Université IBN TOFAIL – Kénitra
ECOLE NATIONALE DES SCIENCES APPLIQUEES

Cours Microcontrôleurs Avancés

CH7: Le Convertisseur Analogique Numérique ADC

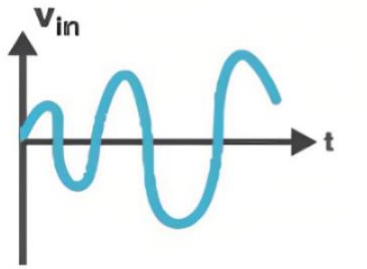
Par:

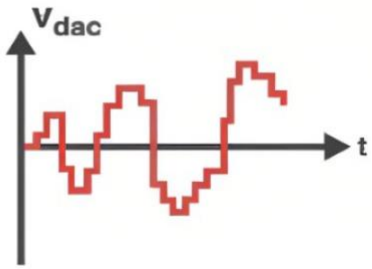
Hassan EL FADIL

elfadil.h@gmail.com

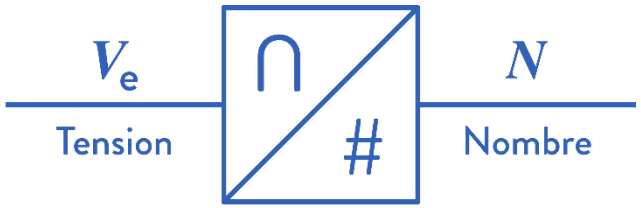
Définition

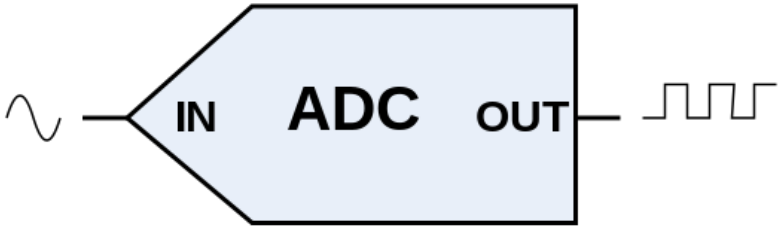
Un **convertisseur analogique-numérique** (CAN, parfois **convertisseur A/N**, ou en anglais **ADC** pour *Analog to Digital Converter*) est un dispositif électronique dont la fonction est de traduire une grandeur analogique en une valeur numérique codée sur plusieurs bits.





Symboles:





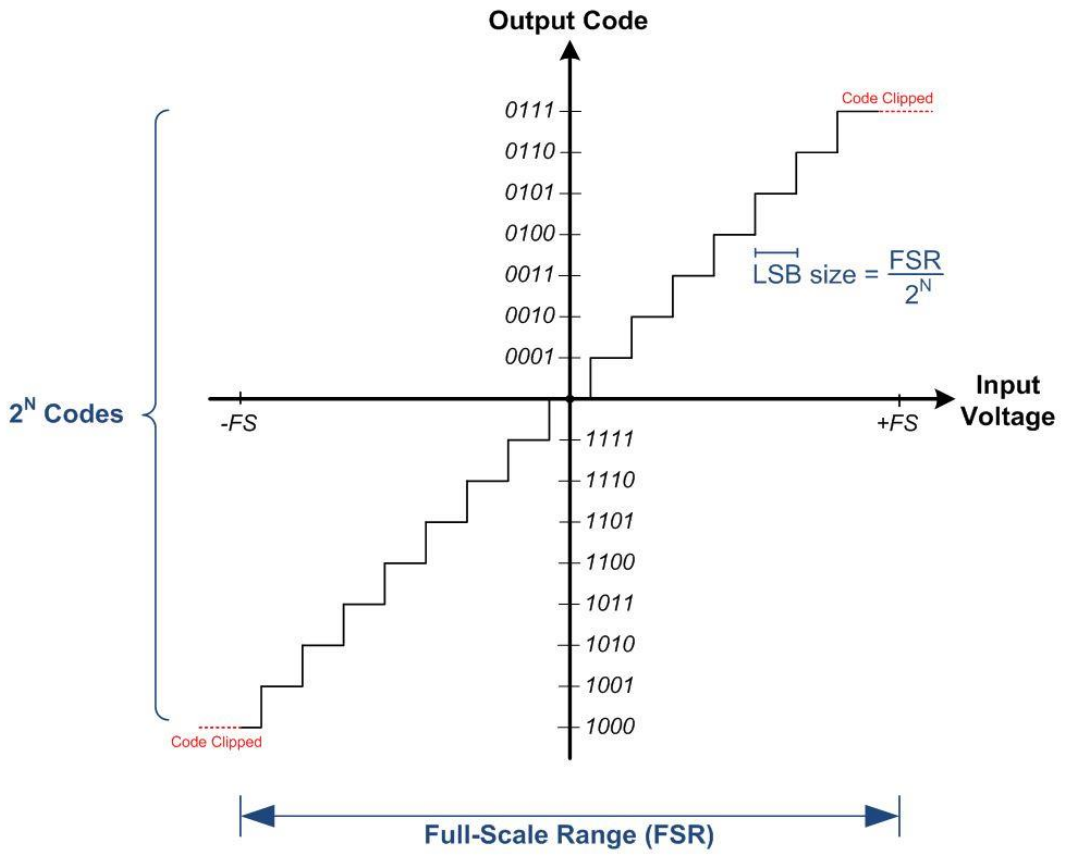
Résolution et Quantum

- **La valeur Pleine échelle** (PE) : Full-Scale voltage (also called 'span'):

$$V_{PE} = V_{max} - V_{min}$$

- **La résolution**: c'est le nombre de bits N du CAN.
- **Le quantum** (LSB voltage), noté **q**, correspond à la quantité élémentaire de variation du signal analogique pour une variation d'un bit:

$$q = LSB = \frac{V_{PE}}{2^N}$$

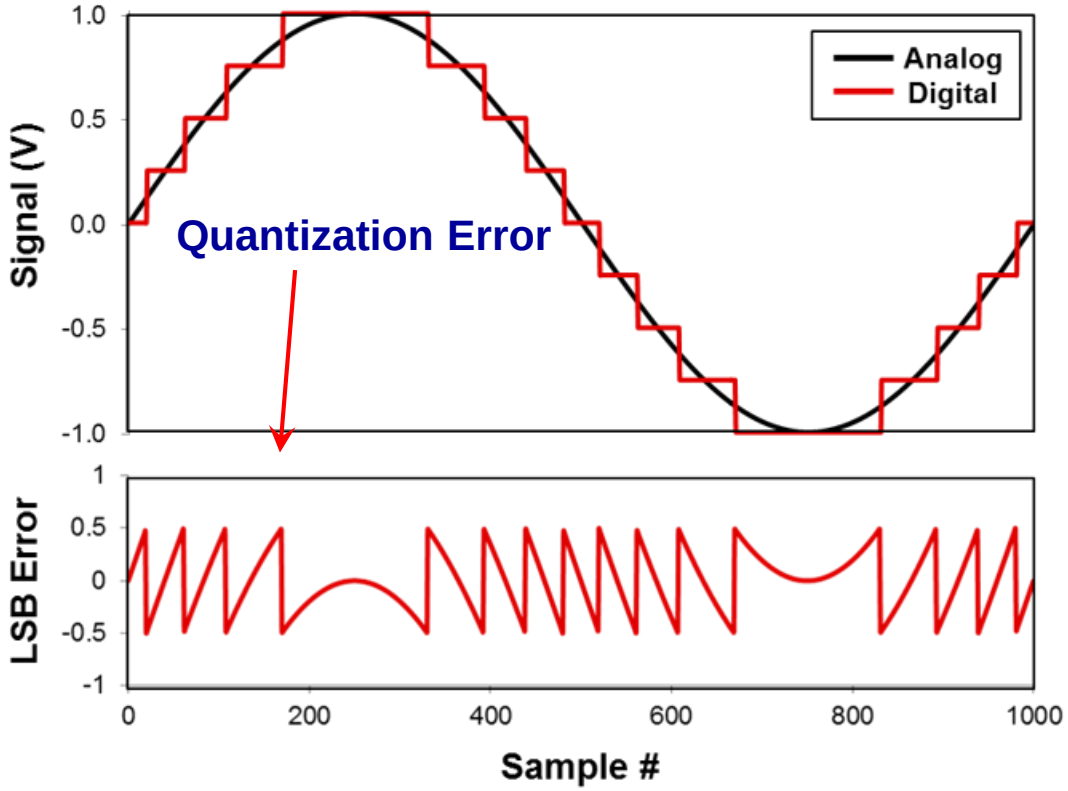


Erreur de Quantification:

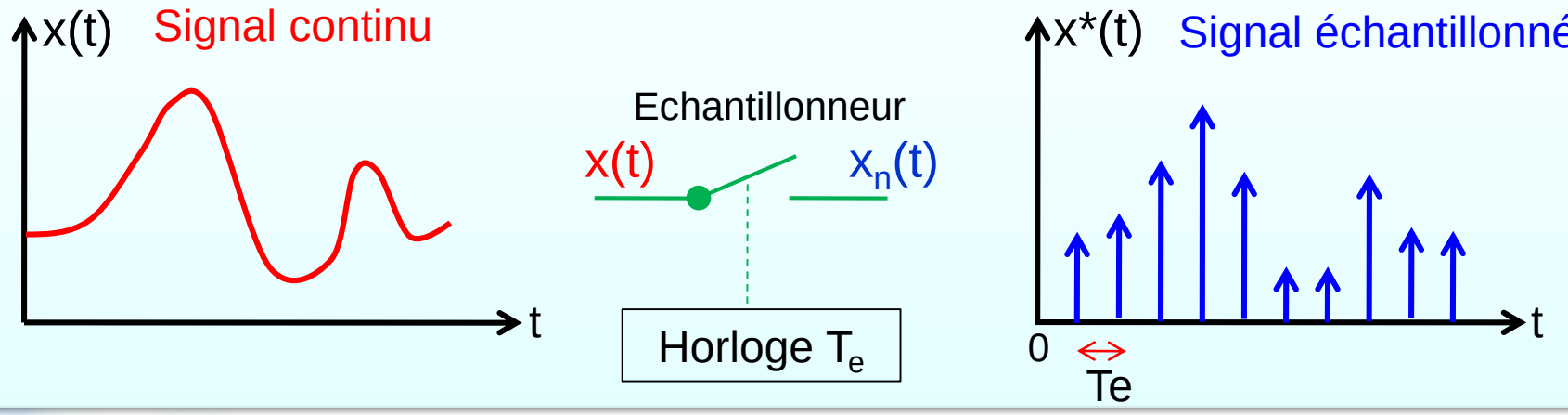
- L'erreur maximale de quantification (Quantization Error) est:

$$\varepsilon = \pm\frac{1}{2}q = \pm\frac{1}{2}LSB$$

- Exemple: **ADC 3 bits** avec une plage de tension [**0, 5V**]:
  - Quantum: $q = \frac{5}{8} = 0.625V$
  - Erreur de Quantification: $\varepsilon = \pm 0.3125V$



Fréquence d'échantillonnage:

