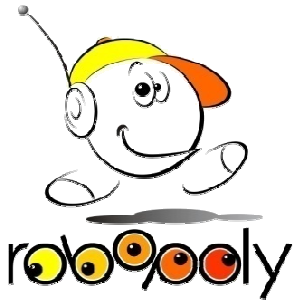
Qu’est-ce que le masquage



Commençons par un exemple, comment récupérer les bits 3..6 de ce nombre binaire ?

0b01001101

Grâce au masquage




```
(0b01001101 & 0b01111000) >> 3
```

Qu’est-ce que ce charabia, me direz-vous.

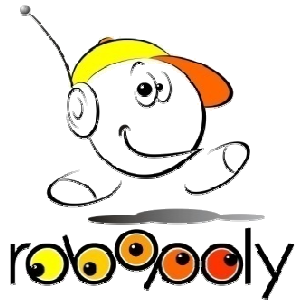
L'opérande &

- Le « et » logique : il faut que les deux bits valent 1

a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

```
  0b01001101
& 0b01111000
------------
  0b01001000
```

L'opérande |

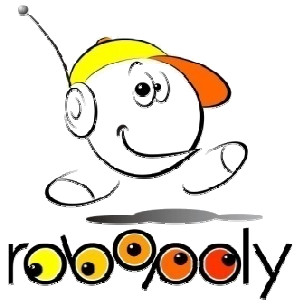


- Le « ou » logique : il faut qu'un des deux bits valle 1

a	b	a|b
0	0	0
0	1	1
1	0	1
1	1	1

```
  0b01001101
| 0b01111000
------------
  0b01111101
```

L'opérande ^

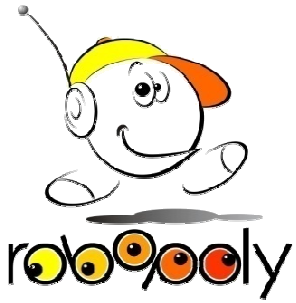


- Le « ou exclusif » logique : il faut qu'un seul des deux bits valle 1

a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

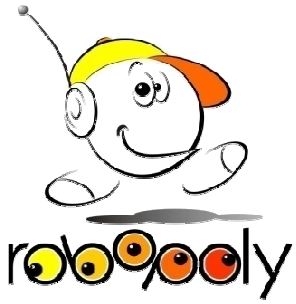
```
  0b001001101
^ 0b001111000
-------------
  0b000110101
```

Résumé



- Le & force les bits à 0
- Le | force les bits à 1
- Le ^ inverse les bits

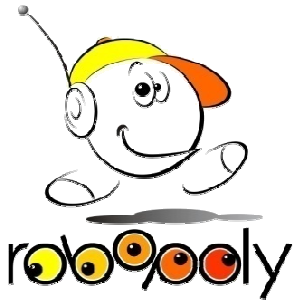
Le décalage de bit



- L'opérande >> et << décalent respectivement les bits à droite et à gauche.

```
0b01001000 >> 3 = 0b00001001
```

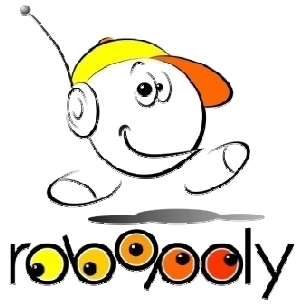
L’inversion de bit



- l’opérande ~ inverse le nombre bit à bit

```
~0b01001101 =
 0b10110010
```

Reprenons l'exemple

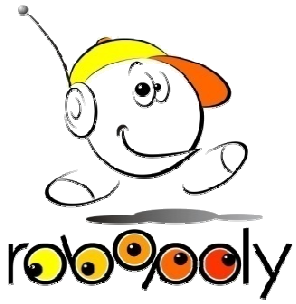



```
(0b01001101 & 0b01111000) >> 3
```

On masque d'abord puis on décale les bits.

