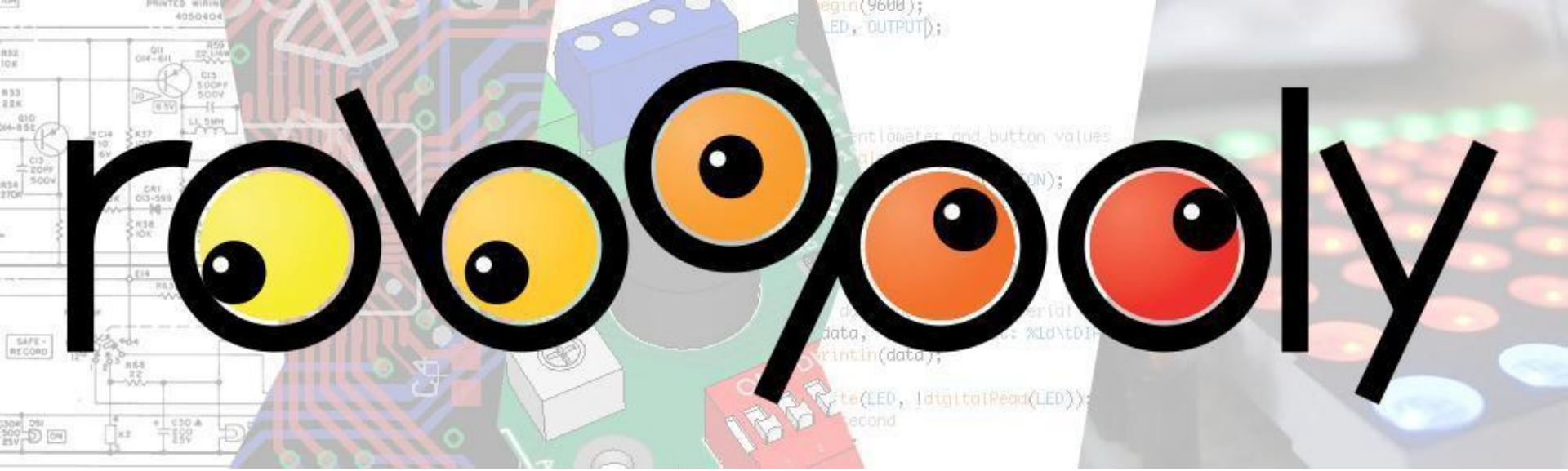
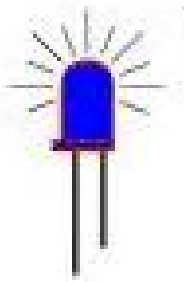
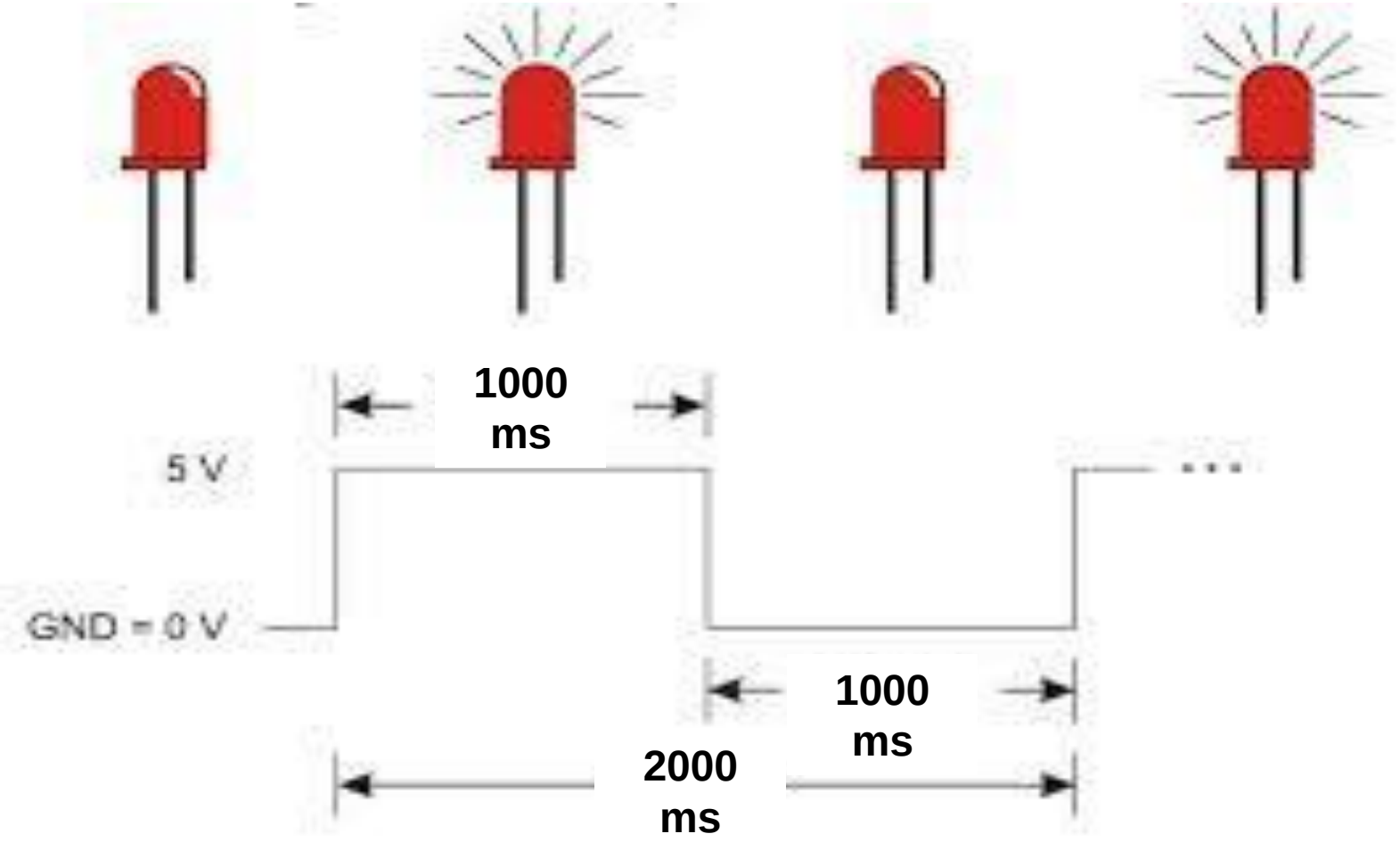


INTERRUPTIONS ET TIMERS

Système d´alarme



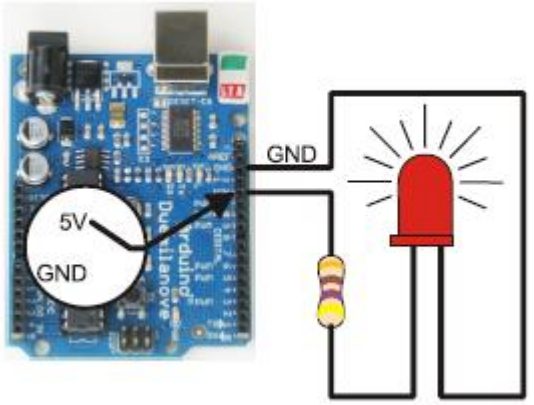
Blinking LED (2 secondes)



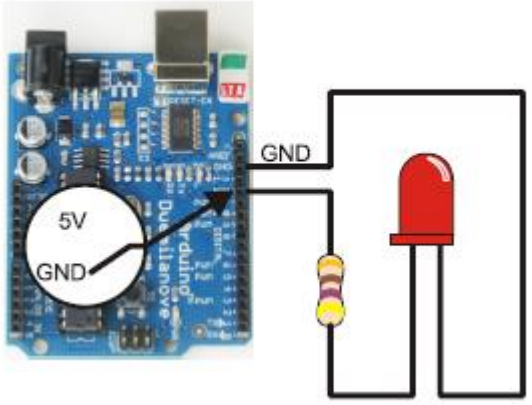
Blinking LED (2 secondes)

```
#define LED_PIN 13;

void setup()
{
  pinMode(LED_BLINK_PIN, OUTPUT);
}

void loop()
{
 digitalWrite(LED_BLINK_PIN, HIGH);
 delay(1000);
 digitalWrite(LED_BLINK_PIN, LOW);
 delay(1000);
}
```

```
digitalWrite(13, HIGH);
```




```
digitalWrite(13, LOW);
```