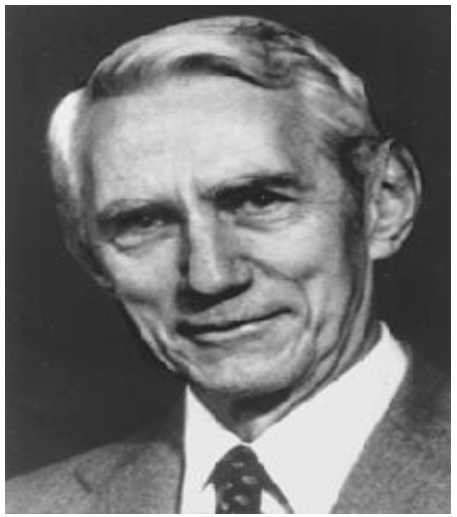


Théorème de Shannon: Pour pouvoir reconstituer un signal continu à partir d'un train d'échantillons de période T_e, on choisit F_e ($\boldsymbol{F_e = 1/T_e}$) telle que:

$$\boldsymbol{F_e > 2f_{max}}$$

$\boldsymbol{f_{max}}$: Fréquence maximale du signal à échantillonner

En pratique, on choisit $\boldsymbol{F_e = 5 \text{ à } 10 \times f_{max}}$

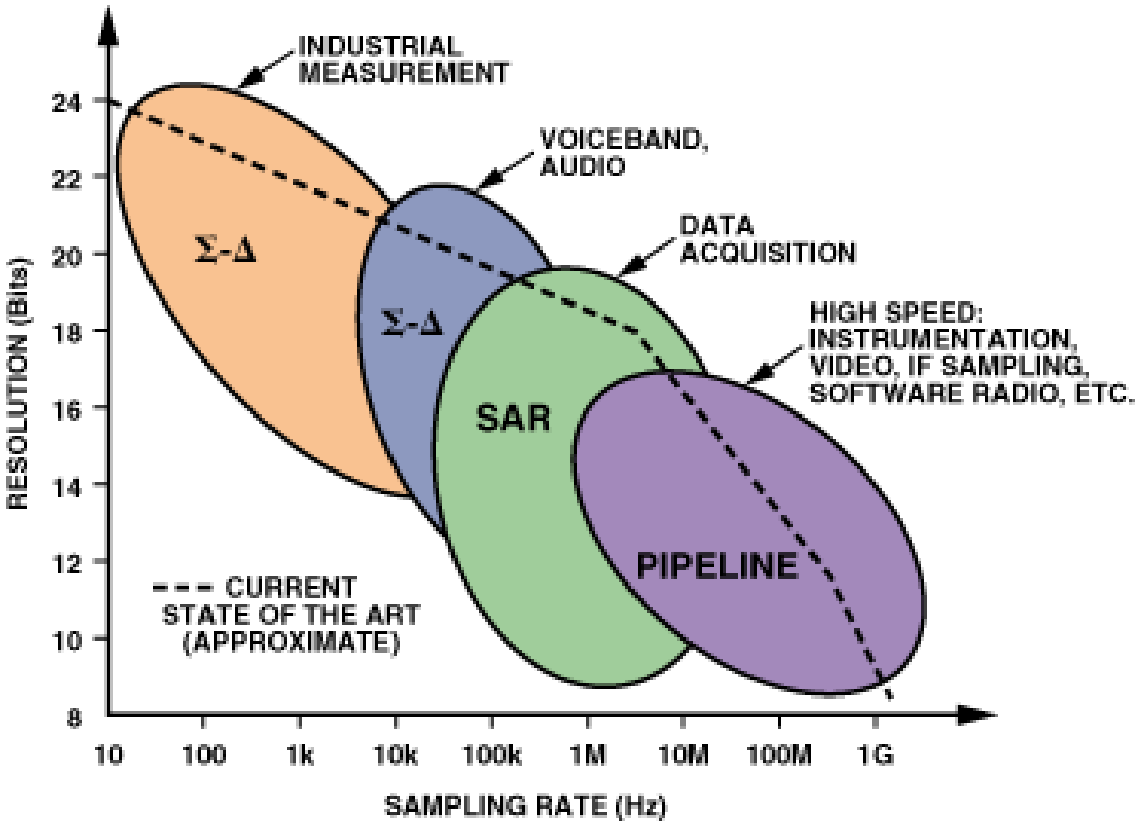


Claude Shannon

Claude Elwood Shannon (1916 - 2001) est un ingénieur en génie électrique et mathématicien américain. Il est l'un des fondateurs, de la théorie de l'information.

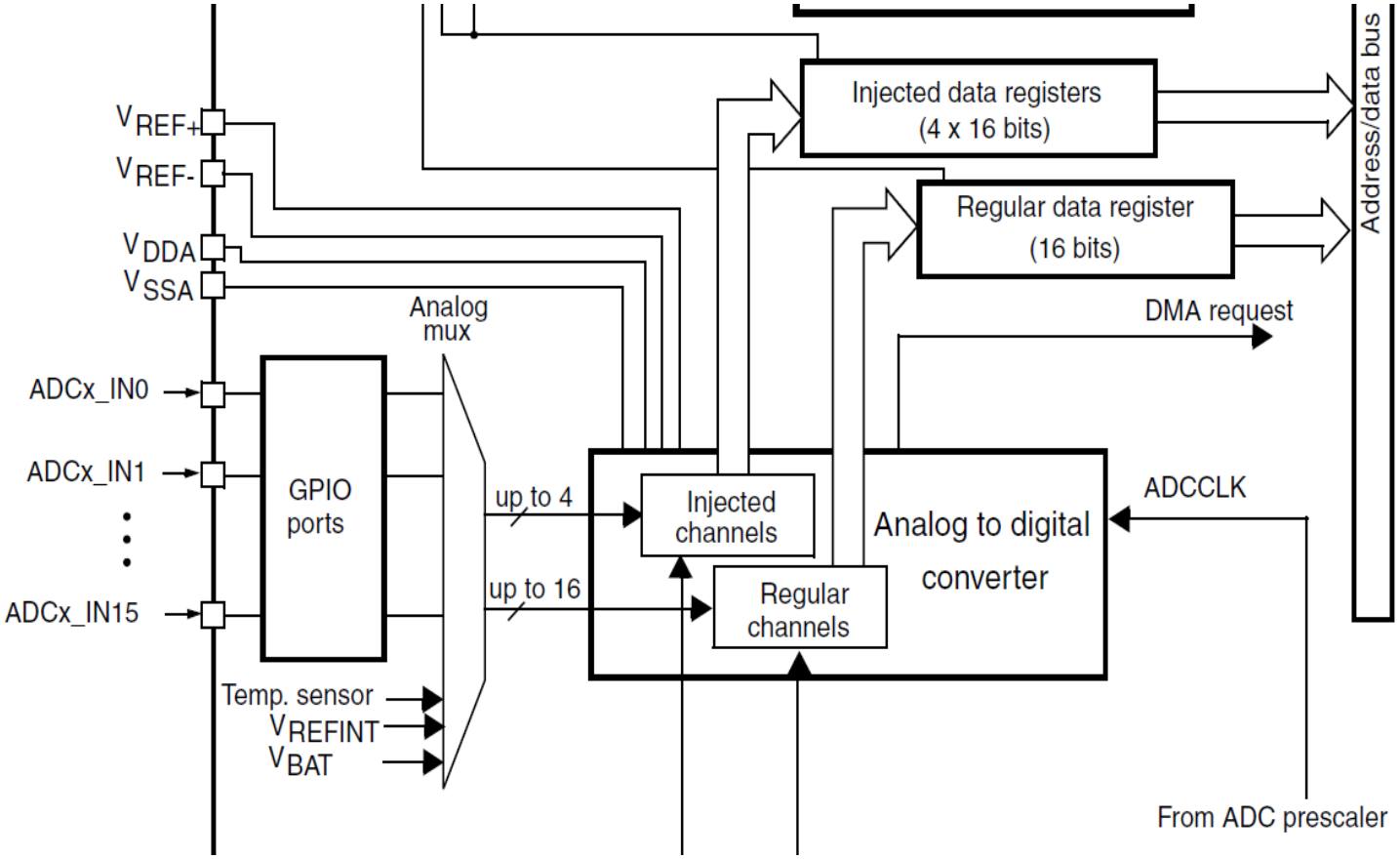
Les Architectures des ADC

- **CAN à approximation successive**: Successive-Approximation ADC (SAR):
  - pour acquisition de données
  - Très utilisés dans le contrôle, le monitoring, analyse faible et moyennes fréquences
  - Moyenne fréquence et moyenne résolution
- **CAN Sigma-Delta (Σ-Δ)**:
  - Pour la mesure et l'instrumentation industrielles de précision, la voix et l'audio
  - Faible fréquence et bonne résolution
- **Pipelined ADC** :
  - Pour les applications à grande vitesse (taux d'échantillonnage supérieurs à 5 MSPS: Megasamples per second)
  - Mais moyenne résolution



L'ADC du STM32F401RE

- **L'ADC 12** bits du STM32F401RE est un CAN à approximations successives.
- Résolution configurable 12 bits, 10 bits, 8 bits ou 6 bits
- Il dispose de jusqu'à 19 canaux multiplexés lui permettant de mesurer les signaux provenant de 16 sources externes, de deux sources internes et du canal V_{BAT}.
- La conversion A/N des canaux peut être effectuée en mode simple, continu, scan ou discontinu.
- Le résultat de l'ADC est stocké dans un registre de données de 16 bits.

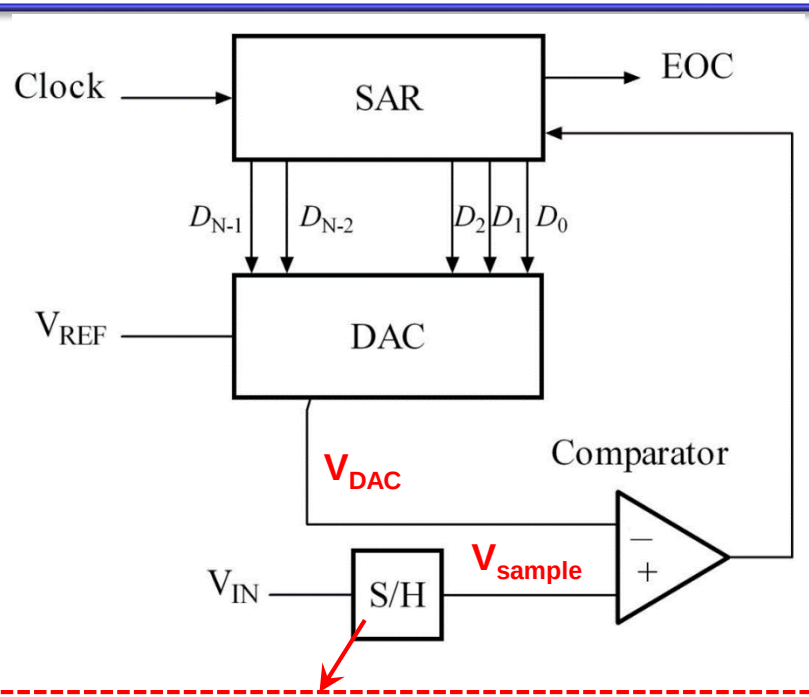


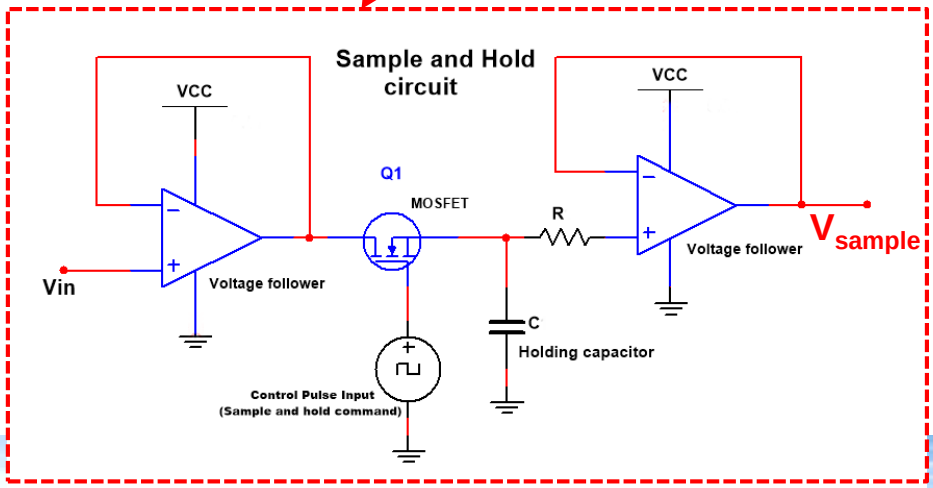
L'ADC du STM32F401RE

L'ADC SAR:

Le circuit CAN à approximations successives se compose généralement de quatre sous-circuits principaux :

1) Un circuit échantillonneur-bloqueur pour acquérir la tension d'entrée Vin.
2) Un comparateur de tension analogique qui compare Vin à la sortie du DAC interne et transmet le résultat de la comparaison au registre d'approximations successives (SAR).
3) Un sous-circuit de registre d'approximations successives conçu pour fournir un code numérique approximatif de Vin au DAC interne.
4) Un DAC de référence interne qui fournit au comparateur une tension analogique égale à la sortie du code numérique du SAR.