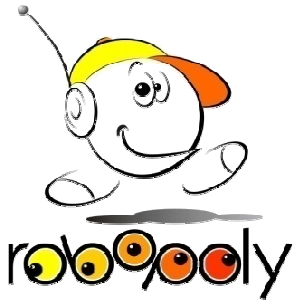
Donc la réponse est bien ce que nous cherchions

```
= 0b00001001
```



LES REGISTRES

Qu'est-ce qu'un registre

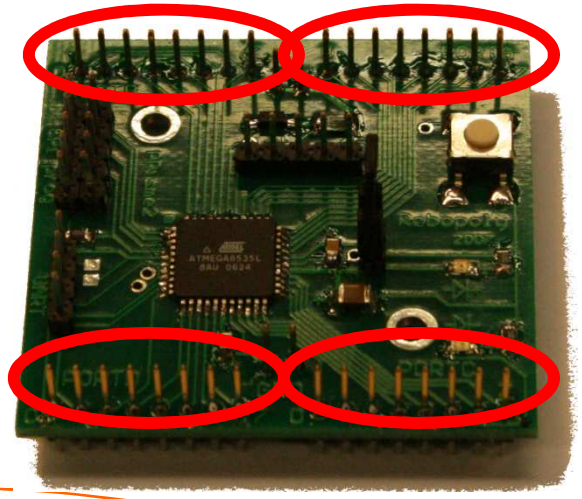


- C'est un endroit physique sur le microcontrôleur dans le quel on peut stocker des valeurs

- Sur le Prisme (ATMega8535) se sont des registres 8 bits (8 informations (0/1))
- Il y a en tout 95 registres

Les entrées / sorties

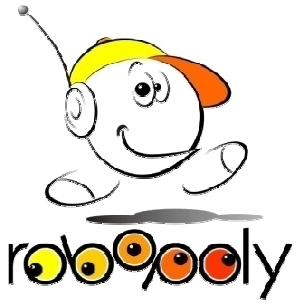
Nom	Description
DDR	Indique si le port est en entrée (0) ou en sortie (1)
PORT	Donne la valeur de sortie
PIN	Donne la valeur d'entrée



```
/*Le PORTA est en
entrée et on lit sa
valeur*/
DDRA = 0b00000000;
valeur = PINA;

/*Le PORTC est en
sortie et on sort une
valeur*/
DDRC = 0b11111111;
PORTC = 0b01010000;
```

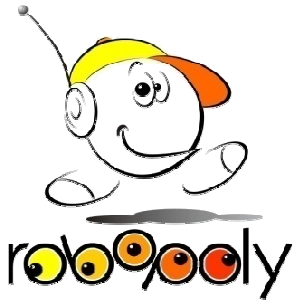
Exemple (1)



Comment fait-on pour copier la valeur des maxi-boutons sur les maxi-leds ?

```
int main( void )
{
   DDRA = 0b00000000;    //PORTA en entrée
   DDRB = 0b11111111;    //PORTB en sortie
   while(1)
      PORTB = PINA;      //Copie
}
```

la fonction digitalRead



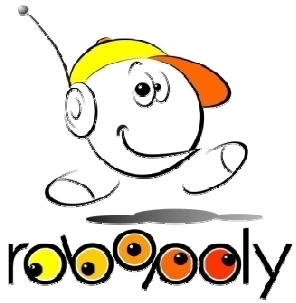
- si on appelle la fonction suivante :

```
digitalRead(A,5);
```

- Le microcontrôleur met le DDRA5 en entrée et lit la valeur du PINA5 avec ce code

```
DDRA &= ~(1<<bit);  // bit = 5
return (PINA >> bit) & 1;
```

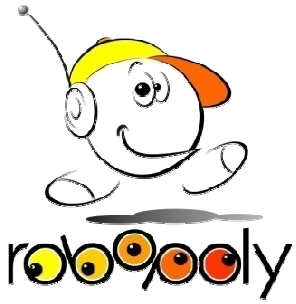
Explication (1)



Opération	Explication
`1<<5`	`0b00000001 << 5 = 0b00100000`
`~(1<<5)`	`~0b00100000 = 0b11011111`

`DDRA &= ~(1<<5)`
 	
`DDRA = DDRA & ~(1<<5)`
	`0baaaaaaaa`
	`& 0b11011111`
	`0baa0aaaaa`

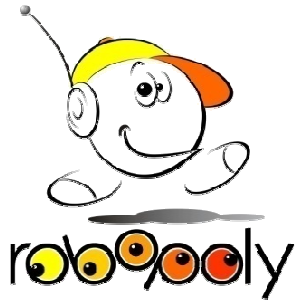
Explication (2)