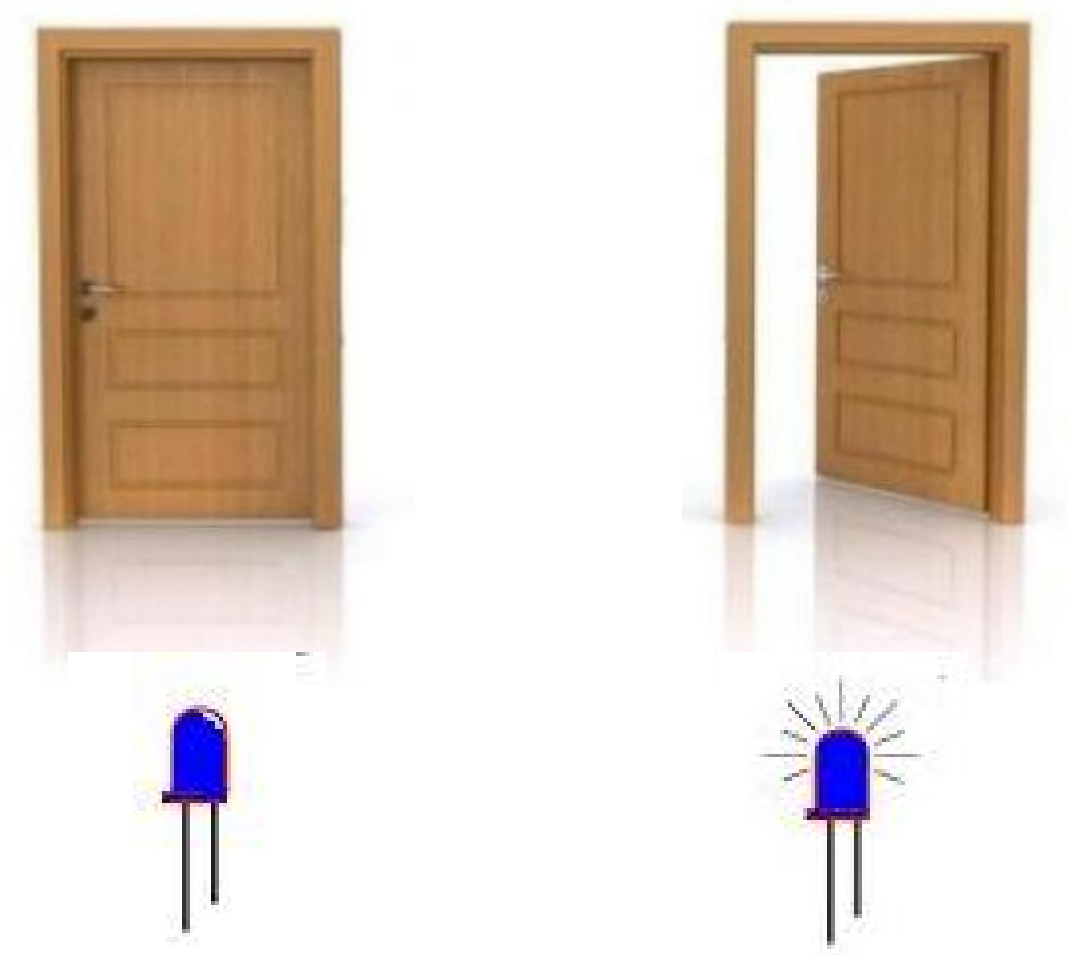


État de la Porte

Capteur = 0

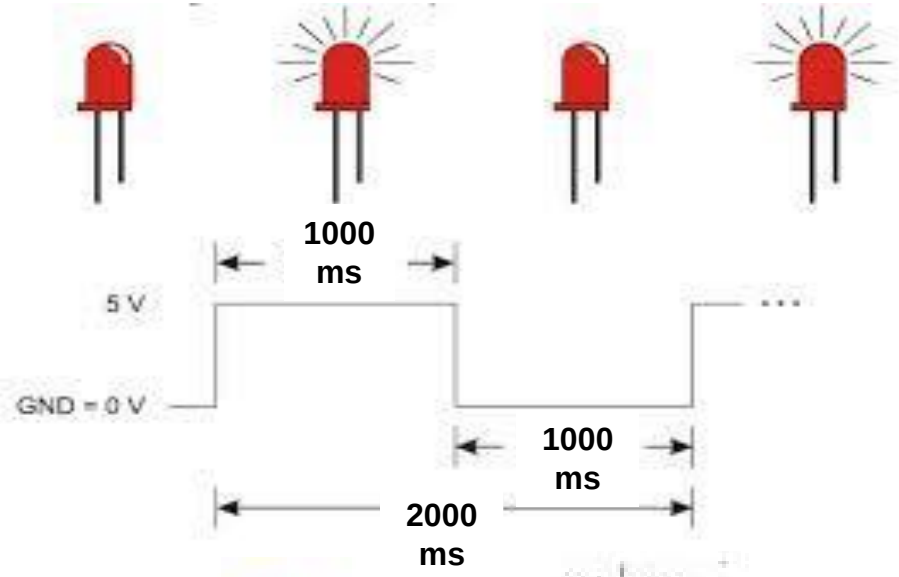
Capteur = 1

État de la Porte: LED Bleu



Blinking LED + État de la Porte

```
void setup()
{
  pinMode(LED_BLINK_PIN, OUTPUT);
  pinMode(LED_PORTE_PIN, OUTPUT);
  pinMode(PORTE_PIN, INPUT);
}

void loop()
{
  digitalWrite(LED_BLINK_PIN, HIGH);
  if (digitalRead(PORTE_PIN))
    digitalWrite(LED_PORTE_PIN, HIGH);
  else
    digitalWrite(LED_PORTE_PIN, LOW);
  delay(1000);

  digitalWrite(LED_BLINK_PIN, L
  if (digitalRead(PORTE_PIN))
    digitalWrite(LED_PORTE_PIN, HIGH);
  else
    digitalWrite(LED_PORTE_PIN, LOW);
  delay(1000);
}
```



1 s. pour les voleurs !!!

État de la porte plus souvent

```
void setup()
{
  pinMode(LED_BLINK_PIN, OUTPUT);
  pinMode(LED_PORTE_PIN, OUTPUT);
  pinMode(PORTE_PIN, INPUT);
}

void loop()
{
  digitalWrite(LED_BLINK_PIN, HIGH);
  if (digitalRead(PORTE_PIN))
    digitalWrite(LED_PORTE_PIN, HIGH);
  else
    digitalWrite(LED_PORTE_PIN, LOW);
  delay(1000);

  digitalWrite(LED_BLINK_PIN, LOW);
  if (digitalRead(PORTE_PIN))
    digitalWrite(LED_PORTE_PIN, HIGH);
  else
    digitalWrite(LED_PORTE_PIN, LOW);
  delay(1000);
}
```

PAS ESCALABLE !!!

Recode les délais

La LED clignote plus vite !

Contrôler un capteur ?

```
while(1)
  valeur = digitalRead(pin_capteur);
```

- Deux possibilités
  - Polling

  - Interruption externe

Interruptions, the Arduino way.

attachInterrupt (Prismino ~= Leonardo)

```
void setup()
{
  pinMode(LED_BLINK_PIN, OUTPUT);
  pinMode(LED_PORTE_PIN, OUTPUT);
  pinMode(PORTE_PIN, INPUT);
  attachInterrupt(PORTE_PIN, verifierPorte, CHANGE);
}
```

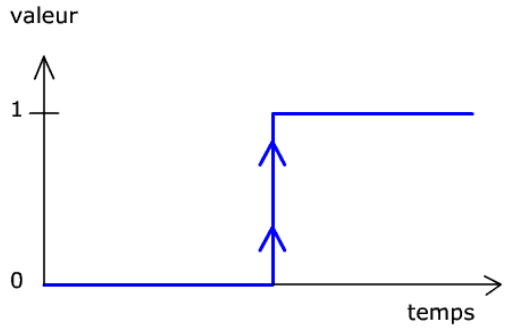
CALLBACK function: la fonction qu'on appele quand l'interruption arrive.

```
void loop()
{
  digitalWrite(LED_BLINK_PIN, HIGH);
  delay(500);
  digitalWrite(LED_BLINK_PIN, LOW);
  delay(500);
}
```

```
void verifierPorte()
{
 if (digitalRead(PORTE_PIN))
    digitalWrite(LED_PORTE_PIN, HIGH);
  else
    digitalWrite(LED_PORTE_PIN, LOW);
}
```

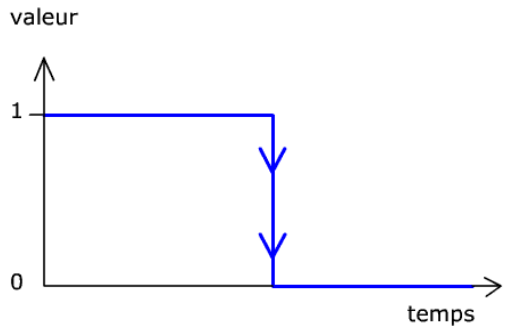
CHANGE to trigger the interrupt whenever the pin changes value
RISING to trigger when the pin goes from low to high,
FALLING for when the pin goes from high to low.
LOW to trigger the interrupt whenever the pin is low
HIGH to trigger the interrupt whenever the pin is high. (*Arduino Due only*)

Les flancs



- de 0 à 1 = flanc montant

La porte s'ouvre !



- de 1 à 0 = flanc descendant

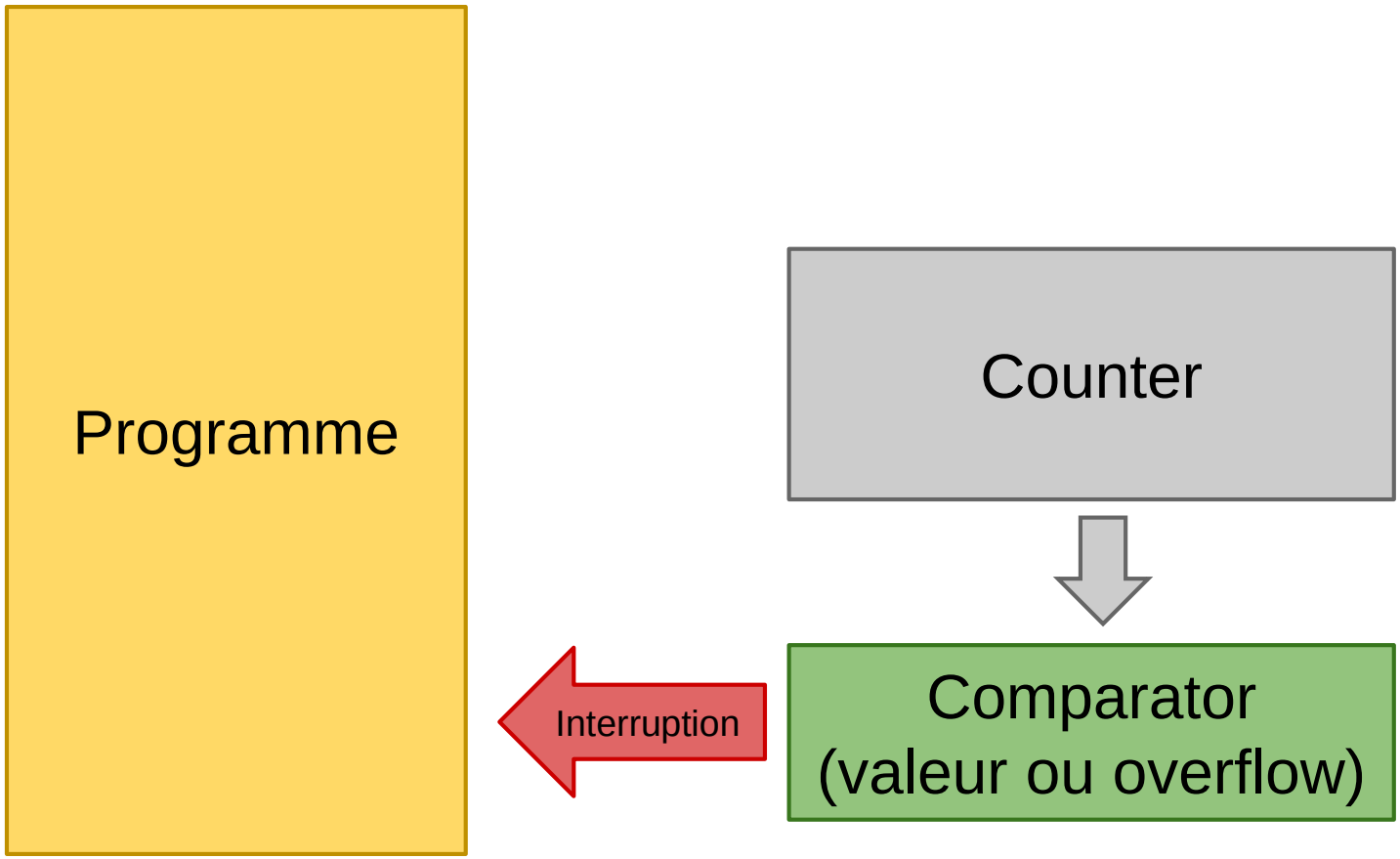
La porte se ferme !

Interruptions, pourquoi?

Comment faire
plusieurs
choses à la fois?

Une autre interruption : Les Timers

Compter le temps : le Timer



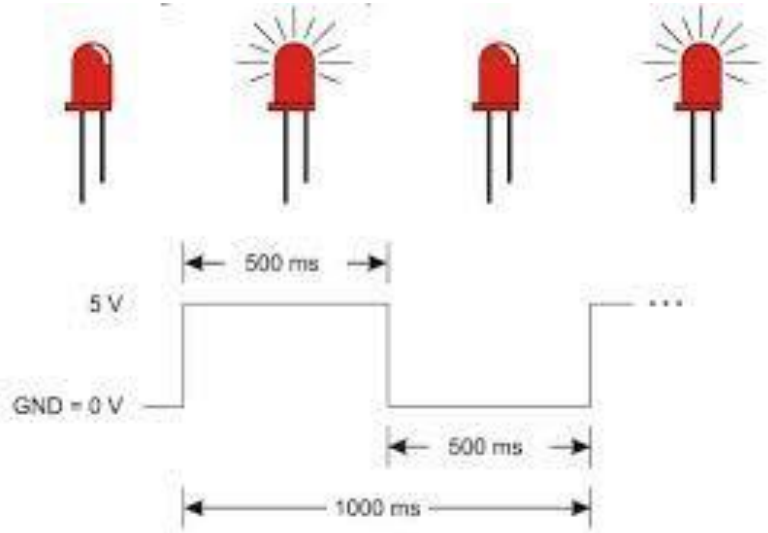
Interruptions, types

- External
  - Viens d'un pin externe
  - "Je veux faire quelque chose quand une valeur mesurée change"
  - attachInterrupt(interrupt, fonction, quand)
- Internal
  - Viens d'un périphérique interne e.g.: Timer
  - "Je veux faire quelque chose (**fonction**) dans **x** ms pour **n** fois"
  - setTimer(fonction, x, n)

Blinking LED + État de la Porte

```
int ledState = LOW;

void setup()
{
  pinMode(LED_BLINK_PIN, OUTPUT);
  pinMode(LED_PORTE_PIN, OUTPUT);
  pinMode(PORTE_PIN, INPUT);
  setTimer(blinkLED, 5);
}

void loop()
{
    if (digitalRead(PORTE_PIN))
       digitalWrite(LED_PORTE_PIN, HIGH);
    else
       digitalWrite(LED_PORTE_PIN, LOW);
}

void blinkLED()
{
    if (ledState == LOW)
        ledState = HIGH;
    else
        ledState = LOW;
    digitalWrite(LED_BLINK_PIN, ledState);
}
```

Appel la fonction chaque:
5*100ms = 500ms



Les Timers: Librairie Robopoly

La librairie robopoly contient des fonctions toute faites pour les utiliser plus facilement: https://github.com/Robopoly/prismino-library

int8_t setTimer(func_t callbackFunction,
uint16_t interval);

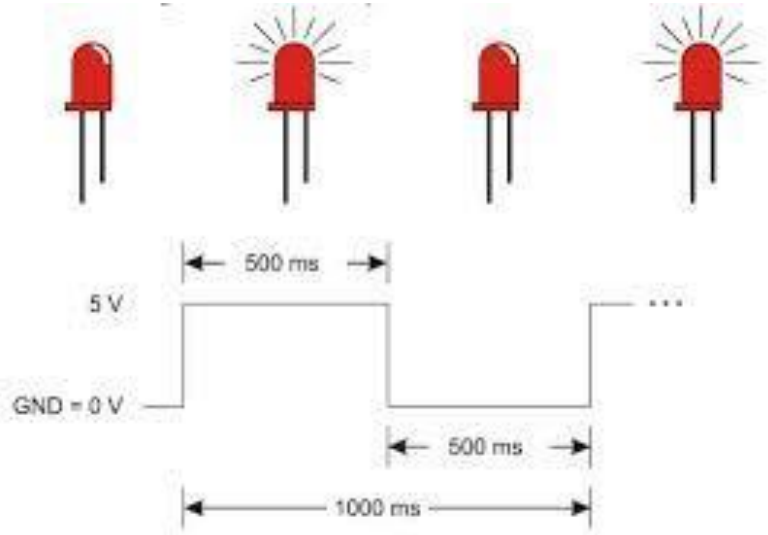
Appel la fonction *“callbackFunction”*, chaque: *“interval”**100ms.

int8_t setTimer(func_t callbackFunction,
uint16_t interval,
uint8_t callNumber = 0);

Appel la fonction *“callbackFunction”*, chaque: *“interval”**100ms, *“callNumber”* fois.