L'ADC du STM32F401RE

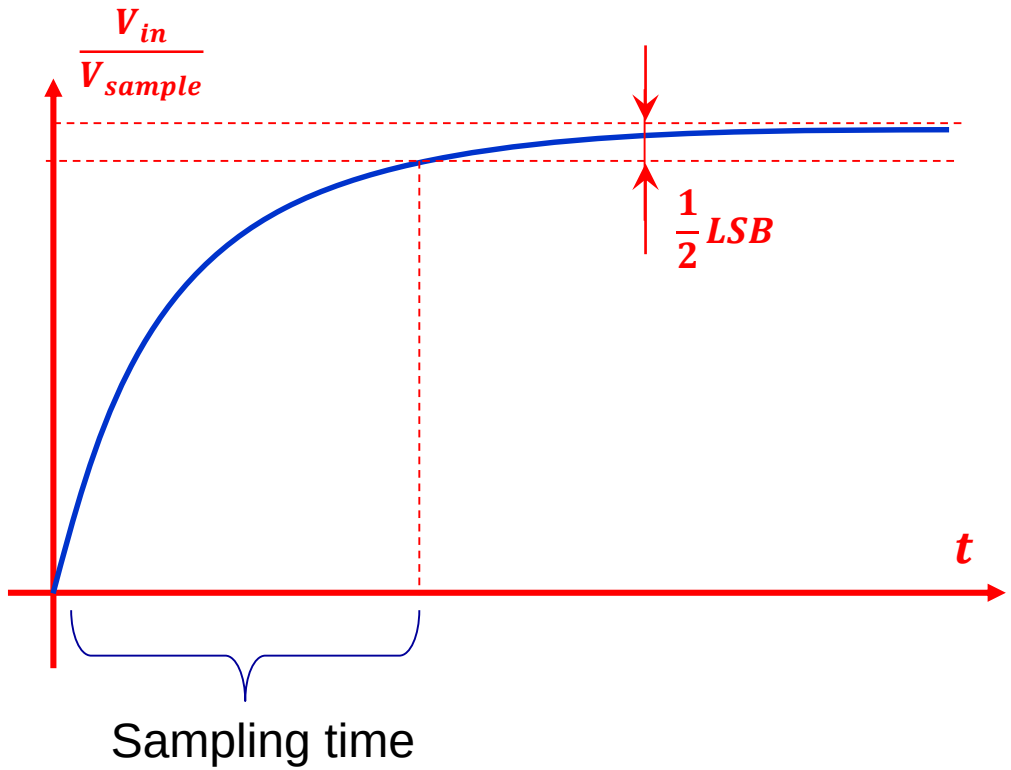
L'ADC SAR: Phase d'échantillonnage-Blocage

- Quand le transistor et ON (prise de l'échantillon), la tension V_{sample} évolue comme suit :

$$V_{sample} = V_{in} \times \left(1 - e^{-\frac{t}{\tau}}\right) \qquad \tau = R_{on}C$$

R_{on} : Résistance Ron du Transistor

- Le temps d'échantillonnage est programmé (ADC_SMPR1 and ADC_SMPR2). C'est le temps du fermeture de transistor.
- Il doit être choisi de telle sorte que l'erreur de prise d'échantillon soit inférieure au ½ LSB. Ceci est fondamental pour assurer une conversion de précision.
- Quand le transistor est OFF, le condensateur maintient (Hold) la valeur prise de la tension d'entrée jusqu'au nouveau échantillon. Ce temps doit être supérieur au temps de conversion du CAN.

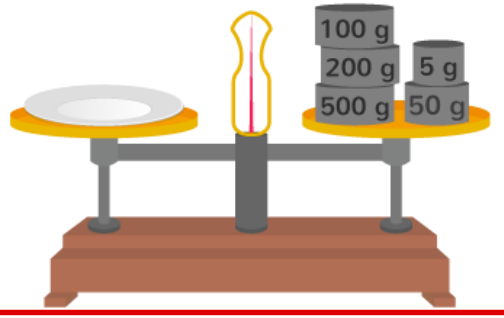


L'ADC du STM32F401RE

L'ADC SAR: Phase de Conversion

- Après la fin de la phase d'échantillonage-blocage ait terminée, la phase de conversion commence.
- Illustrons cette phase dans le cas suivant:
  N = 4bits, V_{ref} = 1V et V_{sample}= 0.69V :

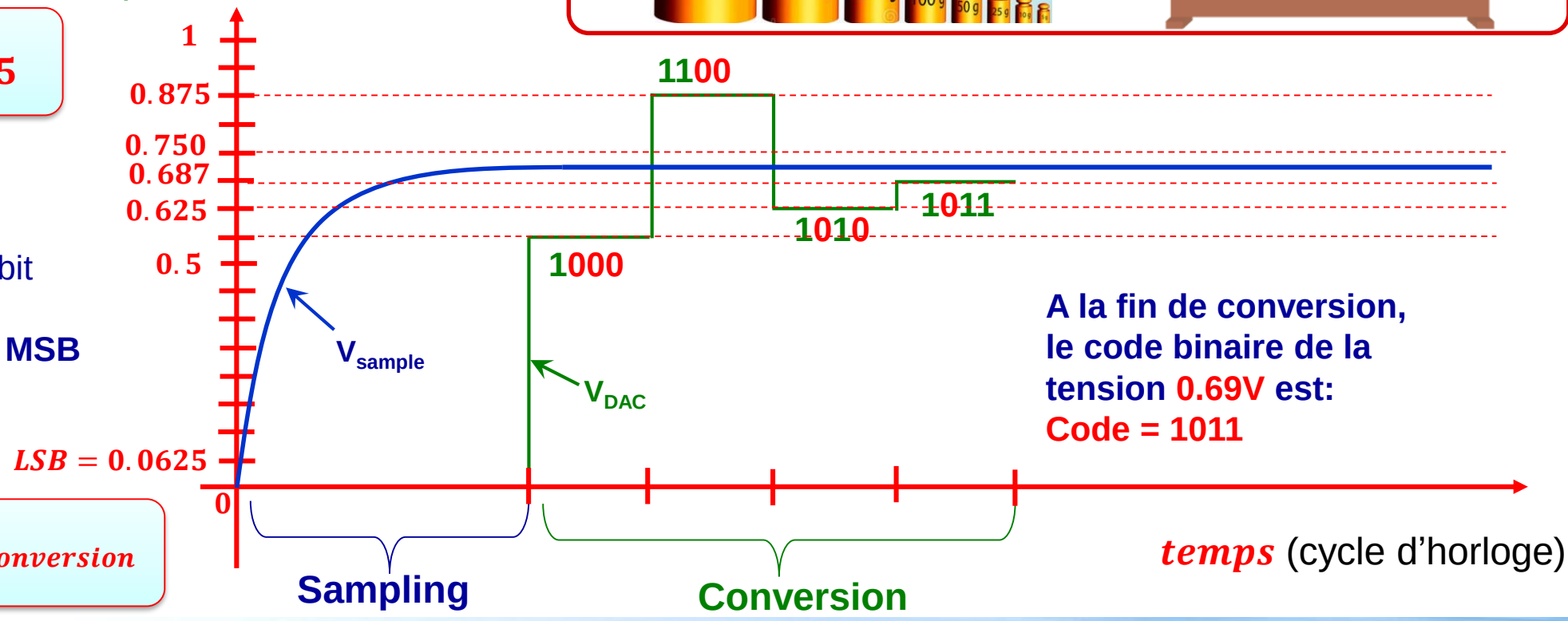
Exemple intuitif: balance pour mesurer une masse



$$LSB = \frac{1}{2^4} = 0.0625$$

- La conversion dure N cycles (ici 4 cycles)
- Pour chaque cycle un bit est déterminé, en commençant par le bit **MSB**
- Le temps total de conversion est:

$$t_{ADC} = t_{sampling} + t_{conversion}$$

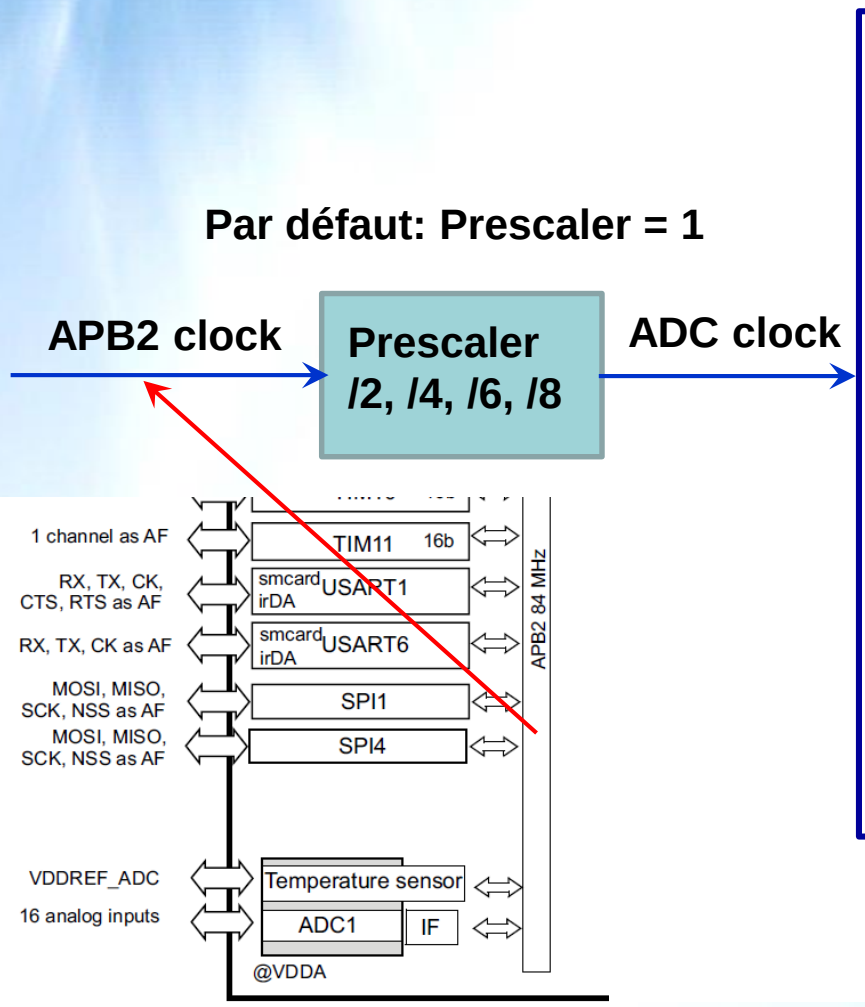


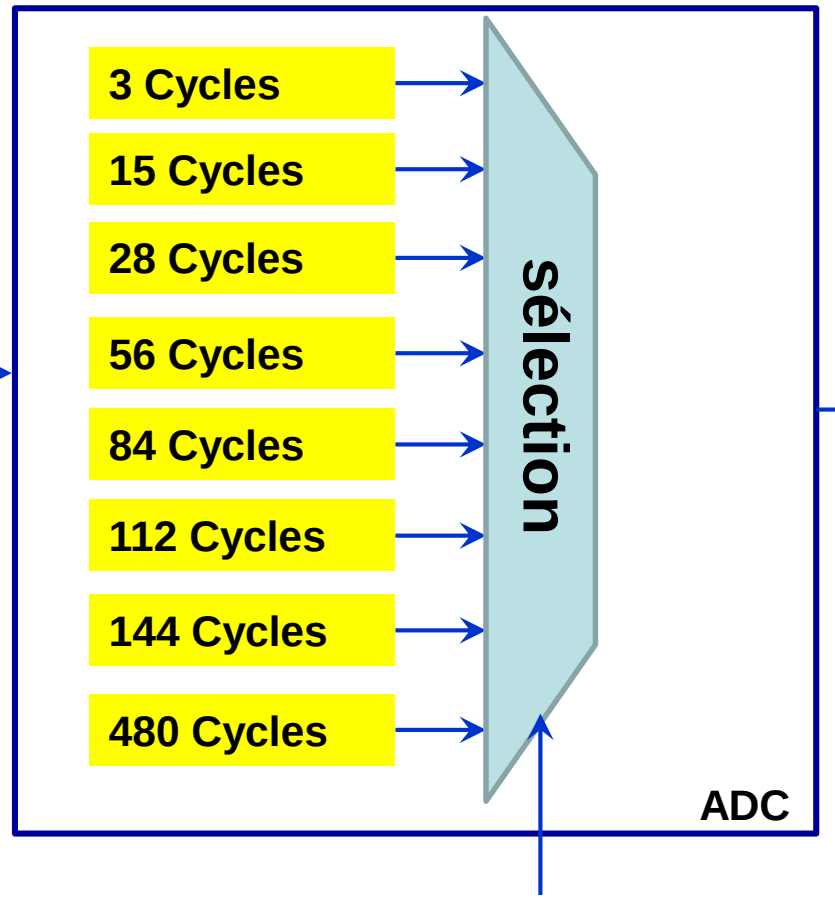
A la fin de conversion, le code binaire de la tension 0.69V est: Code = 1011

