ARDUINO

Interrupções usando baixo nível

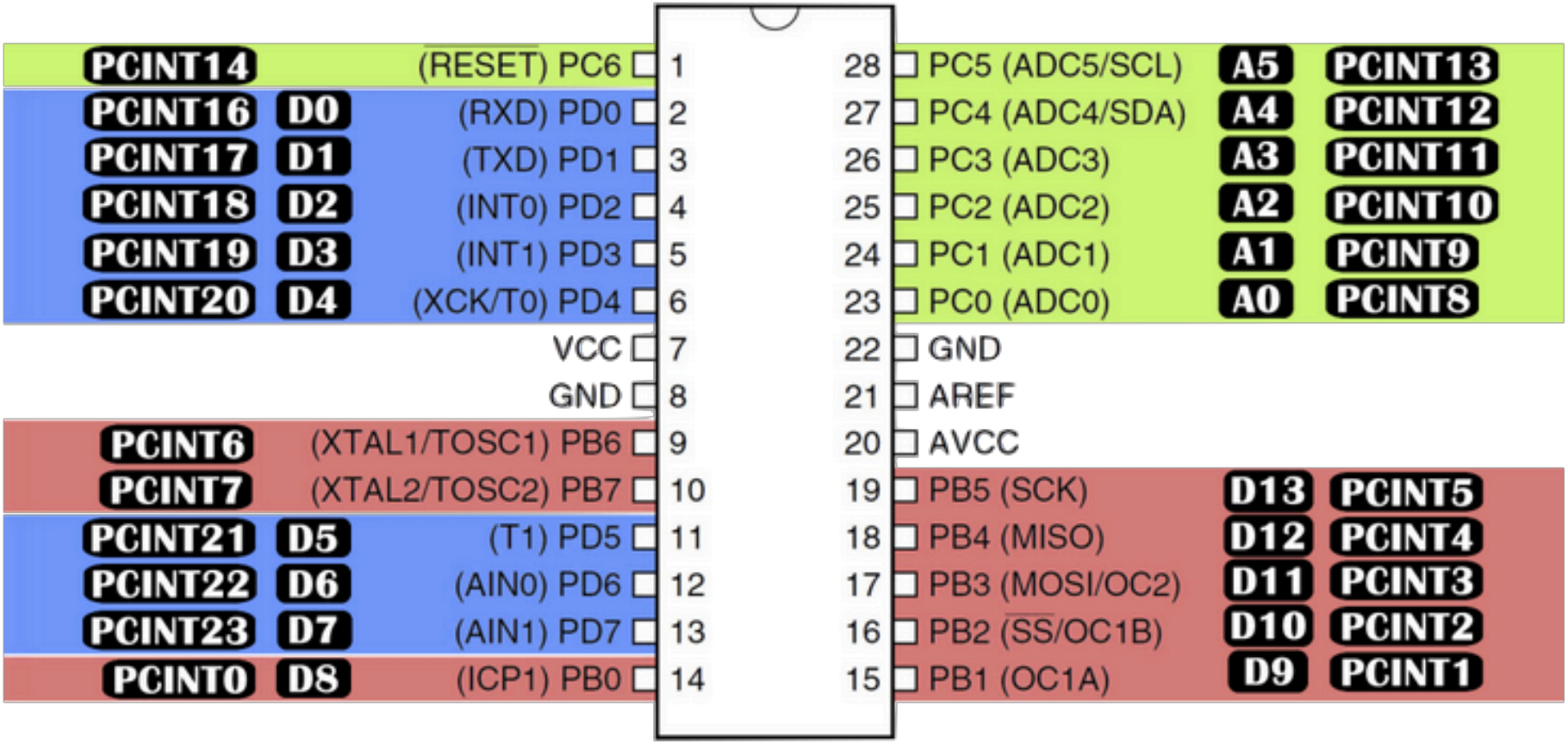
Interrupção de baixo nível (todos os pinos)