Quelles Interruptions ???

```
void setup()
{
  attachInterrupt(PORTE_PIN, verifierPorte, CHANGE);
  setTimer(blinkLED, 5);
}

void loop()
{
  verifierPorte?
  blinkLED?
}

void blinkLED()
{
  …
}

void verifierPorte()
{
  ...
}
```



Autres fonctions pour les interruptions dans le Arduino

attachInterrupt(interrupt, ISR, mode)
attachInterrupt(pin, ISR, mode)

detachInterrupt(*interrupt*)
detachInterrupt(*pin*)

noInterrupts()
interrupts()

Happy Interrupting !

FIN

Questions?

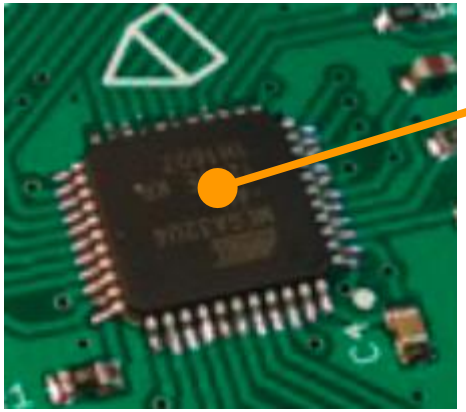
Prochains évenements

Lundi 1 décembre
Caméra linéaire

Lundi 8 décembre
Présentation des règles grand concours

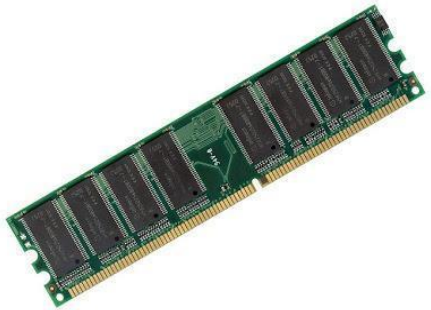
28.3.15 (à confirmer)
Grand concours !

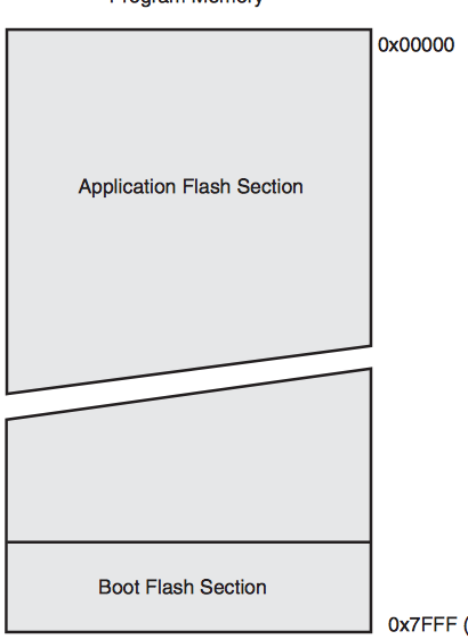
Il y a quoi dedans?

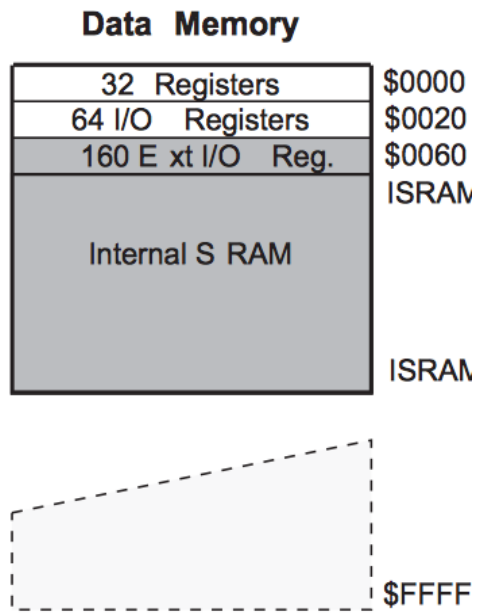


Hard disk

RAM







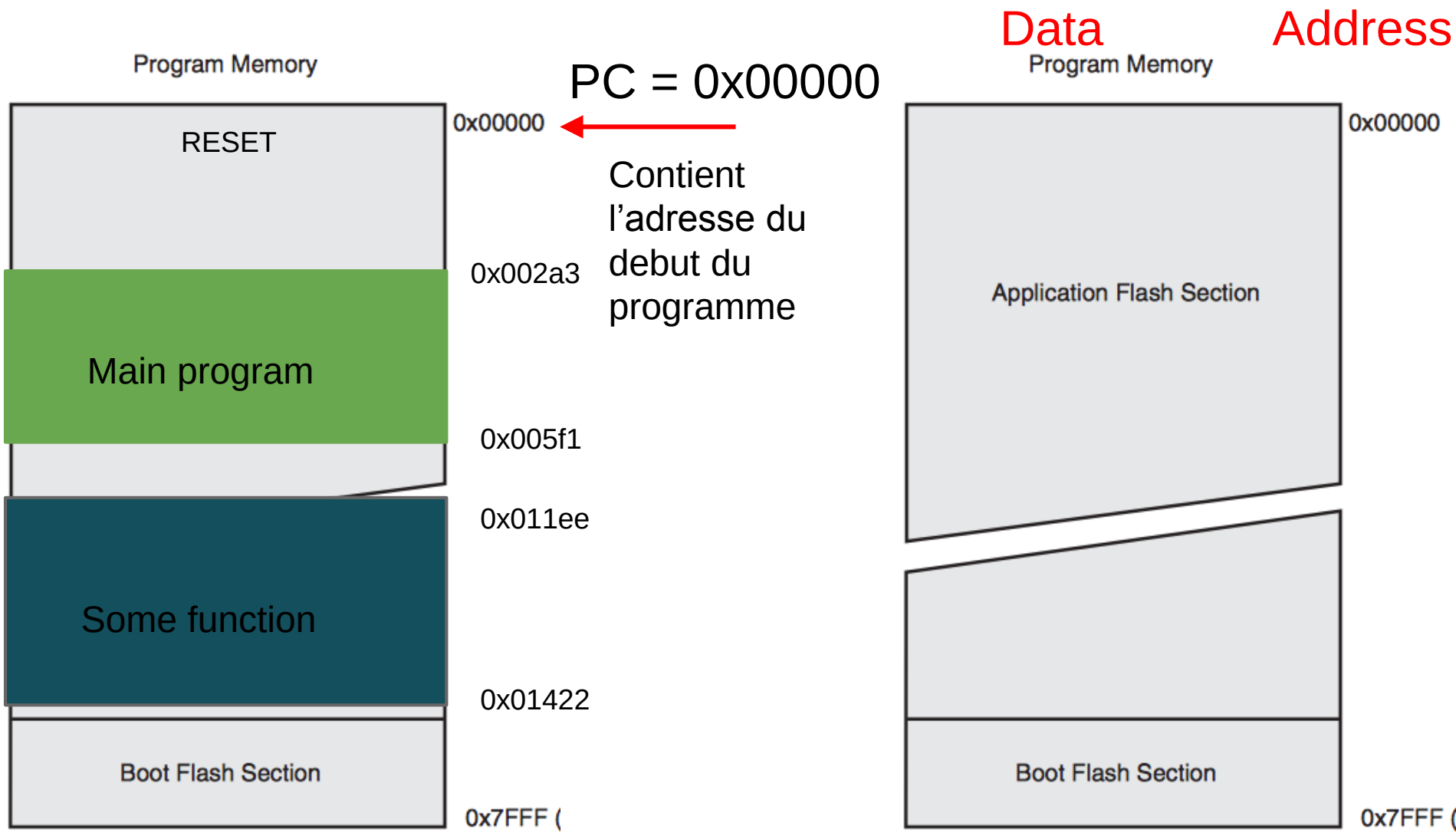
Ou, les datasheets

Atmel AtMega32U4 datasheet http://www.atmel.com/Images/doc7766.pdf

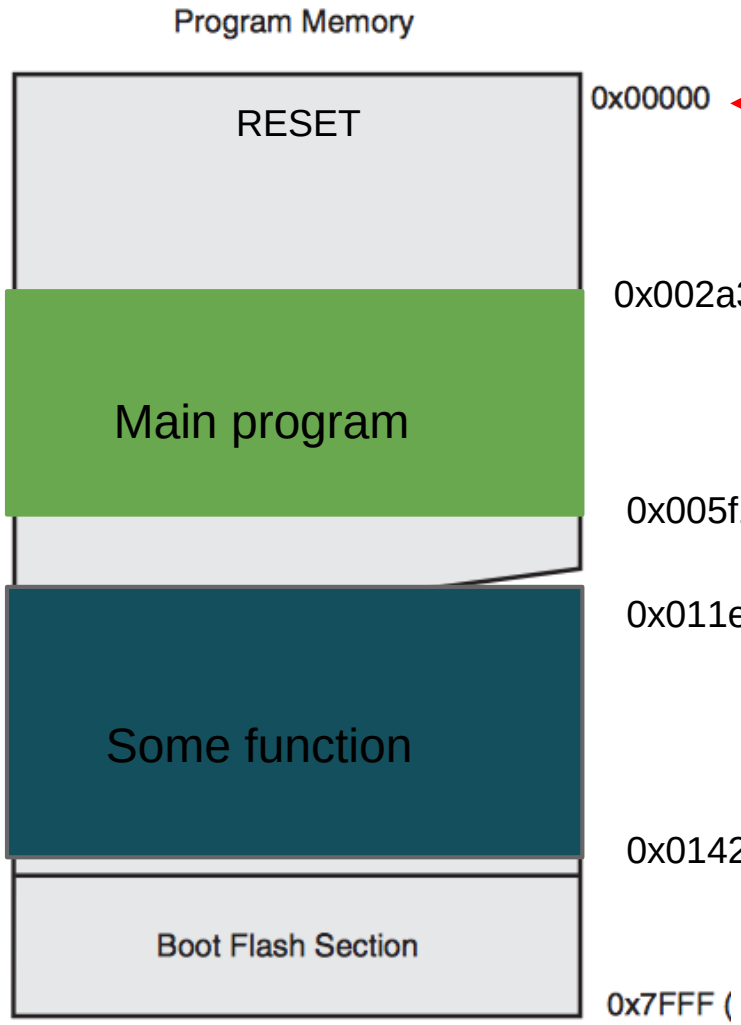
Mon PC

- N’est **pas** un Personal Computer
- Plutôt, un Program Counter
- Marque/compte quelle ligne de code/instruction est à executer
- Au reset, PC = $0x0000
- $ → Adresse, 0x → hexadecimal