L'ADC du STM32F401RE

Programmation de la période d'échantillonnage pour un canal

Par défaut: Prescaler = 1





L'ADC contient 16 canaux.
La période d'échantillonnage de chaque canal peur être programmée indépendamment des autres.

Période d'échantillonnage du canal

Bits SMPx[2:0]:

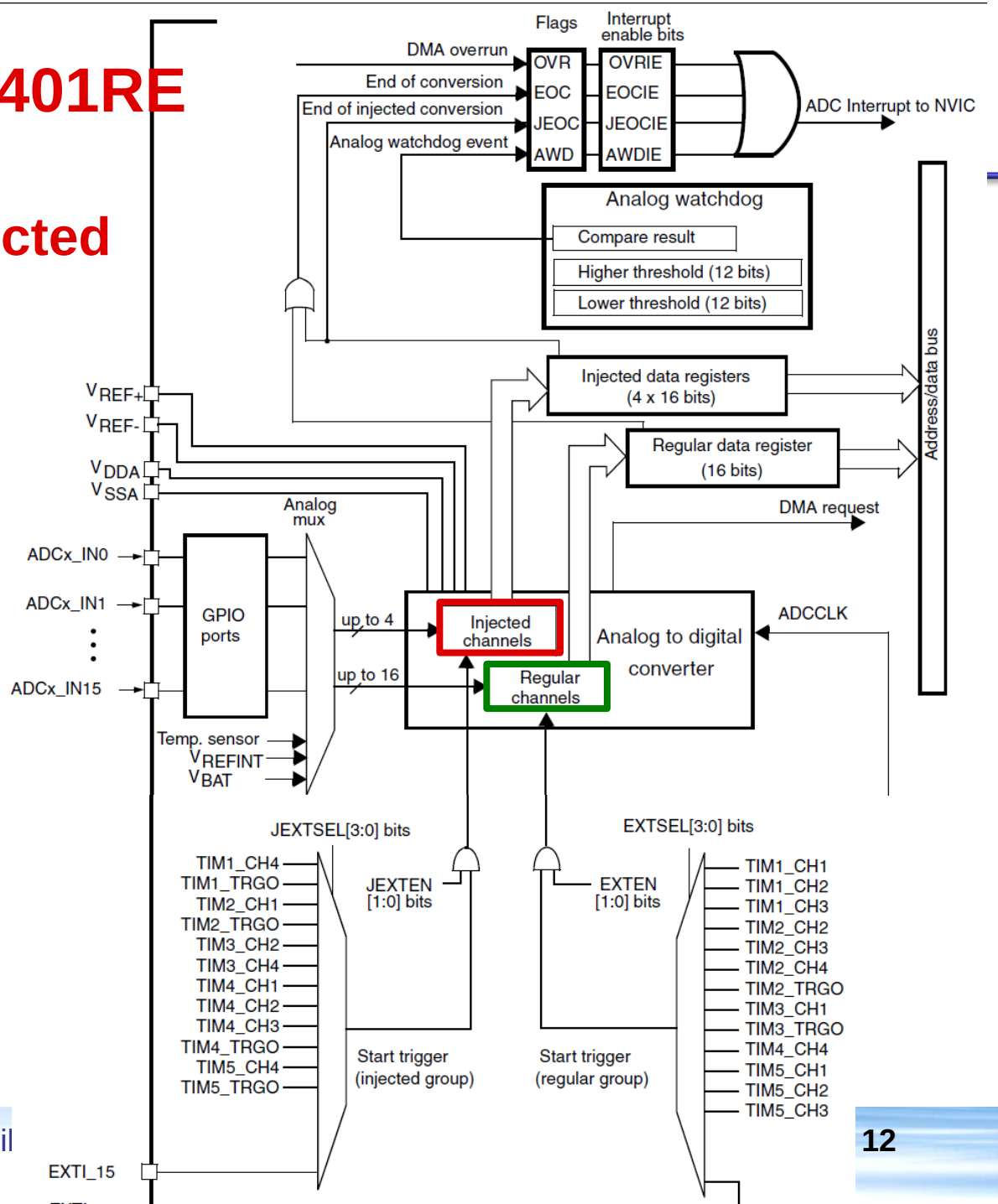
000: 3 cycles
001: 15 cycles
010: 28 cycles
011: 56 cycles
100: 84 cycles
101: 112 cycles
110: 144 cycles
111: 480 cycles

ADC Sample time Registers
Bits SMPx[2:0]: des registres ADC_SMPR1 et ADC_SMPR2

L'ADC du STM32F401RE

Canal Régulier vs. Injecté: Regular vs. Injected

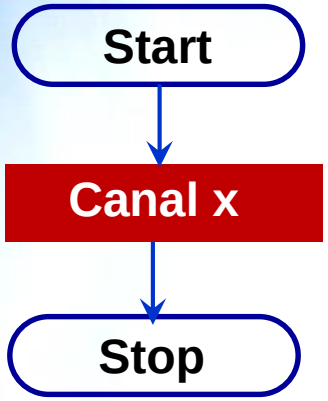
- Vous pouvez configurer l'ADC pour lire une séquence de canaux en boucle. Ces chaînes sont régulièrement converties (Regular).
- En mode injecté, la conversion est déclenchée par un événement externe ou par un logiciel (Injected).
- Une conversion injectée a une priorité plus élevée par rapport à une conversion « régulière » et interrompt ainsi les conversions régulières.



L'ADC du STM32F401RE

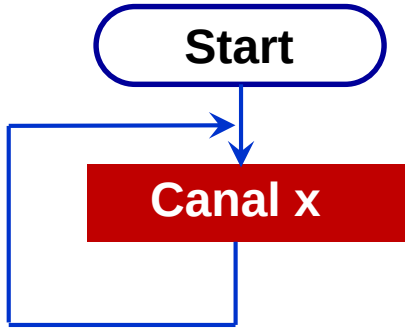
Modes de Conversion:

Mode un seul canal: « Single Channel Mode»



Mode un seul canal, simple conversion
« Single Channel, Single Conversion Mode »

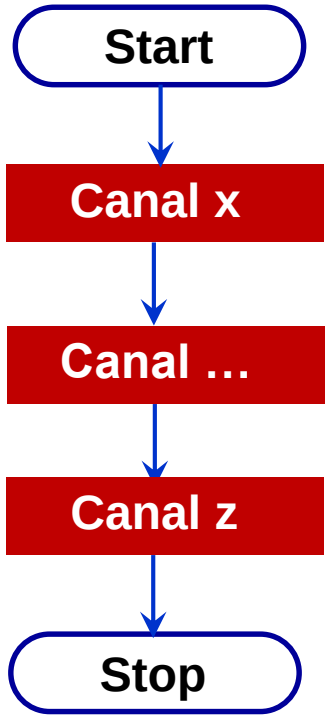
SCAN bit in ADC_CR1 = 0
CONT bit in ADC_CR2 = 0



Mode un seul canal, conversion continue
« Single Channel, Continuous Conversion Mode »

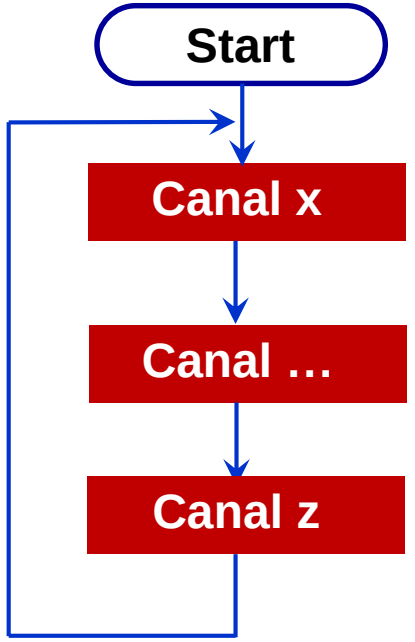
SCAN bit in ADC_CR1 = 0
CONT bit in ADC_CR2 = 1

Mode balayage: « Scan Mode »



Mode balayage, simple conversion
« Scan Mode with Single Conversion »

SCAN bit in ADC_CR1 = 1
CONT bit in ADC_CR2 = 0

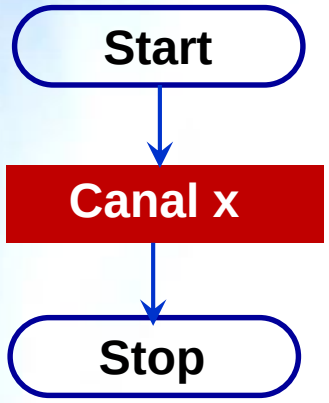


Mode balayage, conversion continue
« Scan Mode with Continuous Conversion »

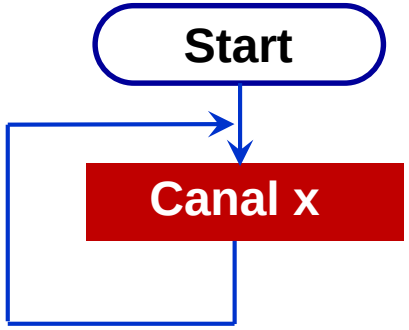
SCAN bit in ADC_CR1 = 1
CONT bit in ADC_CR2 = 1

L'ADC du STM32F401RE

Modes de Conversion:



Mode simple conversion



Mode conversion continue

Mode	Utilisation
Simple Conversion	Lorsque vous avez besoin d'une seule mesure, de temps en temps, sur demande: • Mesure de la tension d'une batterie, • Mesure de température, etc.
Conversion Continue	Lorsque vous devez mesurer en permanence, comme pour un graphique en temps réel: • Mesure de la vitesse de rotation d'un moteur • Numérisez un signal audio, etc.

L'ADC du STM32F401RE

Modes de Conversion:

Mode discontinu: « Discontinuous mode »

Le mode discontinu (**Discontinuous mode**), fait référence à une configuration dans laquelle la conversion analogique-numérique est effectuée de manière intermittente, plutôt que de manière continue.
Dans le contexte des ADC STM32, le mode discontinu permet de réaliser des conversions périodiques, avec des pauses entre chaque conversion. Cela peut être utile dans des applications où une conversion fréquente n'est pas nécessaire, et où l'économie d'énergie est une considération importante

Ce mode est activé en définissant le bit DISCEN ou JDISCEN dans le registre ADC_CR1

Discontinuous mode disabled : DISCEN = 0: for regular channels, JDISCEN = 0 for Injected Channels
Discontinuous mode enabled : DISCEN = 1: for regular channels, JDISCEN = 1 for Injected Channels

L'ADC du STM32F401RE

Chien de Garde Analogique: « Analog Watchdog »:

L'Analog Watchdog (**AWD**) est une fonctionnalité des ADC présente dans les microcontrôleurs STM32. L'AWD est utilisé pour surveiller la tension analogique d'entrée et générer une interruption ou une action spécifique lorsque cette tension dépasse ou ne respecte pas une certaine plage définie.