Pin Change Interrupt

- Tem casos em que **dois pinos de interrupção** não são suficientes
- Há outro tipo de interrupção que podemos usar em **todos** os pinos do Arduino: PCINT (**Pin Change Interrupt**)
- Este método não está embutido na plataforma Arduino, portanto, é preciso aprofundar na programação de mais baixo nível
- Vamos nos **limitar** às placas ATMega328, como o Arduino UNO e o Arduino Nano

Pin Change Interrupt

Note que há três portas:
PB (vermelho), PC (verde)
e PD (azul)



Note que PCINT abrange todos os pinos de D0 a D13 e A0 a A5

A interrupção só ocorrerá quando houver uma mudança no estado do pino escolhido

Pin Change Interrupt Control Register

- Existem apenas **três possíveis vetores de interrupção**, i.e., **só três ISRs** para todos esses pinos
- As ISRs são: **`PCINT0_vect`**, **`PCINT1_vect`** e **`PCINT2_vect`**
- Qualquer mudança de pino entre D0 a D7, D8 e D13 e A0 a A5, **só aciona uma interrupção**
- Caso dois pinos tenham seus estados mudados, **ambos compartilharão a mesma** rotina de interrupção

Pin Change Interrupt Control Register

- Devemos implementar a rotina de interrupção de forma que **esta seja capaz de identificar** em qual dos pinos ocorreu a interrupção
- Para ativar a PCINT em um pino específico precisamos manipular o registrador PCICR (***Pin Change Interrupt Control Register***)

Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	**PCIE2**	**PCIE1**	**PCIE0**
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Os últimos três bits desse registrador são **bits de controle** que servem para **habilitar um grupo PCINT**

Quando **PCIE0** (bit 0) está setado, os pinos **PCINT0 a PCINT7** são ativados, que são equivalentes aos pinos **D8 a D13**

Quando os bits PCIE1 ou PCIE2 são setados, os pinos PCINT8 a PCINT14 (A0 a A5) e PCINT16 a PCINT23 (D0 a D7) são ativados

Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	**PCIF2**	**PCIF1**	**PCIF0**
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Quando as interrupções são acionadas, haverá **bits de sinalização** (*flag bits*) correspondentes que serão setados (valor igual a 1)

Esses *flag bits* são encontrados no registro **PCIFR**

Quando o **PCIE0** está ativado e a interrupção é acionada, o bit **PCIF0** será setado

O mesmo é verdade para o resto dos *flag bits*

Pin Change Mask Register

- **O que fazer para habilitar apenas pinos específicos dentro de um grupo?**
- Isso é feito usando o registrador PCMSK (***Pin Change Mask***)
- Existem três registradores PCMSK, um para cada grupo
- Cada **bit** no registrador PCMSK corresponde a um **pino PCINT**

