**CHAPITRE 5 :**

**Les interruptions exploitables sur ATMega328 (carte Arduino UNO)**

**Interruption (IT)** est la suspension du programme en cours pour exécuter une *routine d'interruption* (**Interruption Service Routine ISR)**.

ATmega328 est équipé d’un complément puissant et flexible de 26 sources d’interruption. Deux des interruptions proviennent de sources d'interruption externes, tandis que les 24 autres interruptions prennent en charge le fonctionnement des périphériques embarqués sur le microcontrôleur.

Les sources d'interruption ATmega328 sont illustrées à la figure 1.

Le RESET a la priorité la plus élevée, suivi des broches externe INT0 (pin4) et INT1 (pin5). Les sources d'interruption restantes sont internes à l'ATmega328.

Quand une interruption se produit, le microcontrôleur termine l’instruction en cours, stocke l’adresse de la prochaine instruction sur la pile et commence à exécuter les instructions dans le répertoire désigné le programme de service d’interruption (ISR) correspondant à la source d’interruption particulière.

Le système d’interruption empêche d’autres interruptions en cours. L'exécution du L’ISR est exécuté en chargeant l’adresse de début de la routine de service d’interruption spécifique à cet événement.

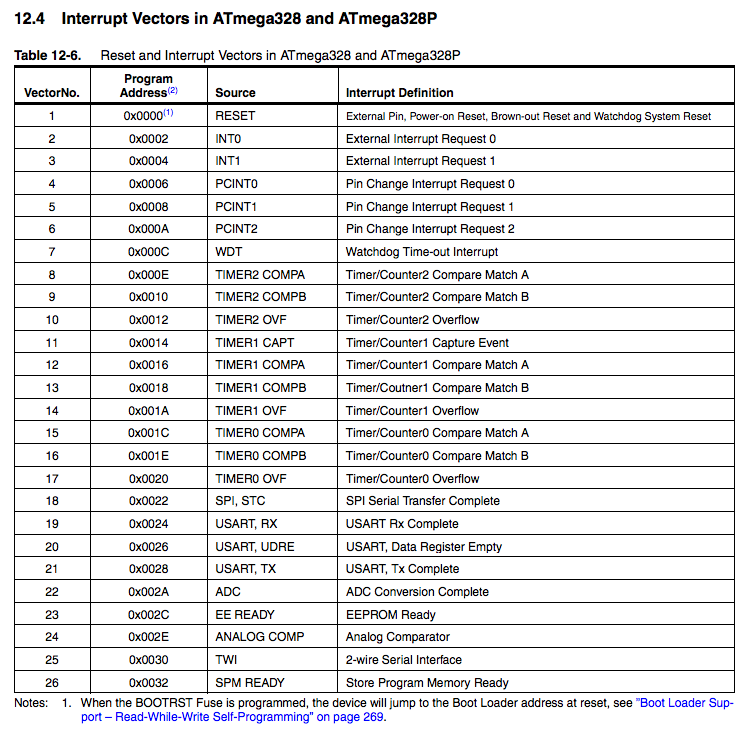


Figure 5.1 : Les sources d'interruption ATmega328.

**Programmation des interruptions :**

**Les interruptions externes :**

Les interruptions INT0 et INT1 peuvent être déclenchées avec un front descendant ou montant (falling/rising edge), un signal de niveau bas (low level) ou un changement d'etat quelconque (any logical change). Les paramètres spécifiques à chaque interruption sont fournis à la figure 5.2.

Il y a trois registres dans le processeur pour gérer les deux pins d'interruption INT0 (pin 4) and INT1 (pin 5).

Le registre EICRA "External Interrupt Control Register" ne contient que 2 fois 2 bits qui pour chaque pin codent le mode vu plus haut.

Le registre EIMSK "External Interrupt Mask Register" contient les 2 bits INT0, INT1 qui autorisent les interruptions 0 et 1.

Le registre EIFR "External Interrupt Flag Register" contient les 2 bits INTF0, INTF1qui sont activés par les interruptions 0 et 1.



**Figure 5.2:** Interrupt INT0 and INT1 Registers.

Après avoir déterminé les pins d’interruption Il faut encore savoir le nom de la fonction d’interruption **associée à une cause d'IT,(** associer au vecteur d'interruption: ISR(INT0\_vect) { nos instructions }) Figure 5.1 .

Exemple :

Chaque fois que le bouton poussoir fait passer le niveau de 1 a 0 sur l'entree INT0(PD2),la fonction d''interruption associee a INT0 est executee. Cette action a pour effet d'inverser l'etat de la LED et de revenir au programme principal. Il est important de comprendre que l'interruption ne dure que quelques microsecondes. En dehors de ces interruptions, le programme principal (fonction loop()) envoie chaque seconde la valeur de la variable cpt.