Voici comment fonctionne généralement l'Analog Watchdog dans les ADC des STM32 :

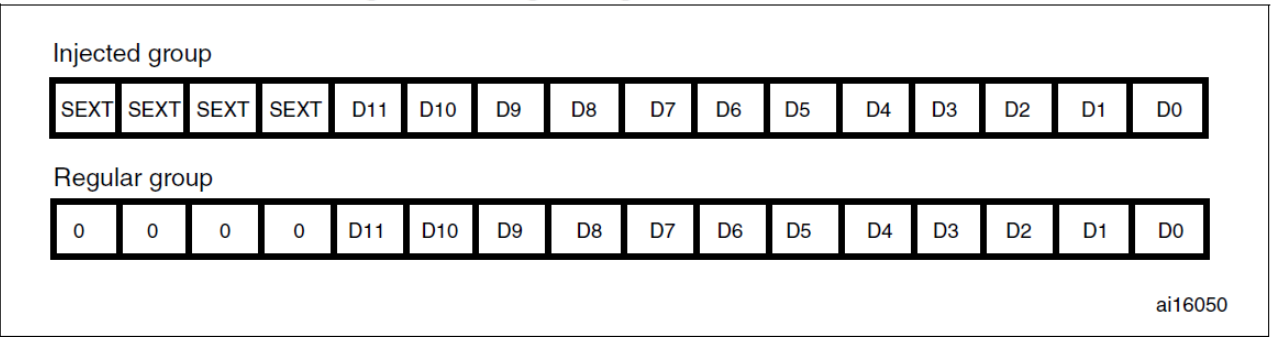
- **Configuration de la plage de tension :** Vous spécifiez une plage de tension supérieure (limite haute) et une plage de tension inférieure (limite basse) que vous souhaitez surveiller. Utiliser les registres 16bits ADC_HTR and ADC_LTR.
- **Activation de l'Analog Watchdog :** Vous activez l'Analog Watchdog pour le ou les canaux ADC à surveiller. Utiliser les bits AWDSGL , AWDEN et JAWDEN du registre ADC_CR1
- **Génération d'interruptions ou d'actions :** Lorsque la tension analogique d'entrée dépasse la plage définie par les limites haute ou basse, l'AWD peut générer une interruption, déclencher une action (par exemple, arrêter la conversion), ou déclencher une routine de gestion d'erreur.
- **Réinitialisation automatique ou manuelle :** Certains microcontrôleurs STM32 offrent la possibilité de réinitialiser automatiquement l'Analog Watchdog après chaque conversion ou de le réinitialiser manuellement via du code

L'ADC du STM32F401RE

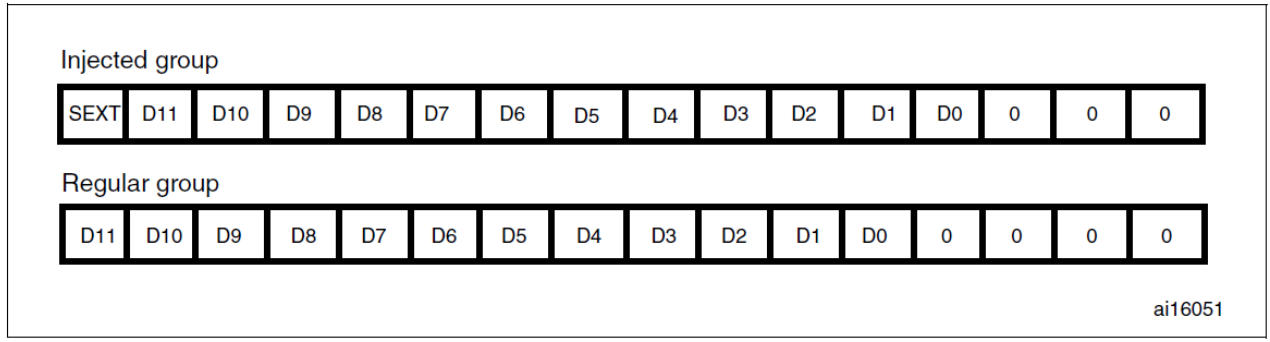
Alignement de données de conversion: « Data alignment »:

- Etant donné que l'ADC est de 12bits et le registre de résultat est de 16 bits, il se pose alors le problème d'alignement de données (soit à droite soit à gauche).
- Le bit ALIGN du registre ADC_CR2 sélectionne l'alignement des données stockées après conversion.
  - ALIGN = 0: Alignement à droite (Right alignment)
  - ALIGN = 1: Alignement à gauche (Left alignment)

Right alignment of 12-bit data



Left alignment of 12-bit data

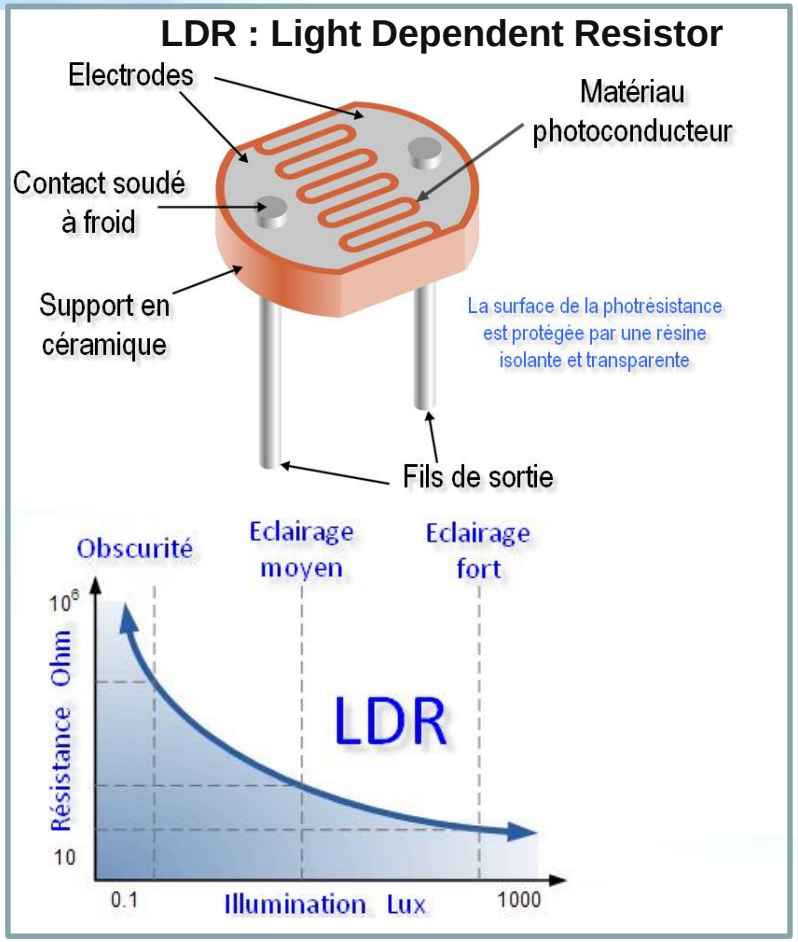


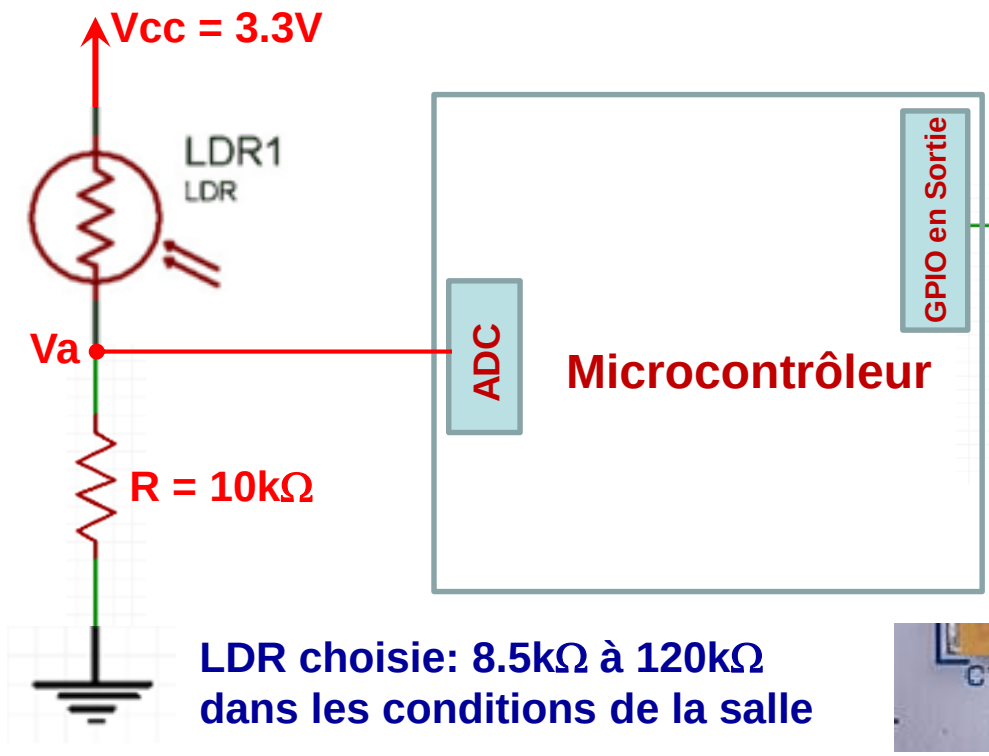
Le bit SEXT représente la valeur du signe (Sign EXTension).

Exemple d' Application:

Cahier des Charges:

- Mesurer l'intensité lumineuse d'une LDR par l'acquisition d'une tension analogique ;
- Eteindre ou allumer la LED LD2 selon le degré d'intensité mesurée:
  - **LDR éclairée: LED Eteinte**
  - **Obscurité: LED Allumée**





LDR choisie: 8.5kΩ à 120kΩ dans les conditions de la salle

Exemple d' Application:

Calculs préliminaires:

- Calculons la tension analogique V_a dans les deux conditions: LDR éclairée et LDR en obscurité:

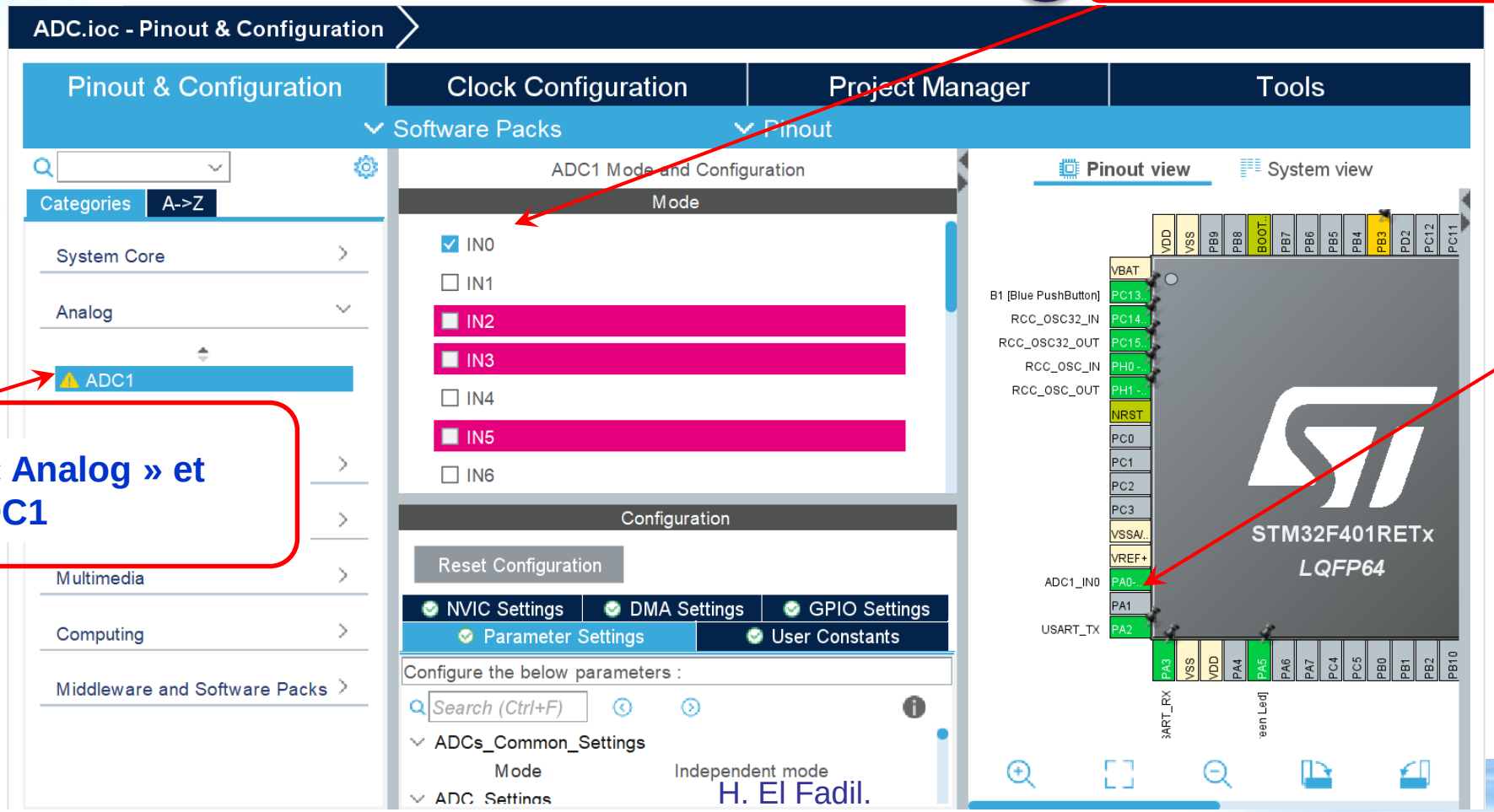
- $\boldsymbol{V_{a\_E} = V_{cc} \times \frac{R}{R+LDR} = 3,3 \times \frac{10}{10+8.5} = 1,784V}$
- $\boldsymbol{V_{a\_O} = V_{cc} \times \frac{R}{R+LDR} = 3.3 \times \frac{10}{10+120} = 0,254V}$
- Puisque le DAC est de 12 bits nous avons:
- $\boldsymbol{3,3\text{V} \rightarrow code: 2^{12} - 1 = 4095}$
- $\boldsymbol{V_{a\_E} = 1,784V \rightarrow code: 2213}$
- $\boldsymbol{V_{a\_O} = 0,254V \rightarrow code: 315}$

Un seuil de séparation entre LDR éclairée et en obscurité sera fixé entre ces deux valeurs: Exemple 2000

Configuration de l'ADC

1 On commence par créer un nouveau projet STM32CubeIDE (**Voir Chapitre 3**)
Appeler votre projet: **ADC**

3 **Ensuite, cocher IN0 (Canal 0)**



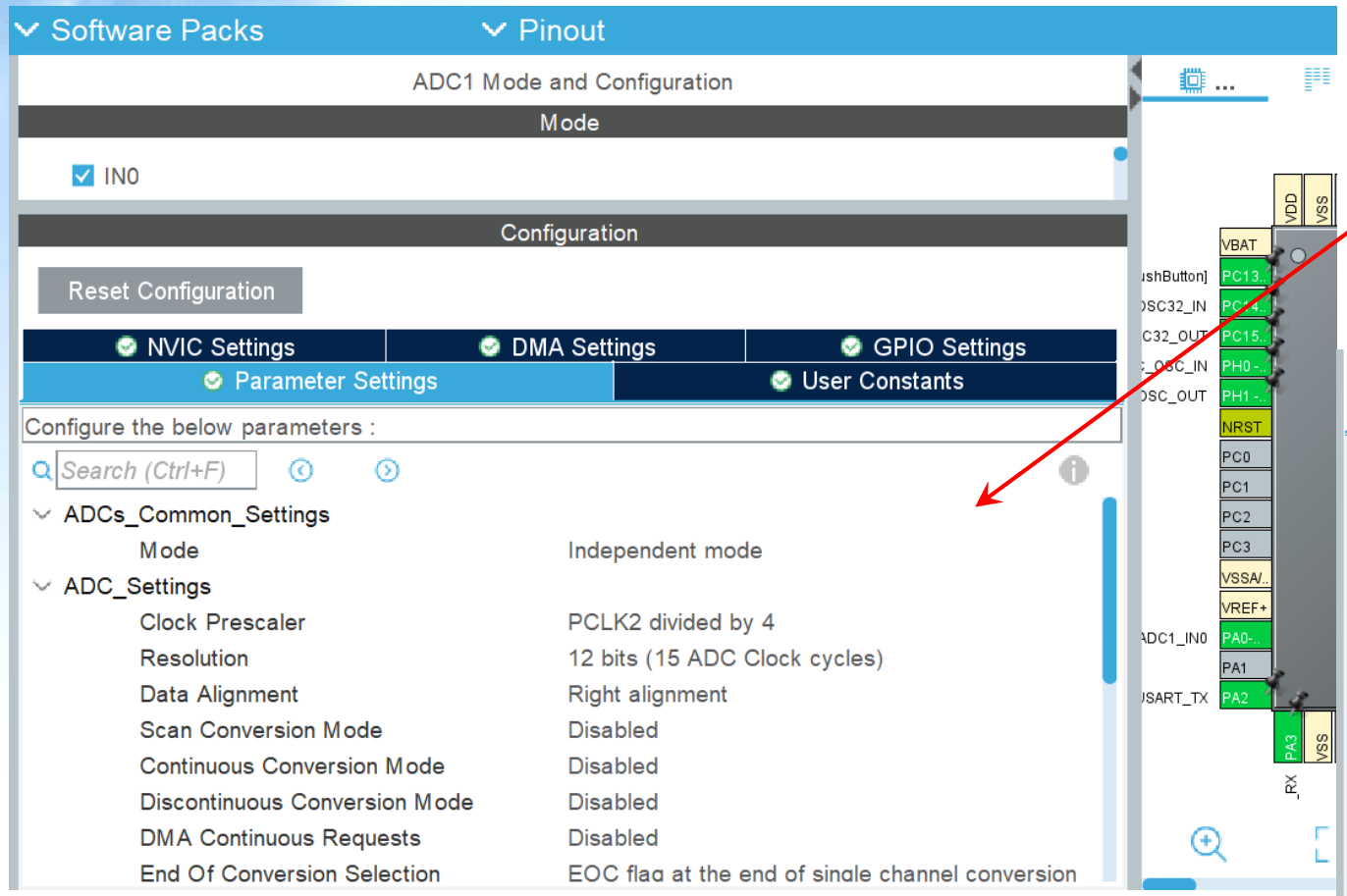
2 **Sélectionnez « Analog » et Cliquer sur ADC1**

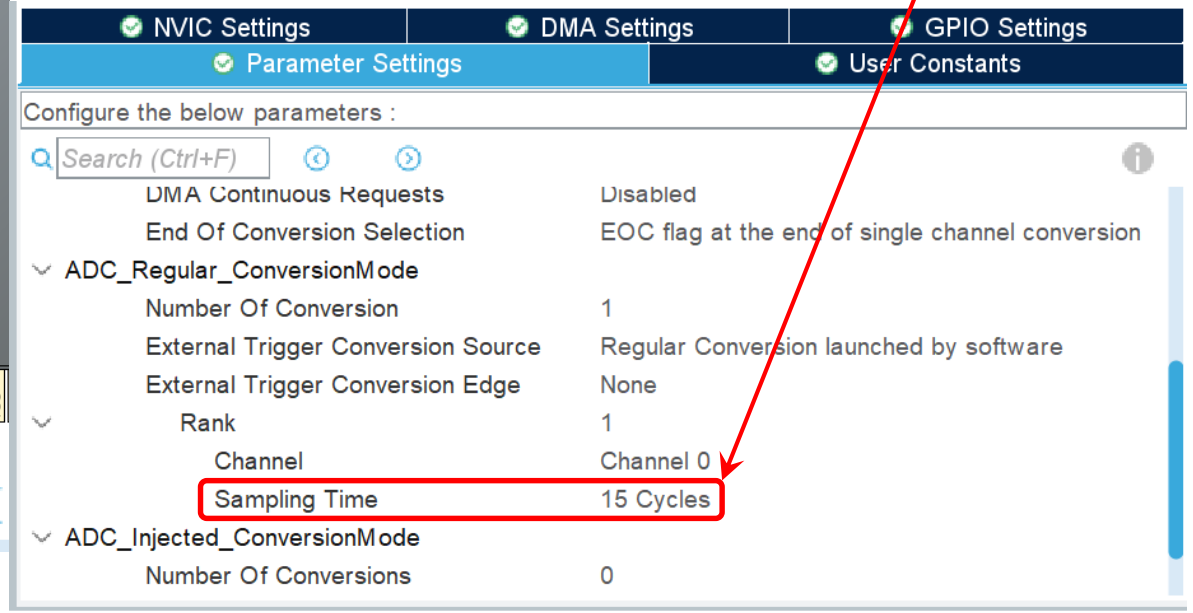
On peut constater ICI que l'entrée analogique du Canal 0 est affectée à la Pin PA0 du microcontrôleur

Configuration de l'ADC

4

Garder tous les paramètres de configuration par défaut à l'exception de « Sampling Time » : mettre 15 Cycles

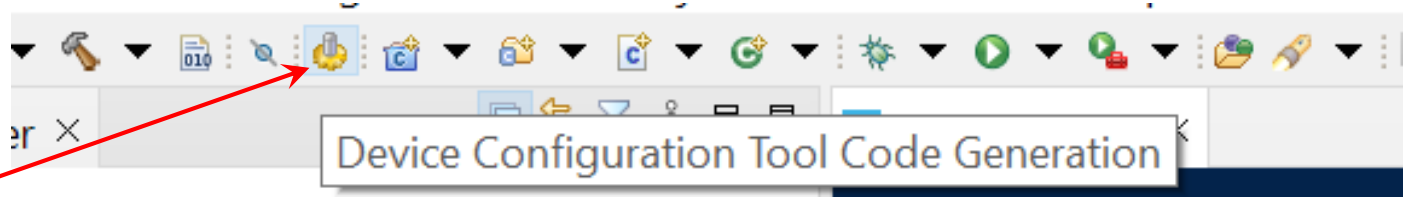




Génération du Code

5

Cliquer ensuite sur « Device Configuration Tool Code Generation »



6

Ajouter ces « fichiers d'en-tête privés": **Private includes**

```
22 /* Private includes ----------
23 /* USER CODE BEGIN Includes */
24 #include <stdio.h>
25 #include <string.h>
26 /* USER CODE END Includes */
27
```

```
/* Private includes ----------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
/* USER CODE END Includes */
```