Pin Change Mask Register

PCMSK0

Bit	7	6	5	4	3	2	1	0
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

PCMSK1

Bit	7	6	5	4	3	2	1	0
		PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

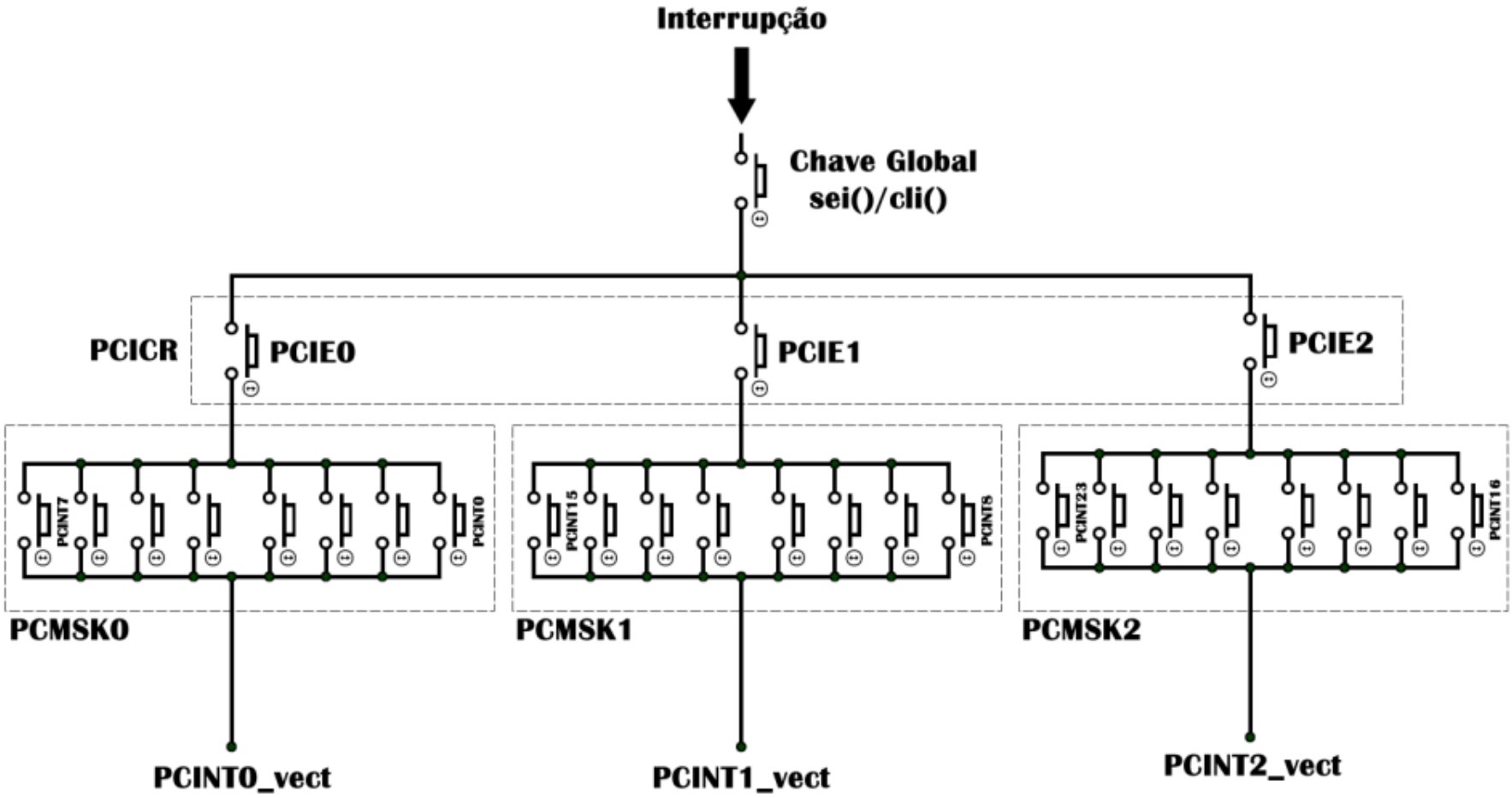
PCMSK2

Bit	7	6	5	4	3	2	1	0
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Global Interrupt Flag (SREG)

- Essa flag é responsável por controlar as interrupções de todo o microcontrolador
- Funciona como uma chave geral
- Uma maneira de modifica-la é através das macros **`sei()`** e **`cli()`**
- **sei()**: habilita as interrupções globalmente;
- **cli()**: bloqueia as interrupções globalmente

Analogia com Chaves



Código 1

Habilitar a interrupção por troca de estados no pino A0

https://www.teachmemicro.com/arduino-interrupt-tutorial/

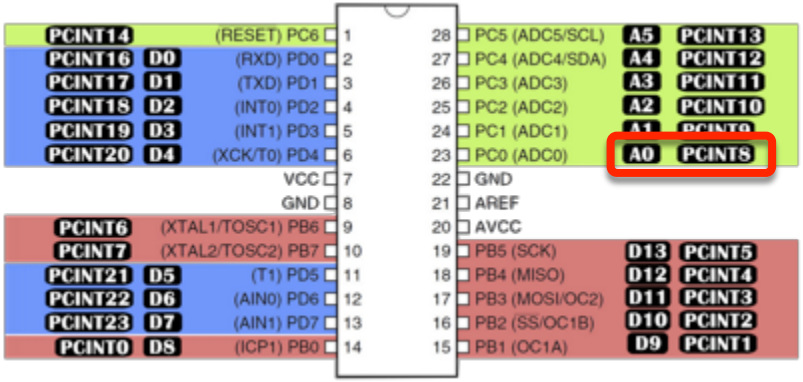
Mask Register

Devemos definir o bit PCMSK para o pino A0

Este pino é o PCINT8 e, portanto, pertence ao PCMSK1

Então, esta linha deve ser incluída no sketch

```
PCMSK1 = B00000001; // habilita PCINT8
```