Memoire programme



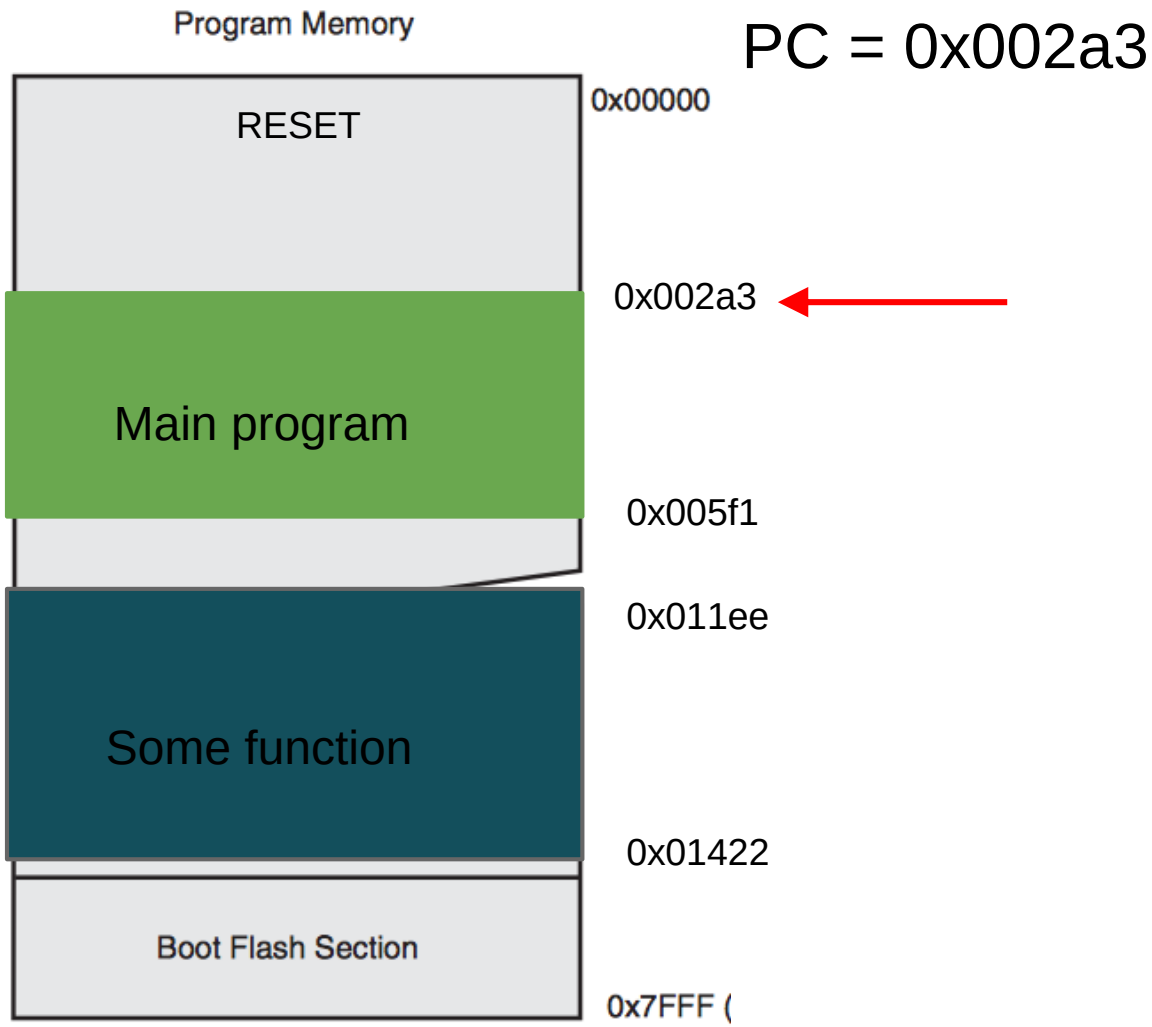
Memoire programme



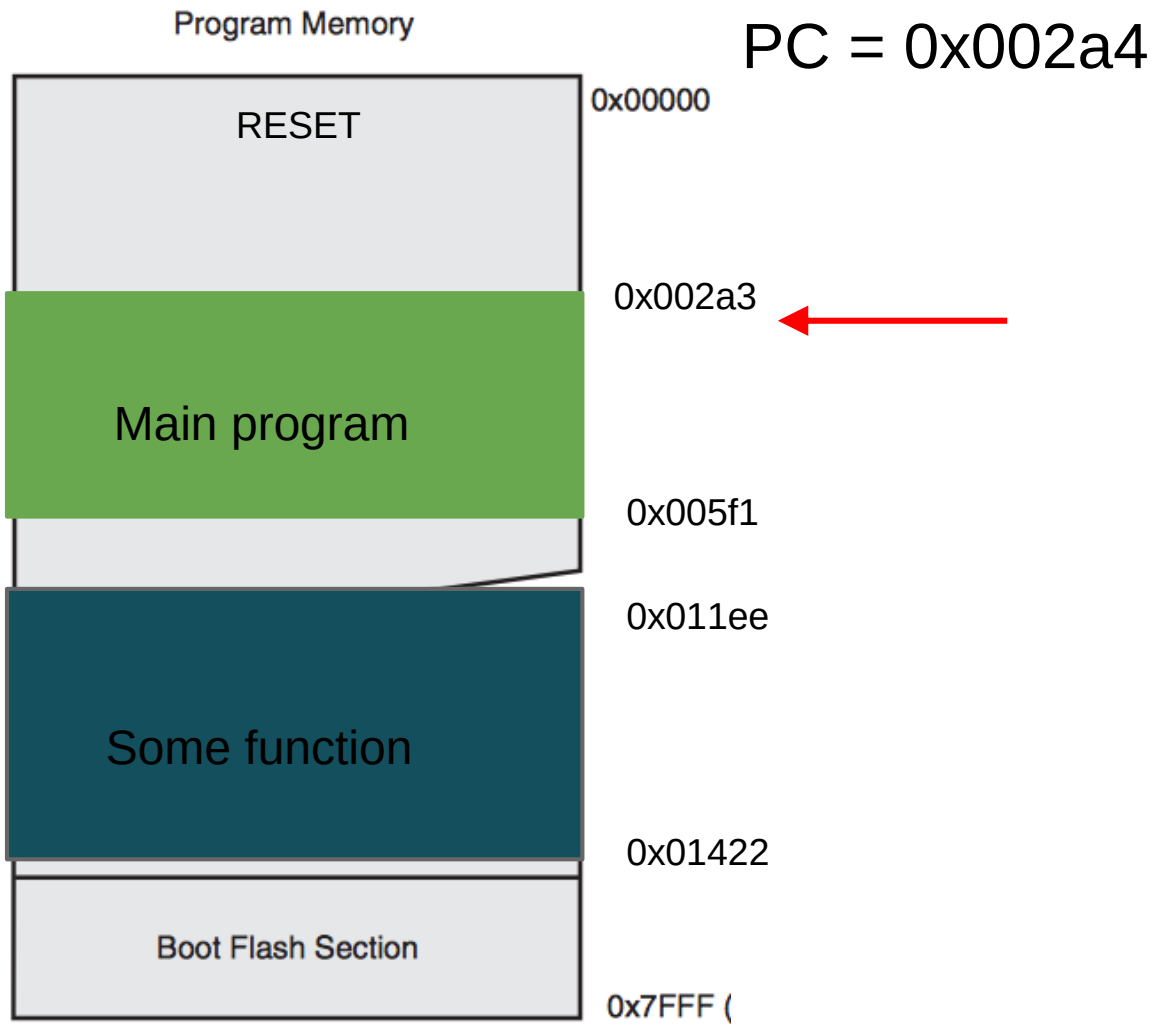
On focalise dans la memoire de programme

(memoire d'instructions)