Modification du Code

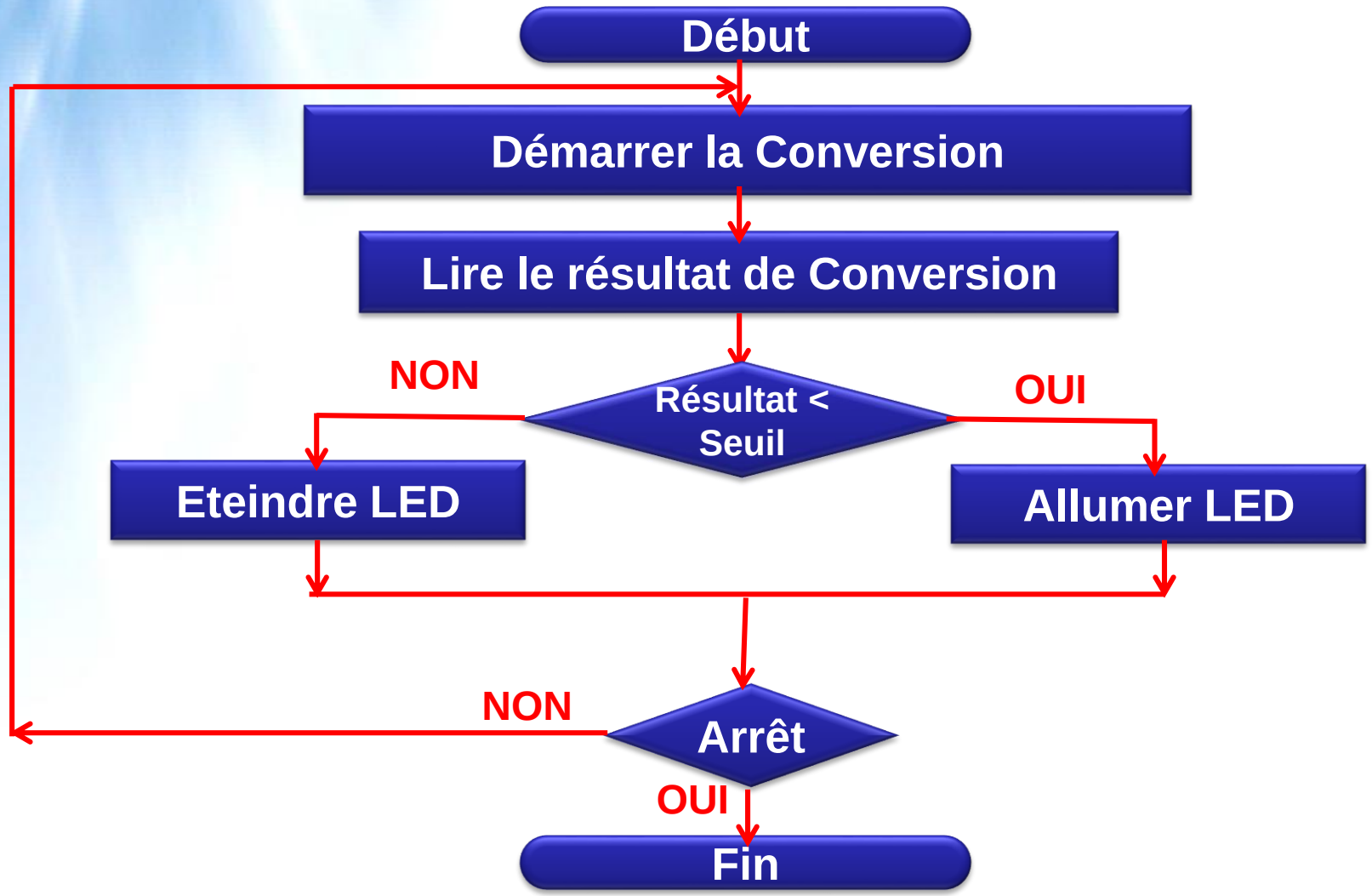
7

Ajouter deux variables privées: lux de type entier non signé et msg de type char

```
43 /* Private variables ----------------
44 ADC_HandleTypeDef hadc1;
45
46 UART_HandleTypeDef huart2;
47
48 /* USER CODE BEGIN PV */
49 uint16_t lux=0;
50 char msg[20];
51
52 /* USER CODE END PV */
```

```
/* Private variables ---------------------------------------*/
ADC_HandleTypeDef hadc1;
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
uint16_t lux=0;
char msg[20];
/* USER CODE END PV */
```

Organigramme du programme (While loop)



Le résultat de conversion sera affecté à la variable lux

Le « Seuil » caractérise de le degré d'obscurité qui déclenche l'allumage de la LED

Le Code de la boucle While

8

Ouvrir le fichier principal main.c

9

Insérer le code suivant dans la boucle while du main.c

```
/* USER CODE BEGIN 3 */
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1,20);
lux=HAL_ADC_GetValue(&hadc1);
if (lux<2000) {
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
}else {
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
}
HAL_Delay(200);
```

```
102  while (1)
103  {
104    /* USER CODE END WHILE */
105
106    /* USER CODE BEGIN 3 */
107      HAL_ADC_Start(&hadc1);
108      HAL_ADC_PollForConversion(&hadc1,20);
109      lux=HAL_ADC_GetValue(&hadc1);
110      if (lux<2000) {
111            HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
112      }else {
113            HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
114          }
115      HAL_Delay(200);
116  }
117  /* USER CODE END 3 */
```

Le Code de la boucle While

REMARQUES:

- **HAL_ADC_Start(&hadc1)** : Cette fonction est utilisée pour démarrer la conversion analogique-numérique (ADC) du canal de l'ADC 1.

- **HAL_ADC_PollForConversion (&hadc1,20):** Cette fonction est utilisée pour attendre que la conversion ADC soit terminée sur l'ADC 1 (**hadc1**). Elle bloque l'exécution du programme jusqu'à ce que la conversion soit terminée ou que le délai d'attente (**timeout=20ms**) expire.

- **HAL_GPIO_WritePin**: C'est une fonction de la bibliothèque HAL qui permet de contrôler l'état d'une broche GPIO (mise à un ou à zéro).

10

Lancer, enfin, la compilation

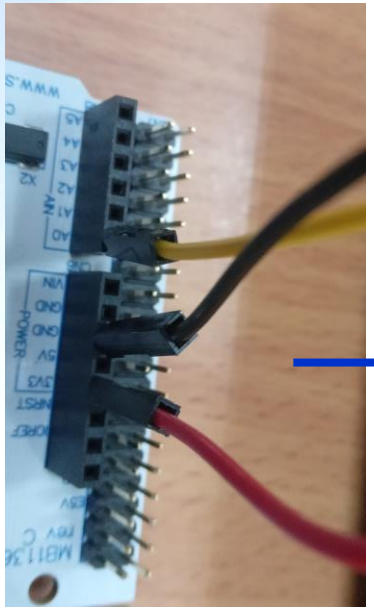
```
Finished building: ADC.list

13:03:38 Build Finished. 0 errors, 0 warnings. (took 2s.379ms)
```

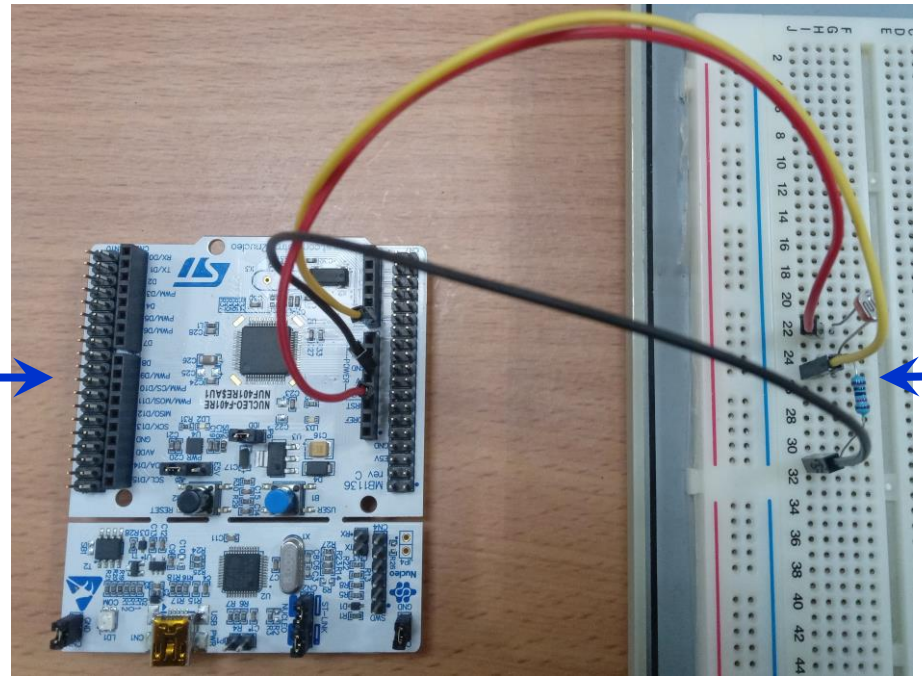
Le banc de Test:

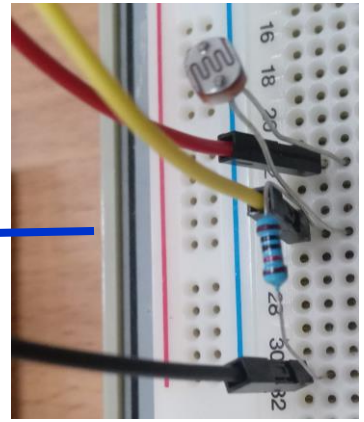
11

Avant de lancer le Débogage, il convient de faire le câblage nécessaire:

- Mettre en série la LDR et la résistance de 10kΩ
- Réaliser les connections ci-dessous
- Relier la carte NUCLEO F401RE au port USB