PCMSK1

Bit	7	6	5	4	3	2	1	0
		PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Control Register

Habilitamos a interrupção de grupo à qual **A0** pertence, no caso, **PCIE1**

Relembrando que quando o grupo **PCIE1** é setado, os pinos **PCINT8** a **PCINT14** são habilitados

```
PCICR  = B00000010; // habilita grupo PCIE1
```

Flags

Devemos limpar qualquer flag de interrupção porque eles não são redefinidos automaticamente

Como **A0** é **PCINT8**, portanto, pertence a **PCIF1**

Serão limpos todos os flags de interrupção

```
PCIFR  = B00000000; // limpa todos os flags
```

ISR

O AVR tem sua própria função ISR() que aceita o vetor de interrupção para cada grupo

O grupo **PCIE1** possui **`PCINT1_vect`** como seu respectivo vetor

Como estaremos usando apenas **A0** neste exemplo, este é o único que precisamos

```
ISR(PCINT1_vect) {
   state = !state;
}
```

Sketch completo

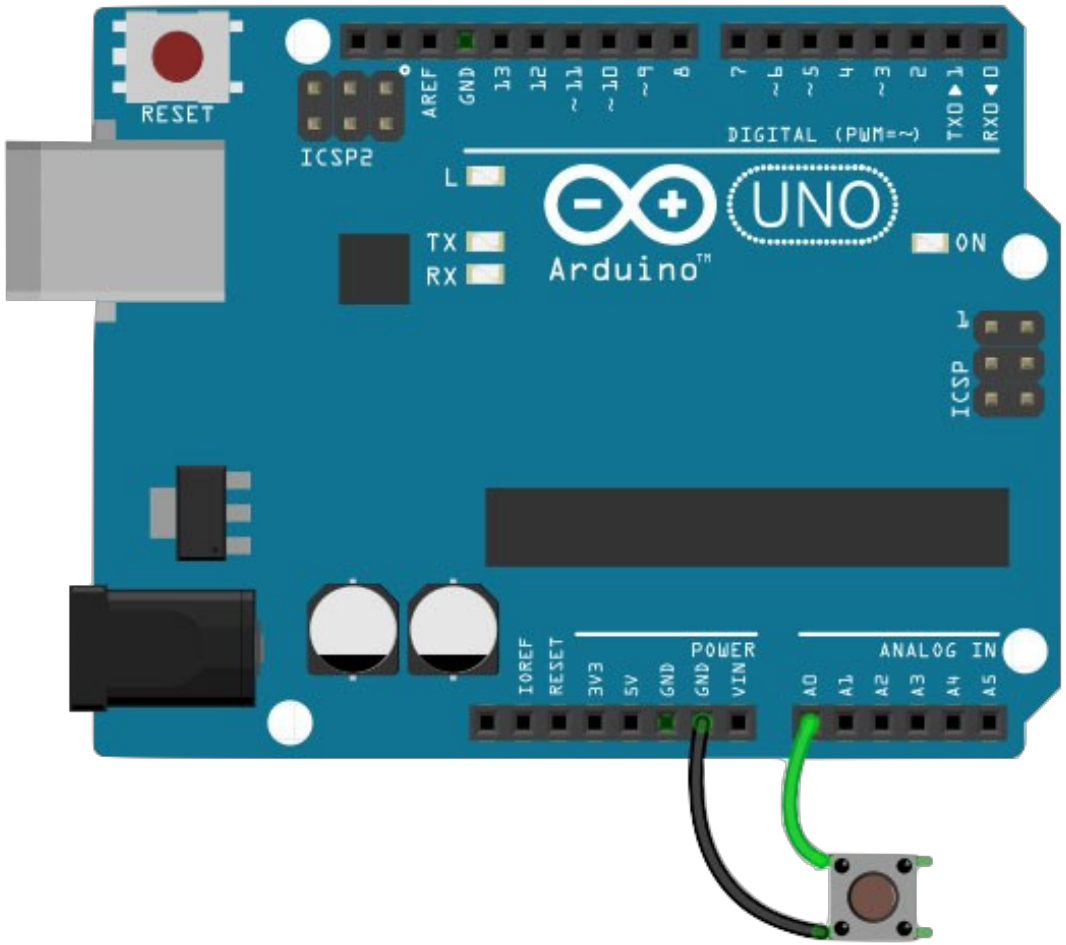
```
const byte ledPin = 13;
volatile byte state = LOW;

void setup(){
   pinMode(A0, INPUT_PULLUP);
   PCICR  = B00000010; // habilita grupo PCIE1
   PCMSK1 = B00000001; // habilita PCINT8
   PCIFR  = B00000000; // limpa todos os flags
}

void loop(){
   digitalWrite(ledPin, state);
}

ISR(PCINT1_vect) {
   state = !state;
}
```