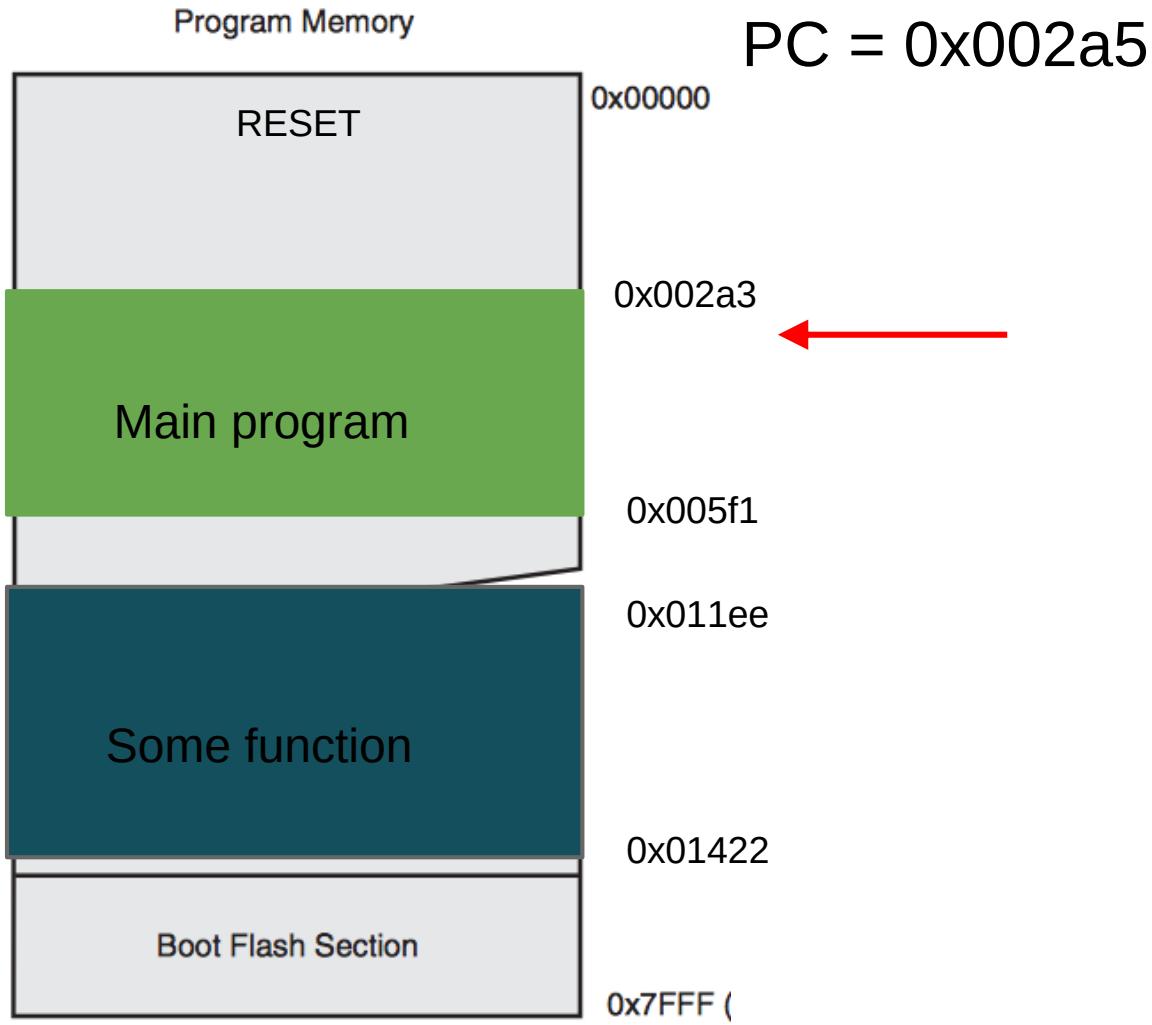
Memoire programme



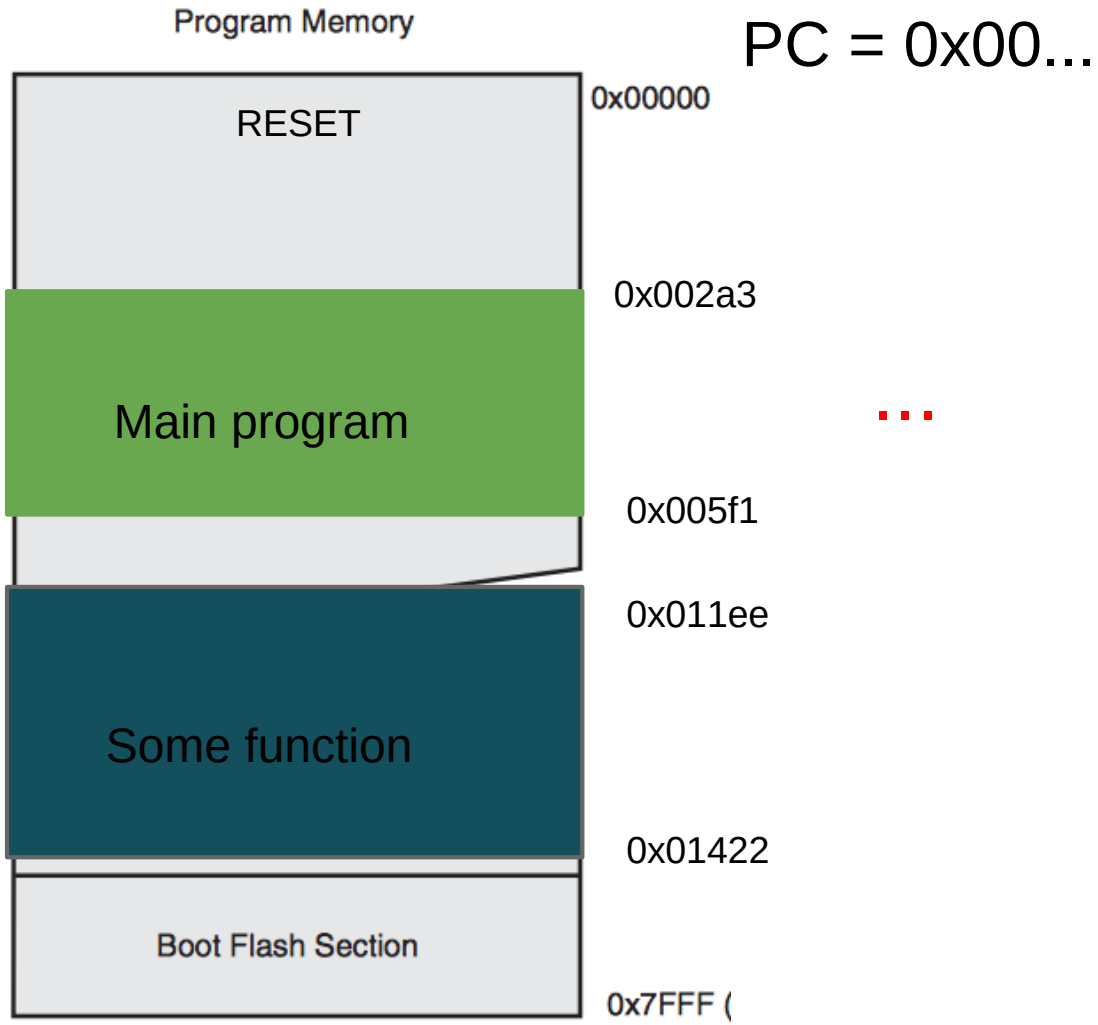
Memoire programme



Memoire programme



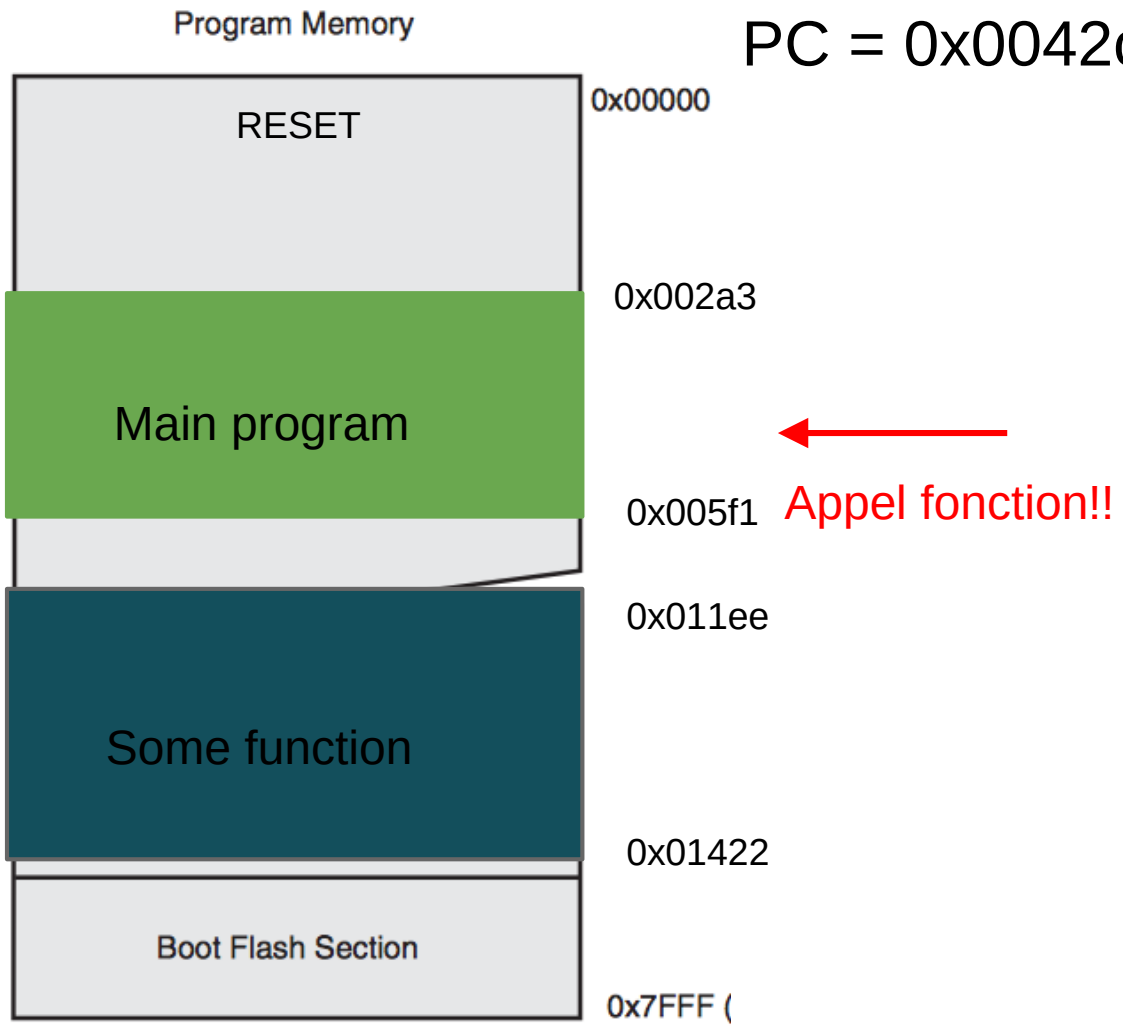
Memoire programme



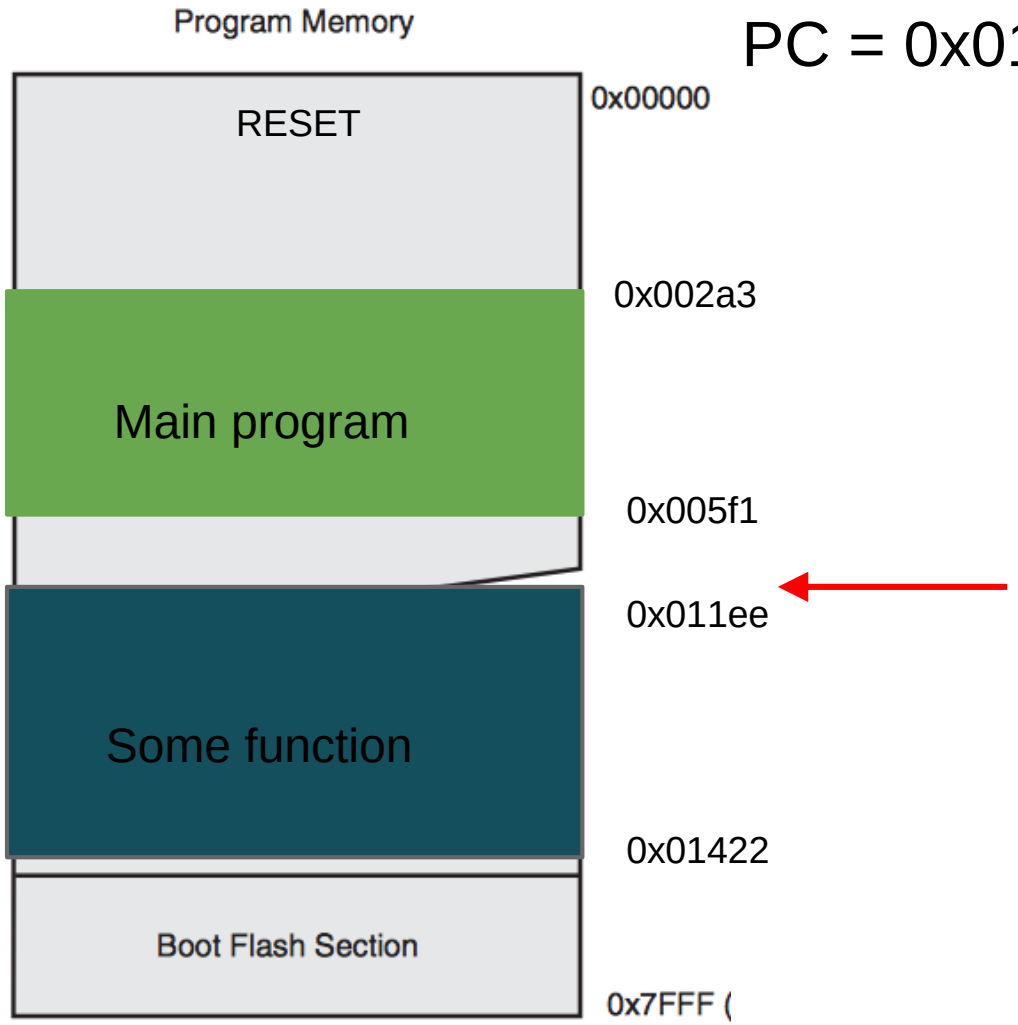
PC = 0x00...

...