Exemple : exploitation IT Externe 0

// Exemple : exploitation IT Externe 0

**ISR(INT0\_vect)** *// ISR INT0 = traitement si interruption INT0*

**{**

**PORTB ^= 0x20;** *// bascule PORTB.5*

**}**

void setup(){ *// configuration pour exploiter IT INT0*

**cli();**

Serial.begin(9600);

DDRB |= 0x20; // PB5 en sortie

PORTB &= ~0x20; // PORTB.5 <-0 (sortie PB5 à 0) DDRD &= ~0x04; // PD2 en entrée

PORTD |= 0x04; // PORTD.2=1 <-> **activer pull-up EICRA = 0x02**; // IT si front sur INT0 (table 12-2)

**EIMSK |= 1;** // source INT0 activée

**sei();** // active les IT en général

}

int cpt=0; // variable globale (hors fonction)

void loop() { // ce traitement peut être interrompu pour INT0 Serial.println(cpt,DEC);

cpt++; delay(1000);

}

Le microcontrôleur ATmega328 accepte des interruptions possibles ***sur toute entrée TOR (Interruptions "Pin Change")***

Chaque front, montant ou descendant, conduit à une interruption Pin Change (PCINT0\_vect, PCINT1\_vect, PCINT2\_vect) permettent d'associer une fonction (ISR) à un changement d'état d'une entrée.

Donc les broches PCINT0 à PCINT23 sont configurables pour déclencher des interruptions suite à des changements d'état ("Pin Change") Les broches sont séparées en 3 sous-groupes, il y a une source d'interruption par sous-groupe, et pour chaque broche on peut activer ou non le système "Pin Change Interrupt"

PCINT0 – PCINT7 <-> broches PB0 à PB7 groupe lié à IT PCINT0\_vect

PCINT8 – PCINT15 <-> broches PC0 à PC7 groupe lié à IT PCINT1\_vect

PCINT16 – PCINT23<-> broches PD0 à PD7 groupe lié à IT PCINT2\_vect

Les registres **PCMSK0, PCMSK1 et PCMSK2** contrôlent, pour chacun de ces groupes (donc pour chaque port B, C D), quelle(s) broche(s) peut(vent) conduire (ou non) à une interruption de type "pin change".

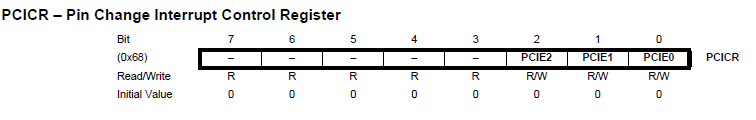
**Activation des interruptions PCINT0 à PCINT23 si bit SREG.7=1 et mise à 1 de PCIEx**

**Registre PCICR** : activation des IT Pin Change pour un groupe

PCICR.0 : activation des IT Pin Change pour les broches du port B (PB0 a PB7)

PCICR.1 : activation des IT Pin Change pour les broches du port C (PC0 a PC6)

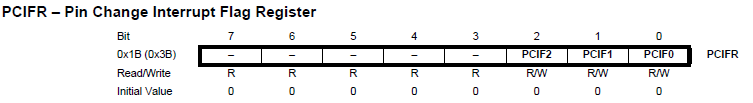
PCICR.2 : activation des IT Pin Change pour les broches du port D (PD0 a PD7)



**Activation à l'intérieur d'un groupe** : le registre **PCMSKx** détermine quelles broches du groupe sont prises en compte pour l'interruption "pin change"



**Flags internes** pour les IT "Pin Change"





**Exemple : mise en oeuvre IT Pin Change sur Port B et Port D**

**// Config pour IT**

**void setup()**

**{**

cli();

PCICR |= 0x05; DDRB&=~0x03; DDRD&=~0x0A; PCMSK0=0x03; PCMSK2=0xA0;

sei();

**}**

**Pin**

**Change**

***//***

***//***

***//***

***//***

***//***

***//***

***//***

***aucune IT possible (SREG.7<-0)***

***Pin***

***PB1 PD7***

***Pin Pin***

***Change***

***et PB0 et PD5 Change Change***

***actif sur port D***

***en entrée en entrée***

***pour broches PB0 pour broches PD7***

***et***

***port B***

***et***

***et***

***PB1***

***PD5***

***IT possibles SREG.7<-1***

ISR(PCINT0\_vect){ ... } // appelée si changement sur PB1 ou PB0

ISR(PCINT2\_vect){ ... } // appelée si changement sur PD7 ou PD5

**Programmation d'interruption interne :**

Pour les autre type d’interruption comme les timers on va les exploiter dans les prochaines sections.

**Les interruptions arduino :**

L'environnement de développement Arduino comporte quatre fonctions intégrées permettant de prendre en charge les interruptions externes INT0 et INT1.

1/ **interrupts()**. This function enables interrupts.

2/ **noInterrupts()**. This function disables interrupts.

3/ **attachInterrupt(interrupt, function, mode)**. This function links the interrupt to the appropriate

interrupt service routine :

• **interrupt.** Interrupt specifies the INT interrupt number: either 0 or 1.

• **function.** Function specifies the name of the interrupt service routine.

• **mode.** Mode specifies what activity on the interrupt pin will initiate the interrupt:**LOW** level

on pin, **CHANGE** in pin level, **RISING** edge, or **FALLING** edge.

4/ **detachInterrupt(interrupt)**. This function turns off the specified interrupt.

Exemple : exploitation IT Externe 0

void setup()

{

attachInterrupt(0, int0\_ISR, RISING);

}

void loop()

{

//wait for interrupts

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//int0\_ISR: interrupt service routine for INT0

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void int0\_ISR(void)

{

//Insert interrupt specific actions here.

}

Référencées :

1/ Fiche technique ATmega16 et ATmega32

2/ www.arduino.cc

3/ B. Cottenceau : Carte ARDUINO UNO Microcontrôleur ATMega328 - 2016-2017.