Circuito


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

Pin Change Interrupt Library

- Se manipular registradores não é sua praia, existe uma biblioteca simples de interrupção de mudança de estados de pinos que você pode usar, que é a **`PinChangeInt`**
- A biblioteca pode ser baixada de **`https://code.google.com/archive/p/arduino-pinchangeint/downloads`**

Pin Change Interrupt Library

```
#include <PinChangeInt.h>
const byte ledPin = 13;
volatile byte state = LOW;

void setup(){
   pinMode(ledPin, OUTPUT);
   pinMode(A0, INPUT_PULLUP);
   PCintPort::attachInterrupt(A0, isr, CHANGE);
}

void loop(){
   digitalWrite(ledPin, state);
}

void isr() {
   state = !state;
}
```

Código 2

Habilitar a interrupção por troca de estados no pino D12

Usaremos um pouco mais de baixo nível

https://portal.vidadesilicio.com.br/pcint-interrupcoes-por-mudanca-de-estado/

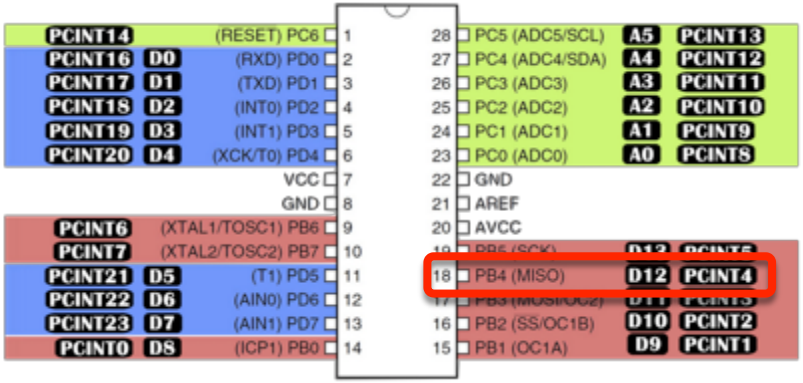
Definição dos Registradores

Pino D12 é equivalente ao pino PB4, que deve ser de entrada com registrador PULL UP

Vamos inicialmente desligar as interrupções globalmente

Pino PB4 possui a interrupção PCINT4 que é habilitada por PCMSK0, que é habilitada por PCIE0

O tratador de interrupção será feito através da macro ISR() que recebe como parâmetro o vetor de interrupção PCINT0_vect



PCMSK0

Bit	7	6	5	4	3	2	1	0
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Código 2

```
void setup() {
  // desliga interrupcoes globalmente
  cli();

  // Equivalente a pinMode(12, INPUT_PULLUP);
  DDRB  &= ~(1 << DDB4);  // Seta D12 como entrada;
  PORTB |= (1 << PORTB4); // Liga Pull-up;

  // Seta registradores de interrupção
  PCICR  |= (1 << PCIE0);
  PCMSK0 |= (1 << PCINT4);

  // liga interrupcoes globalmente
  sei();
}
```