Memoire programme

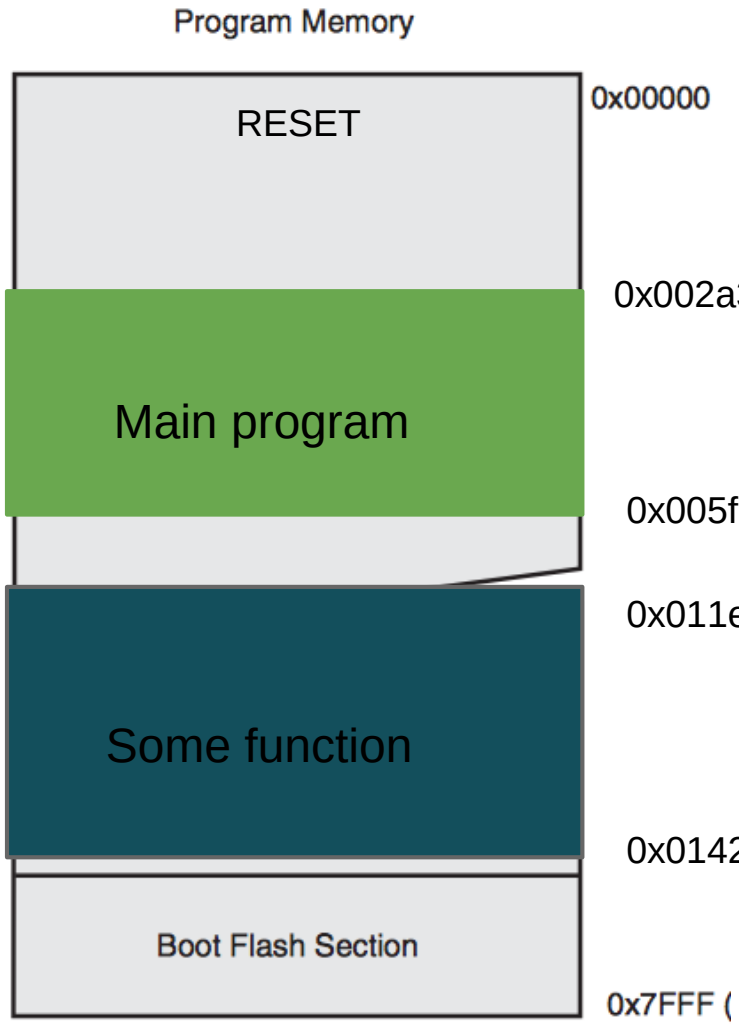


Memoire programme



PC = 0x011ee

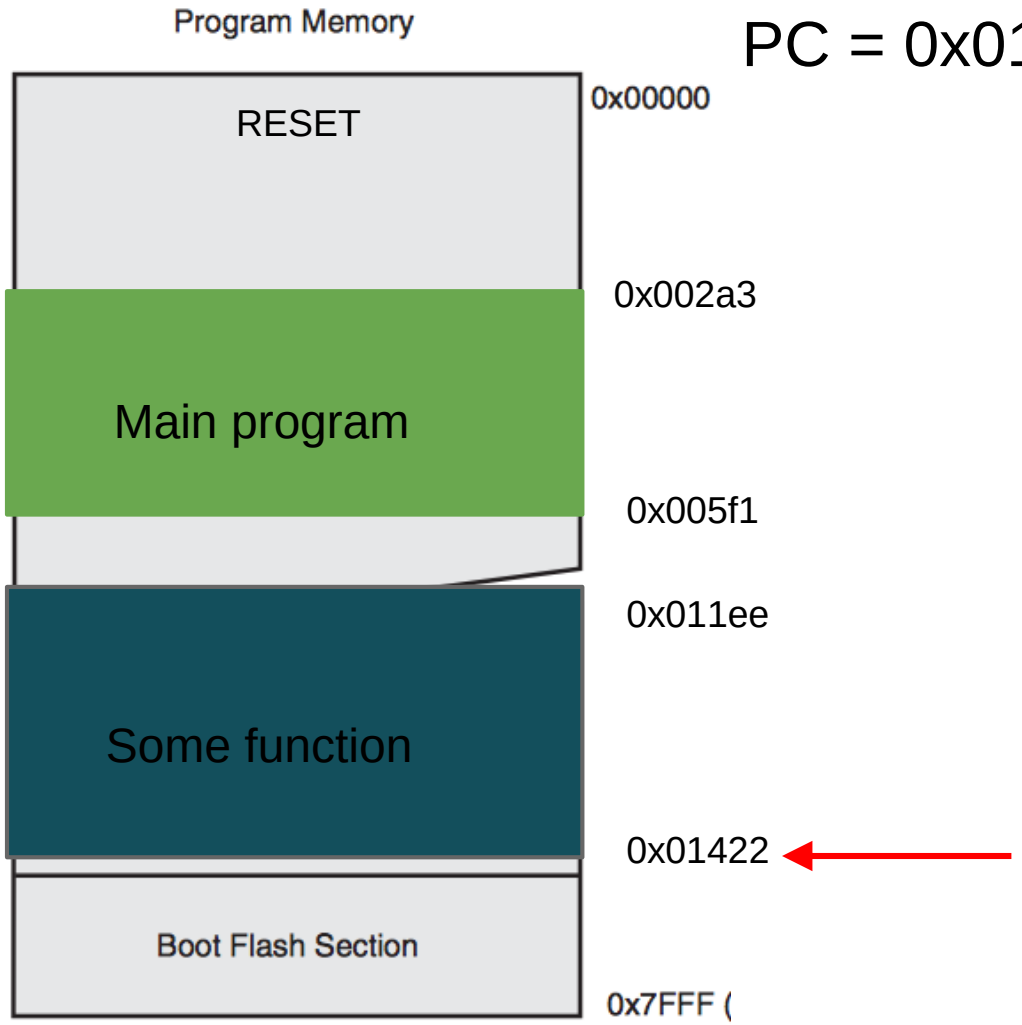
Memoire programme



PC = 0x0....

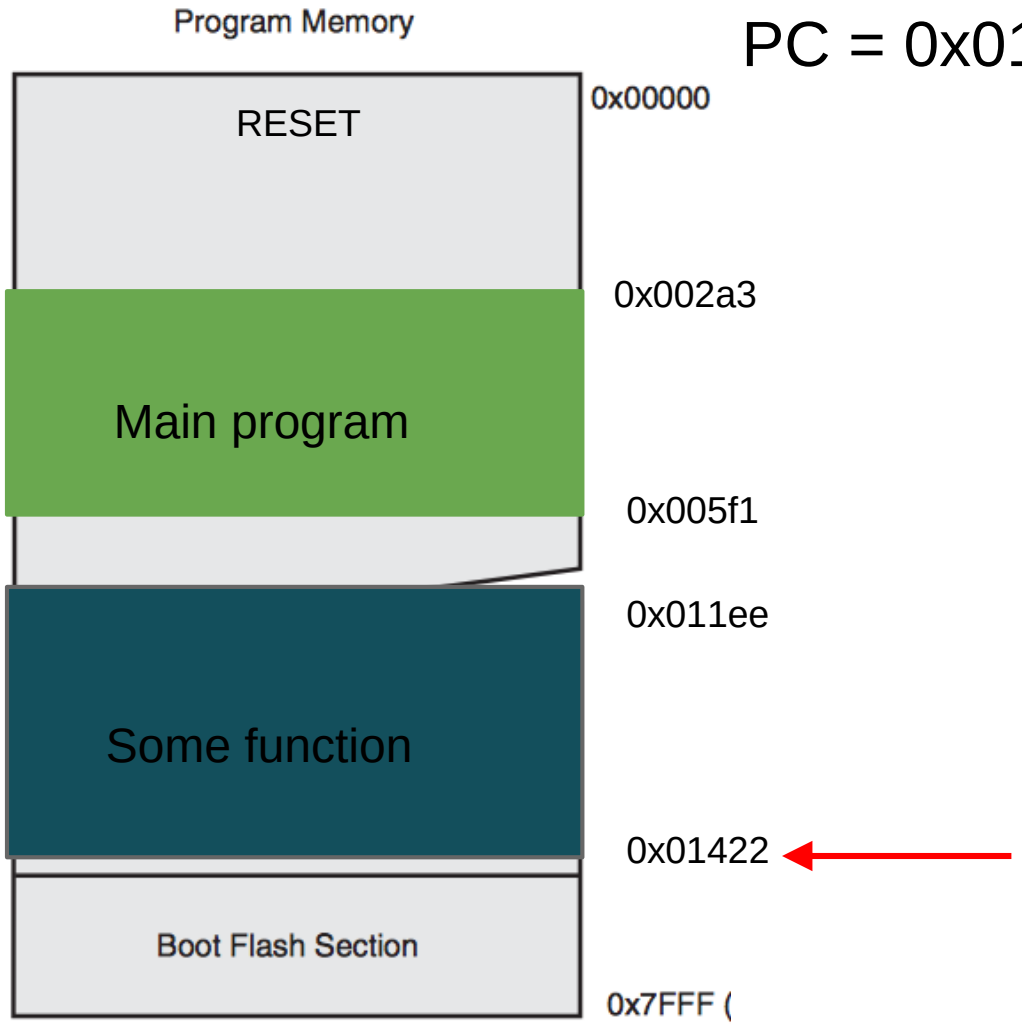
...