Opération	Explication
`PINA >> 5`	`0baaAaaaaa >> 5 = 0b00000aaA`

`(PINA >> 5) & 1`	`0b00000aaA`
	`& 0b00000001`
	`0b0000000A`

la fonction digitalWrite



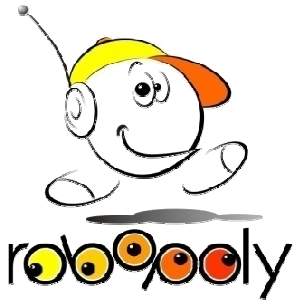
- si on appelle la fonction suivante :

```
digitalWrite(C,2,1);
```

- Le microcontrôleur met le DDRC2 en sorti et écrit la valeur 1 sur le PINC2 avec ce code

```
DDRA |= (1<<bit); // bit = 2; value = 1<<bit
PORTA = (PORTA & (~(1<<bit))) + value;
```

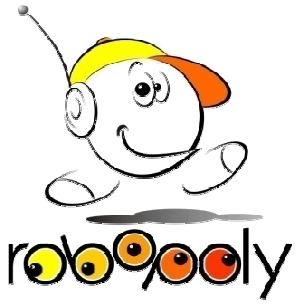
Explication (1)



Action	Explication
`1<<2`	`0b00000001 << 2 = 0b00000100`

`DDRA |= (1<<2)`	`0baaaaaaaa`
	`| 0b00000100`
	`0baaaaa1aa`

Explication (2)



Action	Explication
`~(1<<2)`	`~0b00000100 = 0b11111011`

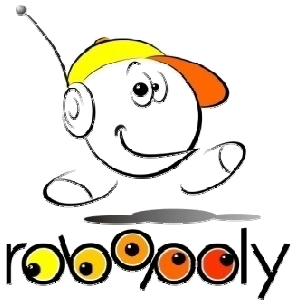
`PORTA & (~(1<<2))`	`0baaaaaaaa`
	`& 0b11111011`
	`0baaaaa0aa`

`value = value << 2`	`0b0000000V << 2 =` `0b00000V00`

`(PORTA & (~(1<<2))) +`
`value`
	`0baaaaa0aa`
	`+ 0b00000V00`
	`0baaaaaVaa`

```
avec value = 1 : PORTA = 0baaaaaa1aa
avec value = 0 : PORTA = 0baaaaaa0aa
```

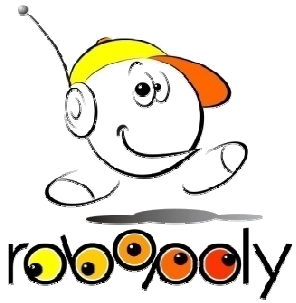
Les registres de configuration



- La configuration du microcontrôleur, on utilise aussi les registres.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	10
0x3E (0x5E)	SPH	–	–	–	–	–	–	SP9	SP8	12
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
0x3C (0x5C)	OCR0	Timer/Counter0 Output Compare Register								85
0x3B (0x5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	49, 69
0x3A (0x5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	70
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	85, 115, 133
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	86, 116, 134
0x37 (0x57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	228
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	181

Comment lire un datasheet



1. Lire les explications qui décrive le fonctionnement du module
2. Puis configurer le uC à l'aide des registres de configuration qui sont décrit à la fin de la section

8-bit Timer/Counter
Register Description

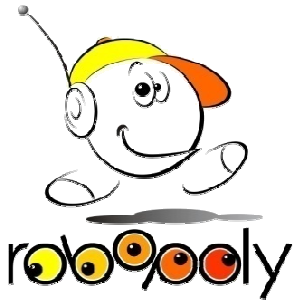
Timer/Counter Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0: Force Output Compare**

The FOC0 bit is only active when the WGM00 bit specifies a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0 is written when operating in PWM mode. When writing a logical one to the FOC0 bit, an

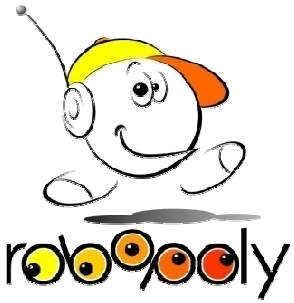
Exemple de configuration de la fonction analogReadPortA




```
//Configure le prescaler, l'interruption et
  active l'ADC
ADCSRA = 0x87;

// Ajuste le résultat dans un seul registre
  et sélectionne l'entrée à convertir
ADMUX = 0x20 + bit;

// Démarre la conversion
ADCSRA |= (1<<ADSC);
```



DES QUESTIONS ?