Código 2

```
void loop() {
  // . . .
}

ISR(PCINT0_vect) {
  if (PINB & (1 << PINB4)) {
    // D12 mudou de LOW para HIGH
  }
  else {
    // D12 mudou de HIGH para LOW
  }
}
```

Código 3

Habilitar a interrupção por troca de estados no pino D10, D11 e D12

Usaremos um pouco mais de baixo nível

https://portal.vidadesilicio.com.br/pcint-interrupcoes-por-mudanca-de-estado/

Definição dos Registradores

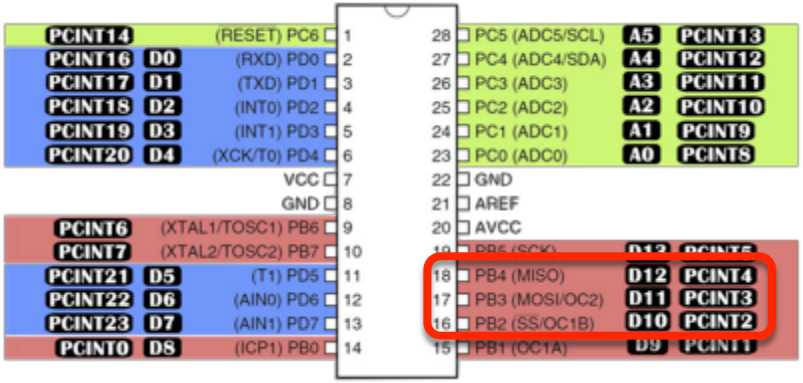
Pinos D10, D11 e D12 são equivalentes aos pinos PB2, PB3 e PB4, que devem ser de entrada e com registrador PULL UP

Pinos PB2, PB3 e PB4 possuem as interrupções PCINT2, PCINT3 e PCINT4 que são habilitadas por PCMSK0 e por PCIE0

A macro ISR() receberá o vetor de interrupção PCINT0_vect

Agora **um mesmo vetor** de interrupção **será chamado por vários pinos**

Para definir qual pino causou a interrupção devemos **guardar um histórico** do ultimo estado de todo o PORTB



PCMSK0

Bit	7	6	5	4	3	2	1	0
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Código 3

```
void setup() {
  cli();

  /* Equivalente a
  pinMode(12, INPUT_PULLUP);
  pinMode(11, INPUT_PULLUP);
  pinMode(10, INPUT_PULLUP);
  */

  DDRB &= ~( (1 << DDB4) | (1 << DDB3) | (1 << DDB2) );

  PORTB |= ( (1 << PORTB4) |
             (1 << PORTB3) |
             (1 << PORTB2) );
```

Código 3

```
  // Seta registradores de interrupção
  PCICR |= (1 << PCIE0);

  PCMSK0 |= ( (1 << PCINT4) |
              (1 << PCINT3) |
              (1 << PCINT2) );

  sei();
}

void loop() {
  //...
}
```

Código 3

```
// Variáveis globais que devem ser declaradas volatile
volatile uint8_t last_PINB = PINB;

/* Função de Tratamento de Interrupção */
ISR(PCINT0_vect) {
  uint8_t changed_bits;
  changed_bits = PINB ^ last_PINB;
  last_PINB = PINB;
```

Código 3

```
if (changed_bits & (1 << PINB4)) {
  if (PINB & (1 << PINB4)) {
    // D12 mudou de LOW para HIGH;
  }
  else {// D12 mudou de HIGH para LOW}
}
else if (changed_bits & (1 << PINB3)) {
  if (PINB & (1 << PINB3)) {
    // D11 mudou de LOW para HIGH
  }
  else {// D11 mudou de HIGH para LOW}
}
else if (changed_bits & (1 << PINB2)){
  if (PINB & (1 << PINB2)) {
    // D10 mudou de LOW para HIGH
  }
  else {// D10 mudou de HIGH para LOW}
  }
}
```