Memoire programme



PC = 0x01422

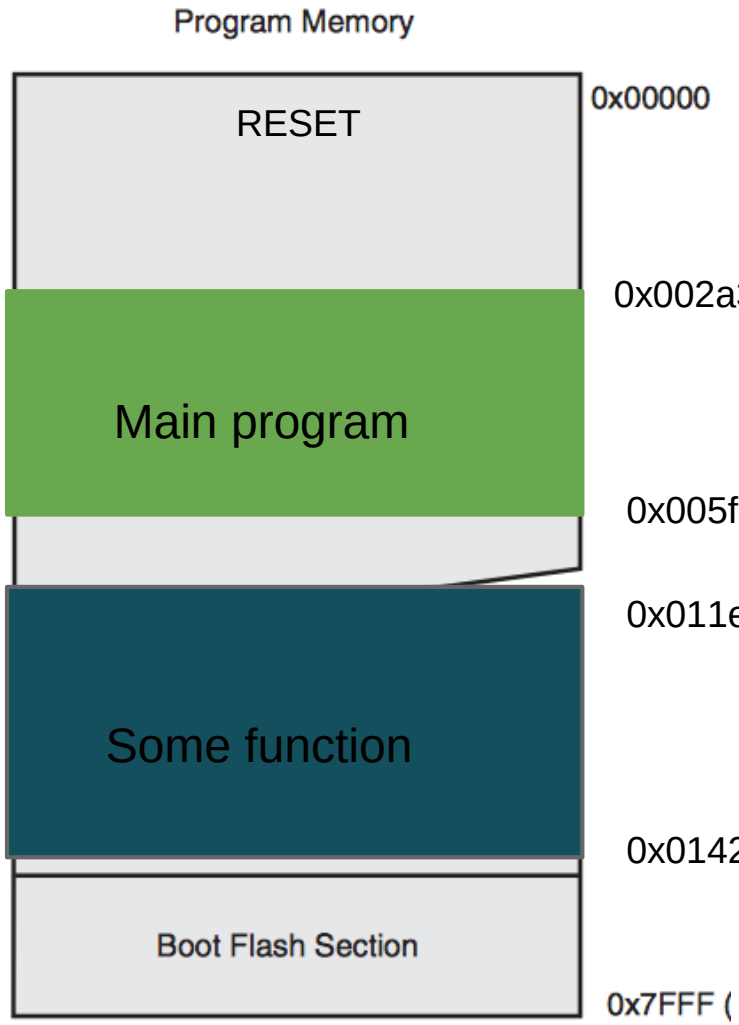
Memoire programme



PC = 0x01422

Fonction appelé avec:
PC = 0x0042c

Memoire programme

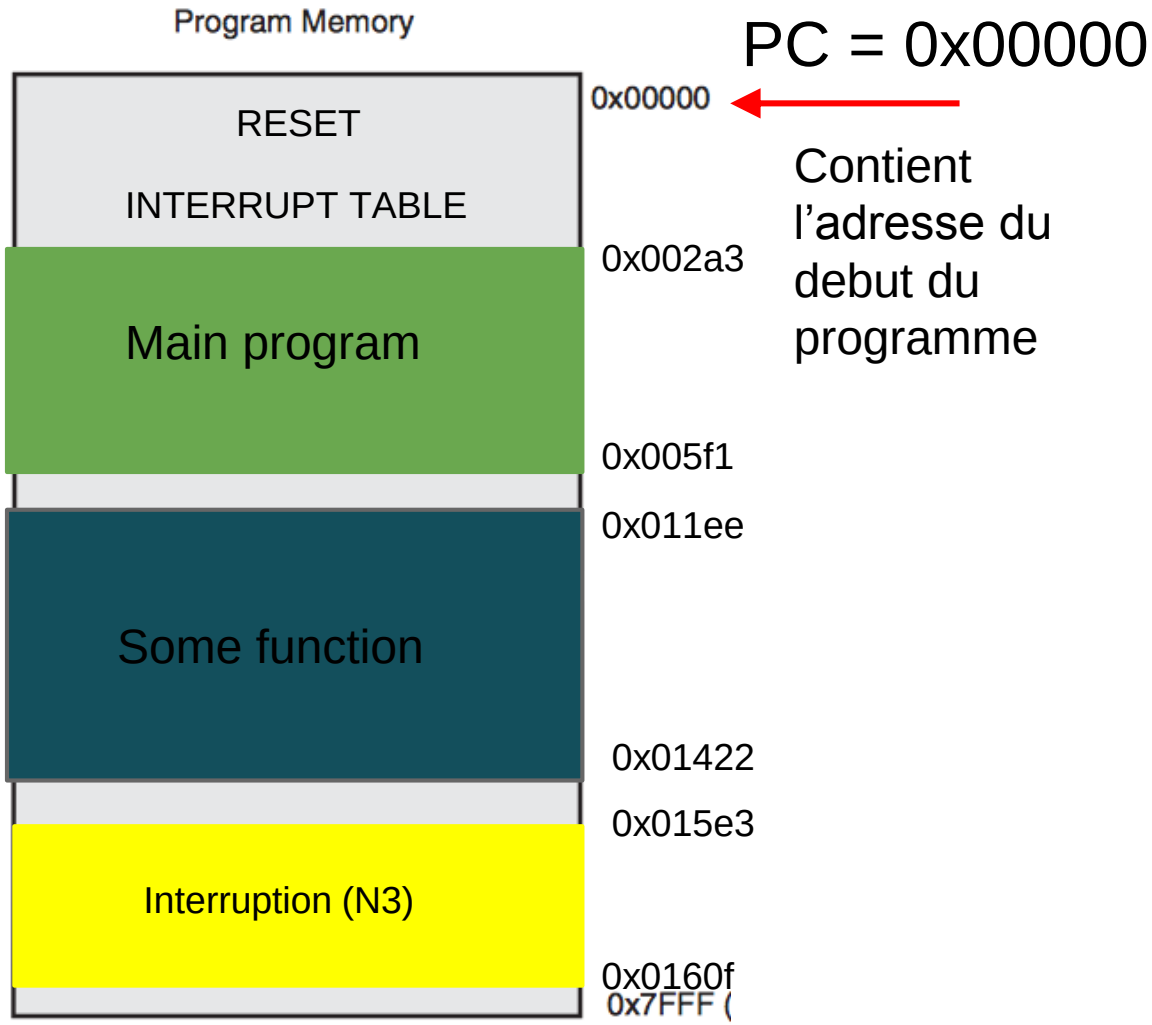


PC = 0x0042d

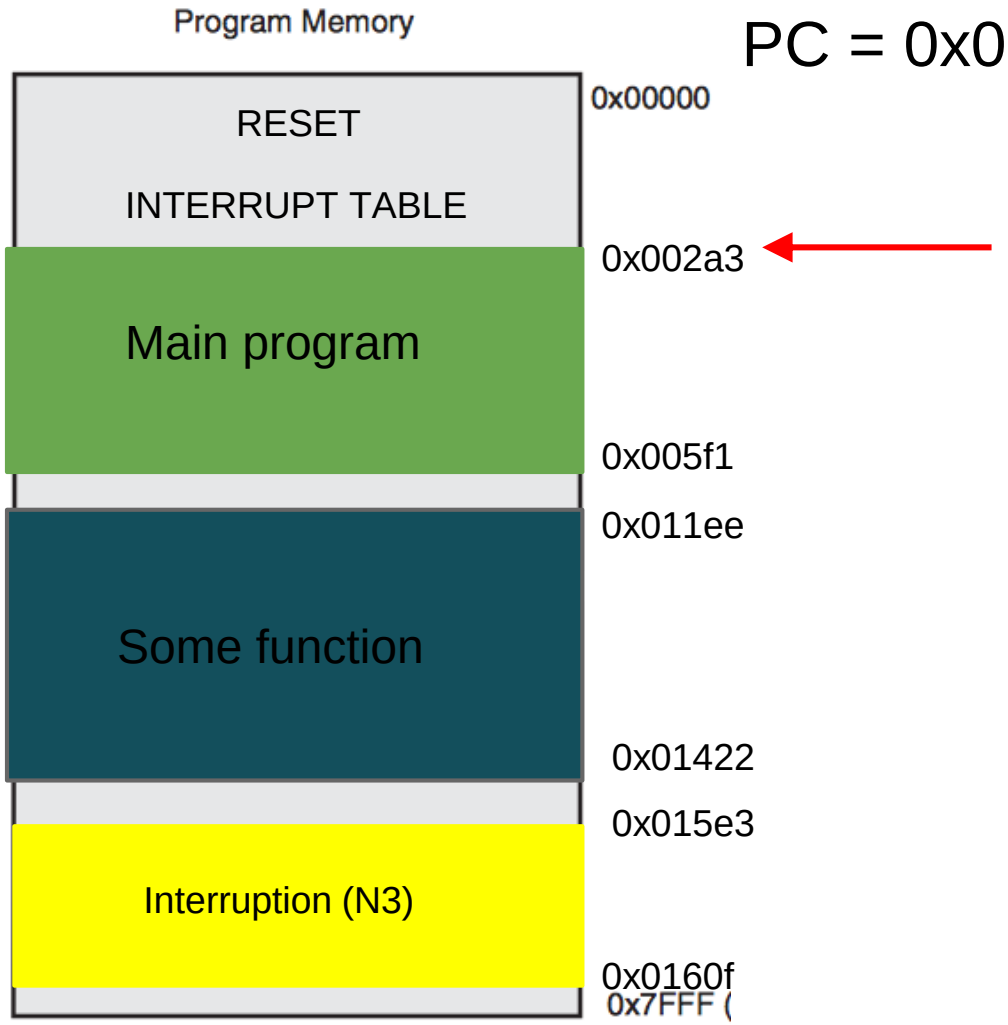
Fonction appelé avec:
PC = 0x0042c

Et avec des interruptions…

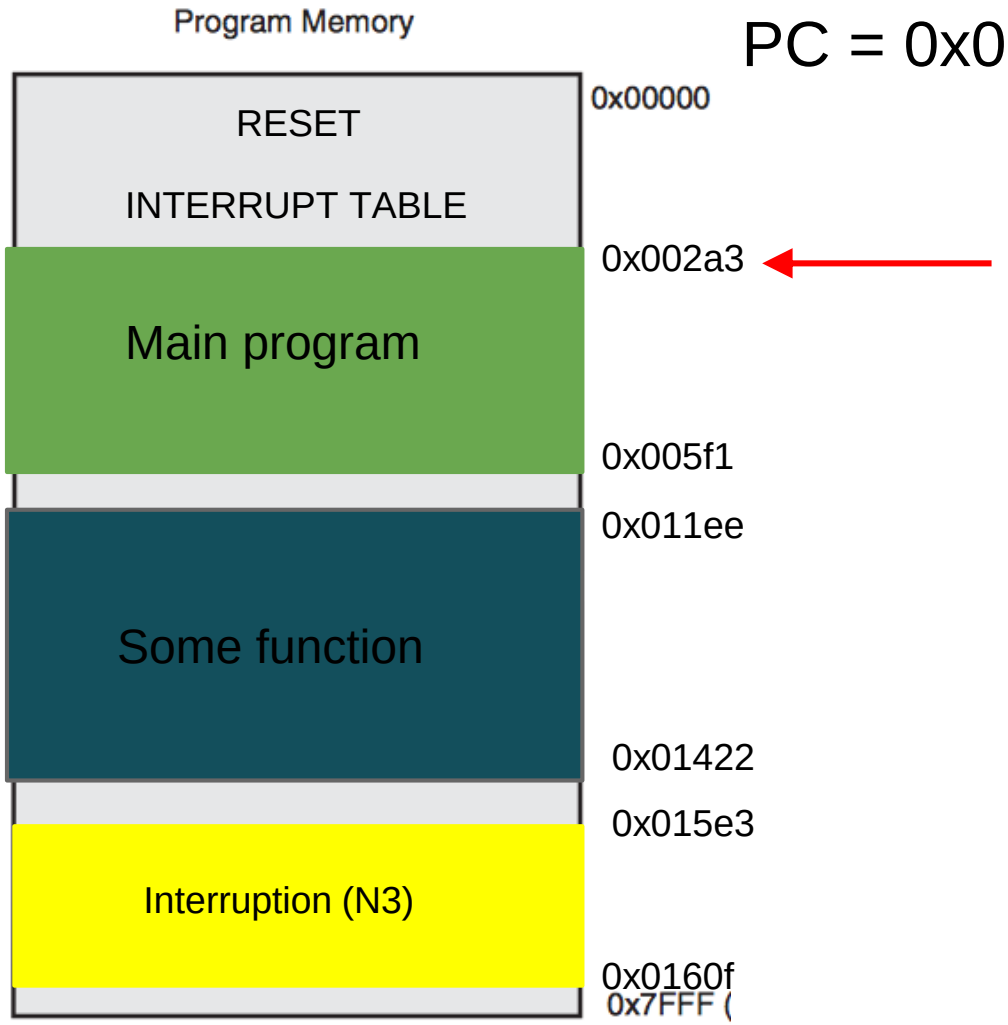
Memoire programme - Interruption!



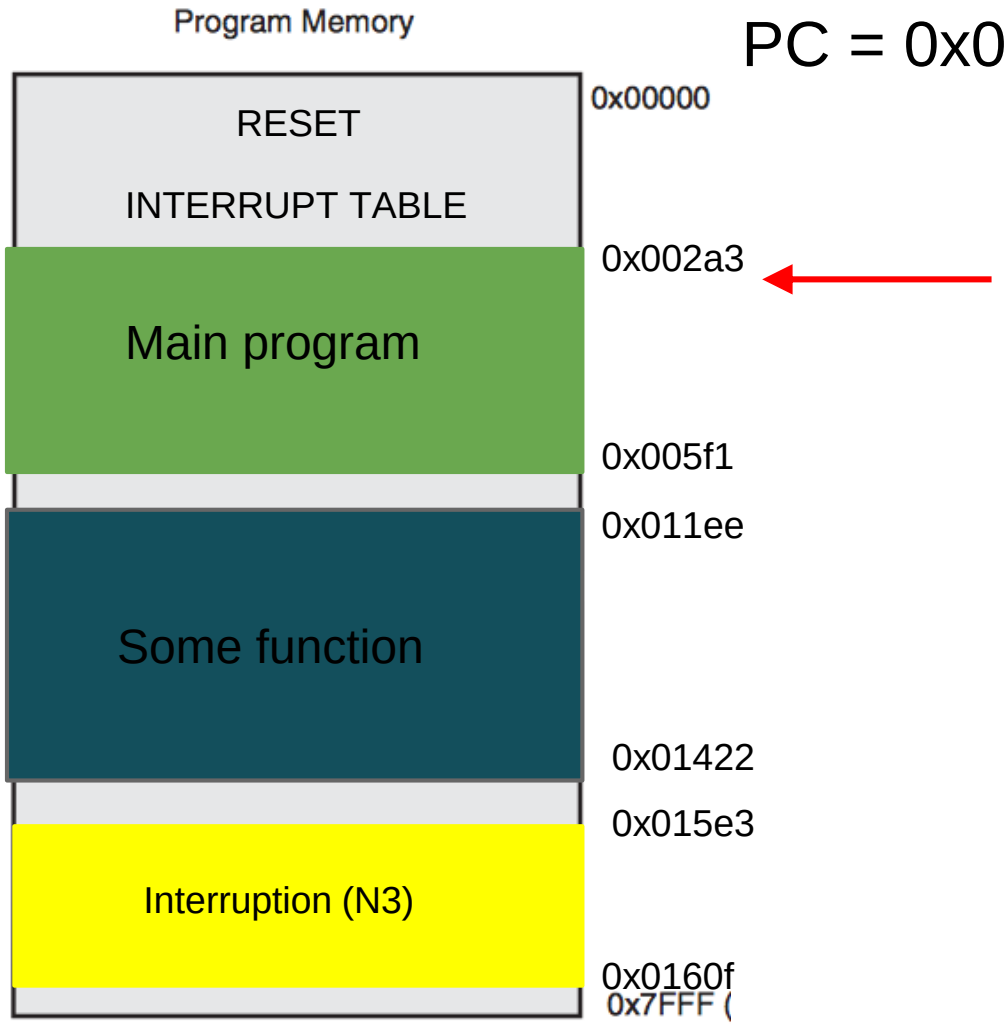
Memoire programme - Interruption!



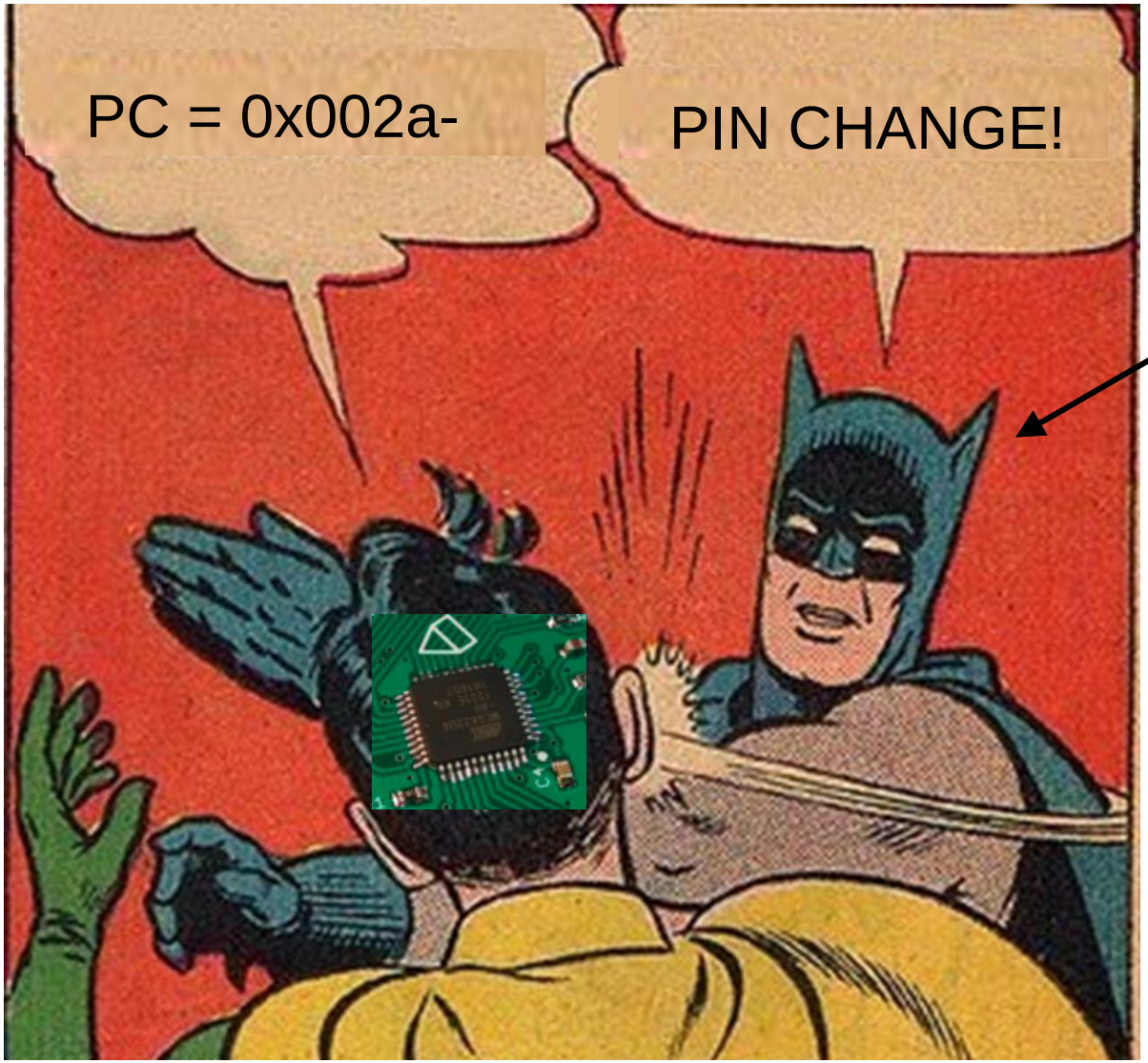
Memoire programme - Interruption!



Memoire programme - Interruption!



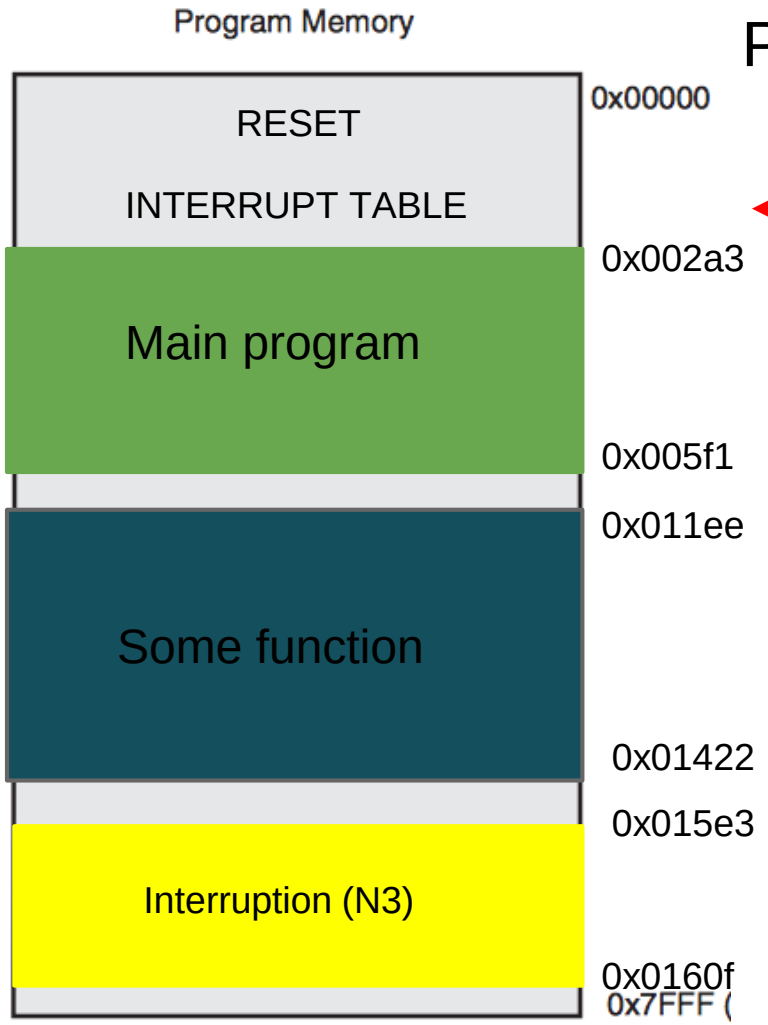
Memoire programme - Interruption!



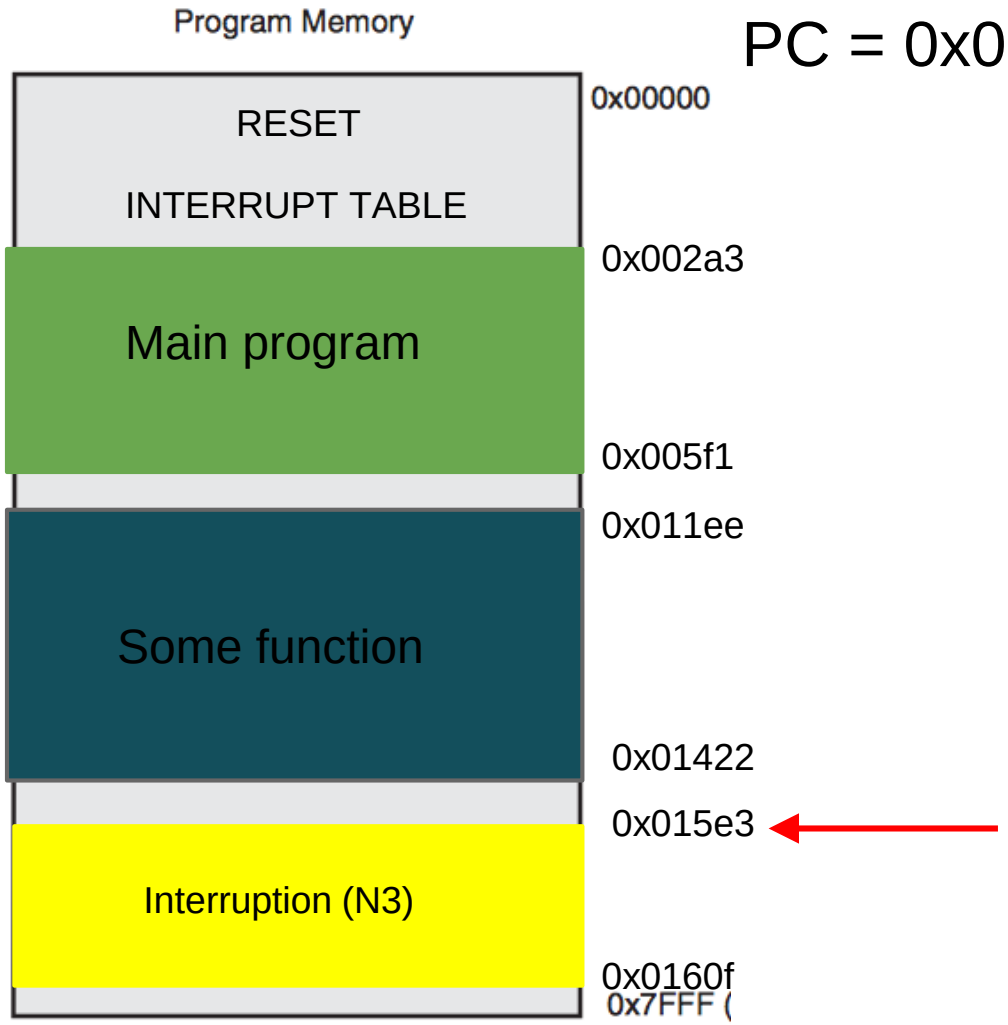
External World

Dans ce cas, on admet c'est l'interruption numero 3

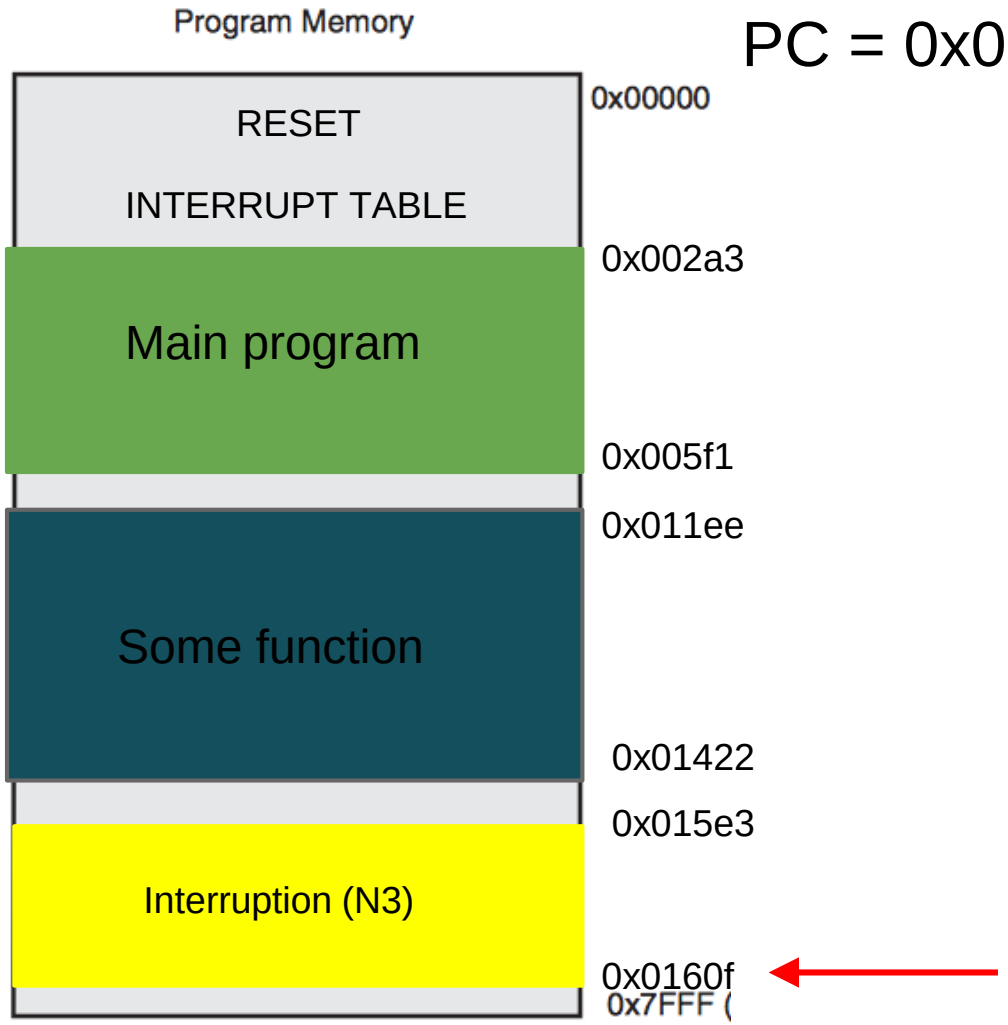
Memoire programme - Interruption!



Memoire programme - Interruption!



Memoire programme - Interruption!



Memoire programme - Interruption!