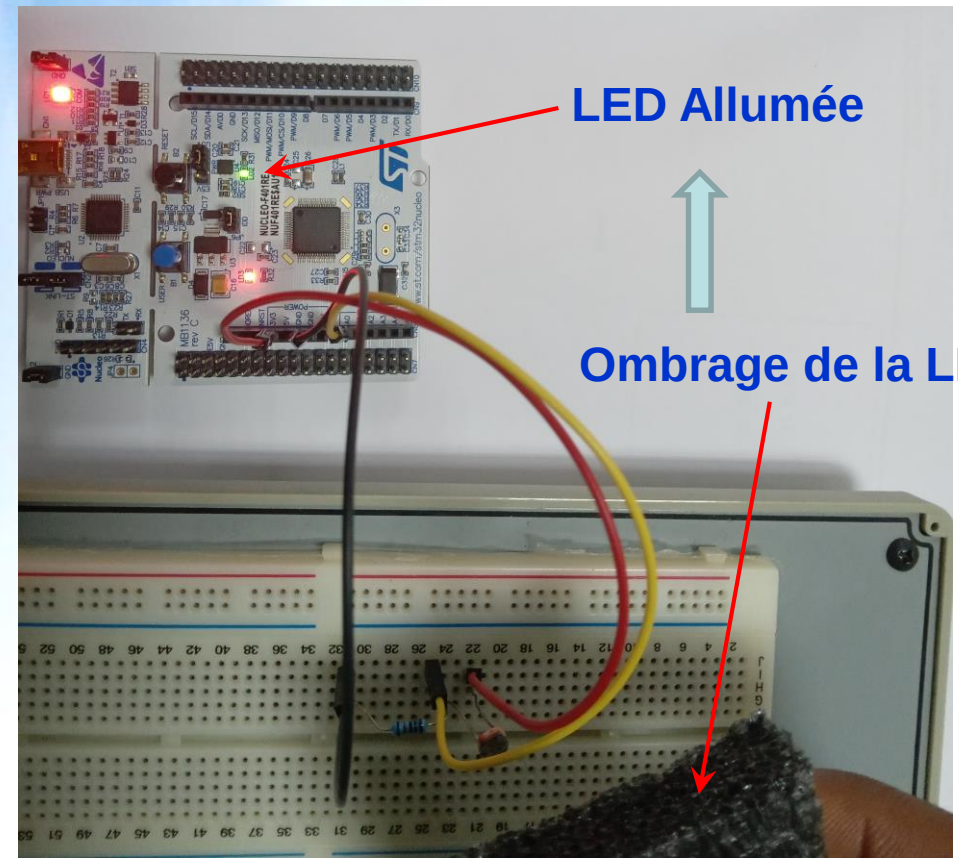


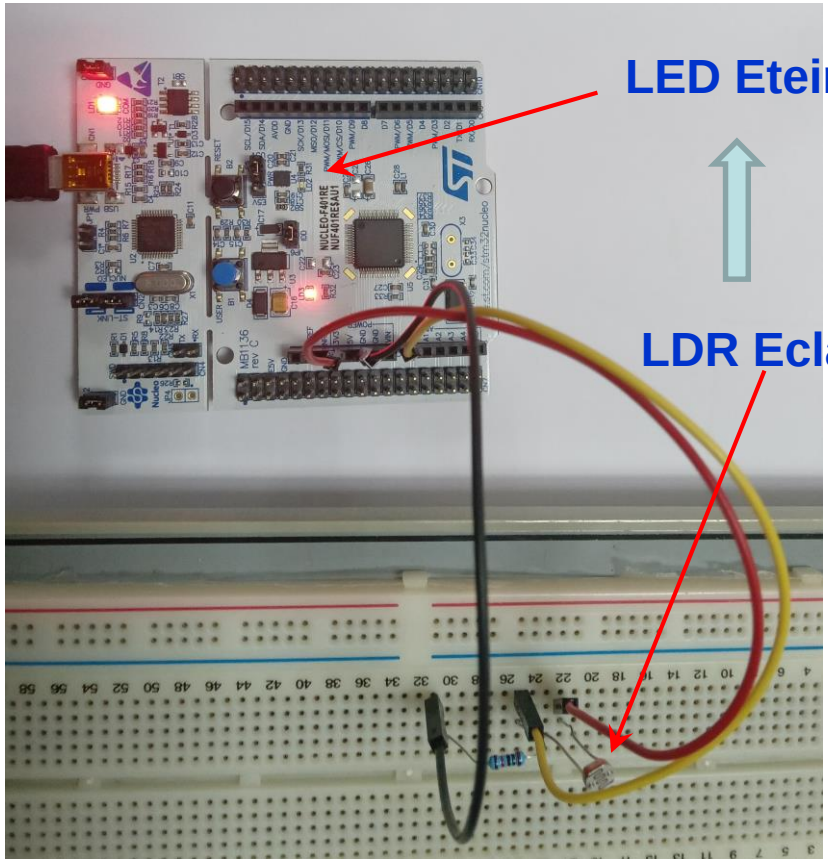
12

Finalement, Lancer l'exécution

VERIFIER !!!

Les Résultats de Test:

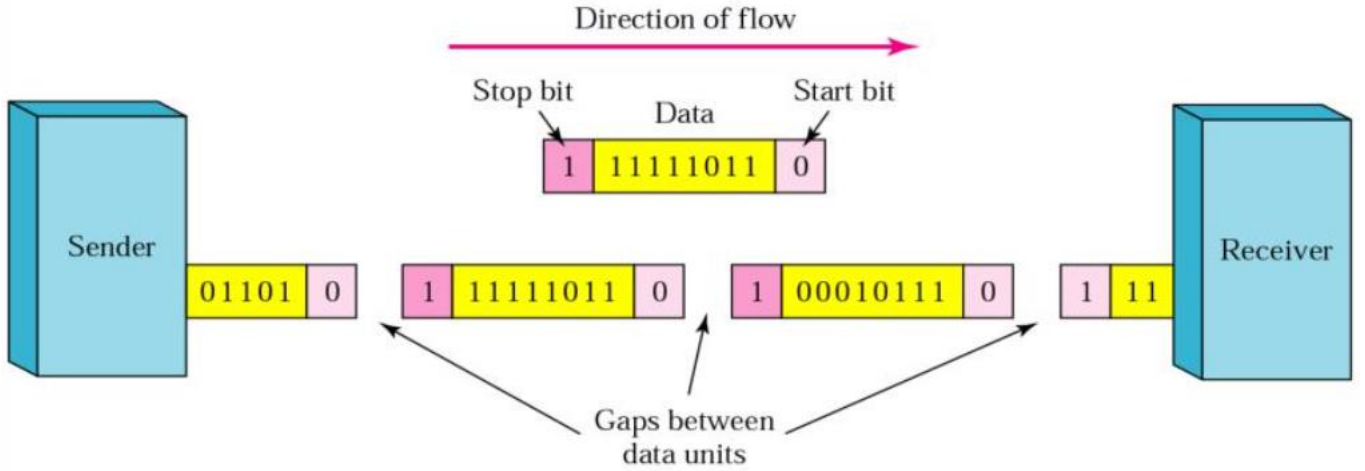




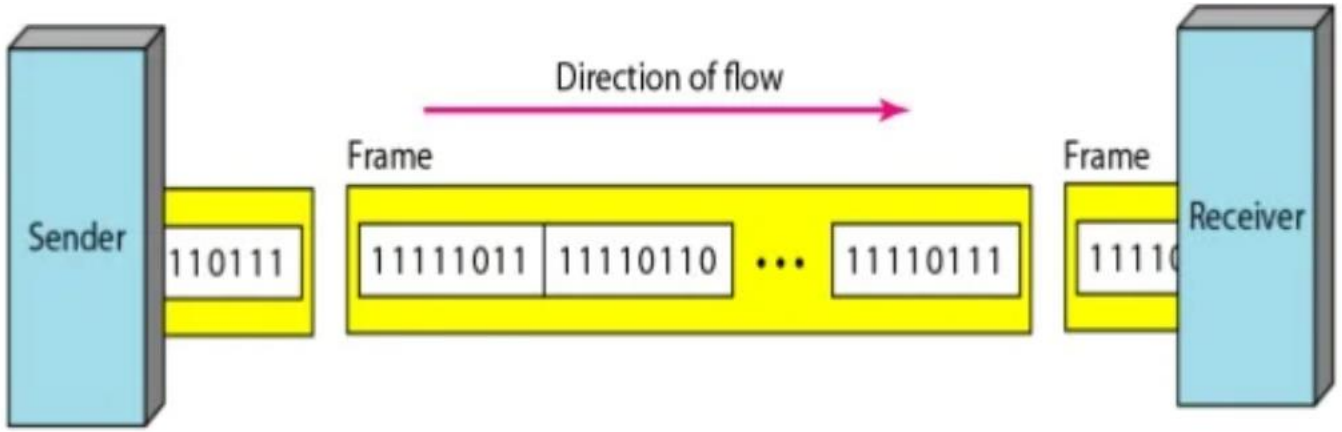
Amélioration de l'Application: Transmission de la donnée convertie par l'ADC en utilisant l'USART

- **UART**: Universal Asynchronous Receiver Transmitter, est un Emetteur-Récepteur Asynchrone Universel.
- **USART**: Universal Synchronous/Asynchronous Receiver Transmitter)

Transmission Asynchrone:



Transmission Synchrone:



USART vs. UART

Comparaison entre USART et UART:

- L'USART utilise des signaux d'horloge et de données pour son bon fonctionnement. Au contraire, UART utilise uniquement des signaux de données.
- USART transmet les données sous forme de blocs tandis que dans UART, un octet est transmis à la fois.
- Les données en mode synchrone USART sont transmises à un rythme défini. Par contre, en UART les données peuvent être transmises à vitesse variable.
- La vitesse de transfert des données dans UART peut varier: 4800 bps, 9600 bps,38400 bps, … A l'inverse, dans le cas de l'USART, le mode synchrone offre un débit de données élevé par rapport au mode asynchrone.
- USART utilise le mode full-duplex, tandis que UART utilise le mode Half-duplex.
- La vitesse globale de l'USART est supérieure à celle de l'UART.

Transmission de la donnée convertie par l'ADC en utilisant l'USART

Cahier des Charges:

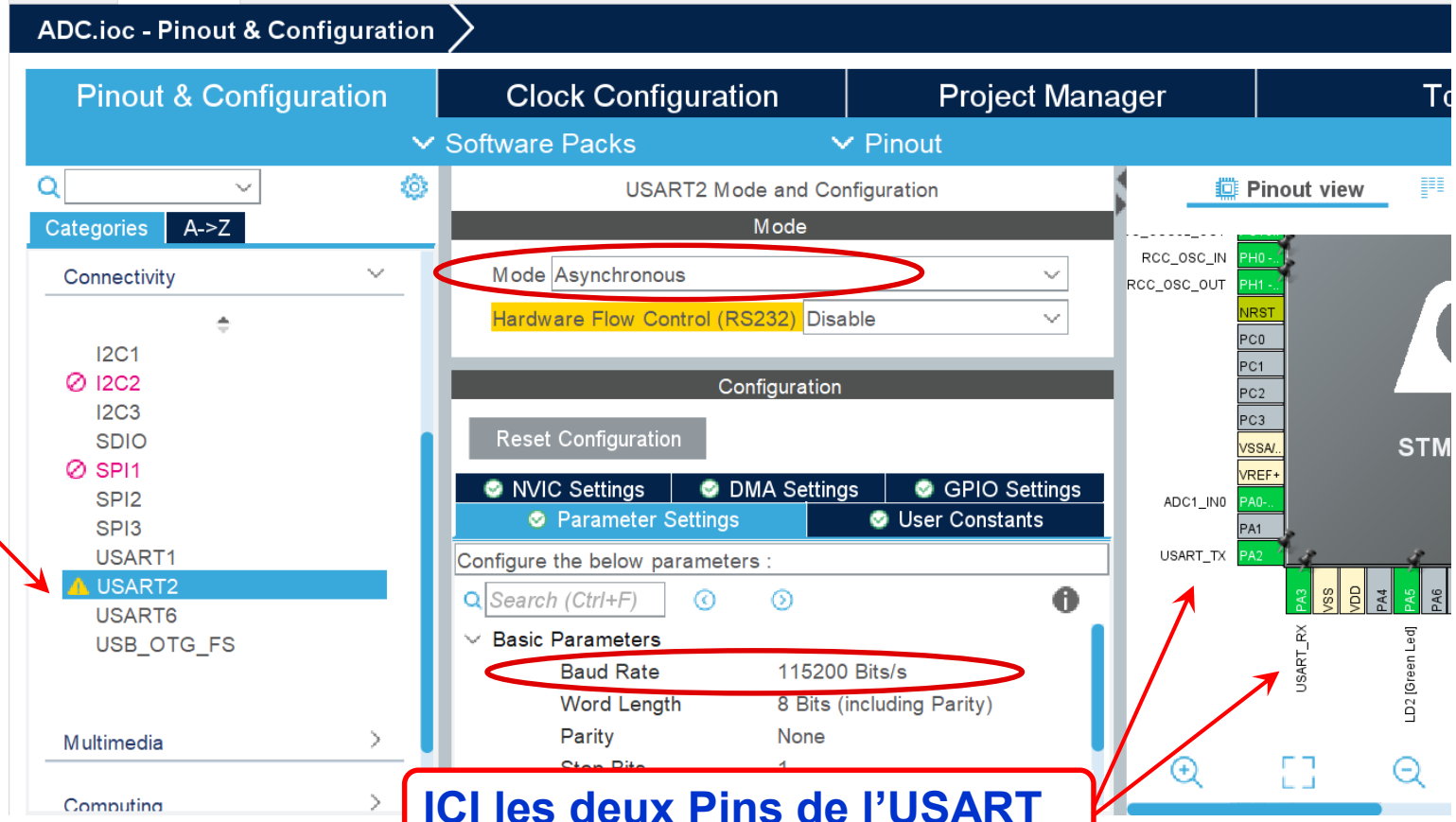
- Il s'agit d »'améliorer le programme de mesurer l'intensité lumineuse d'une LDR en y intégrant la transmission du résultat de conversion via le câble USB en utilisant une transmission par l'USART.

Configuration de l'USART:

1

Commencer par configurer l'USART dans l'interface graphique du STM32CubIDE

- **Choisir Connectivity, USART2**
- **Mode: « Asynchronous »**
- **Débit: 115200 bps**
- **Garder les autres paramètres par défaut**



2

Mettre à jour le code

ICI les deux Pins de l'USART

Transmission de la donnée convertie par l'ADC en utilisant l'USART

Modification du code dans main.c:

3

Modifier, ensuite le code du fichier principal main.c

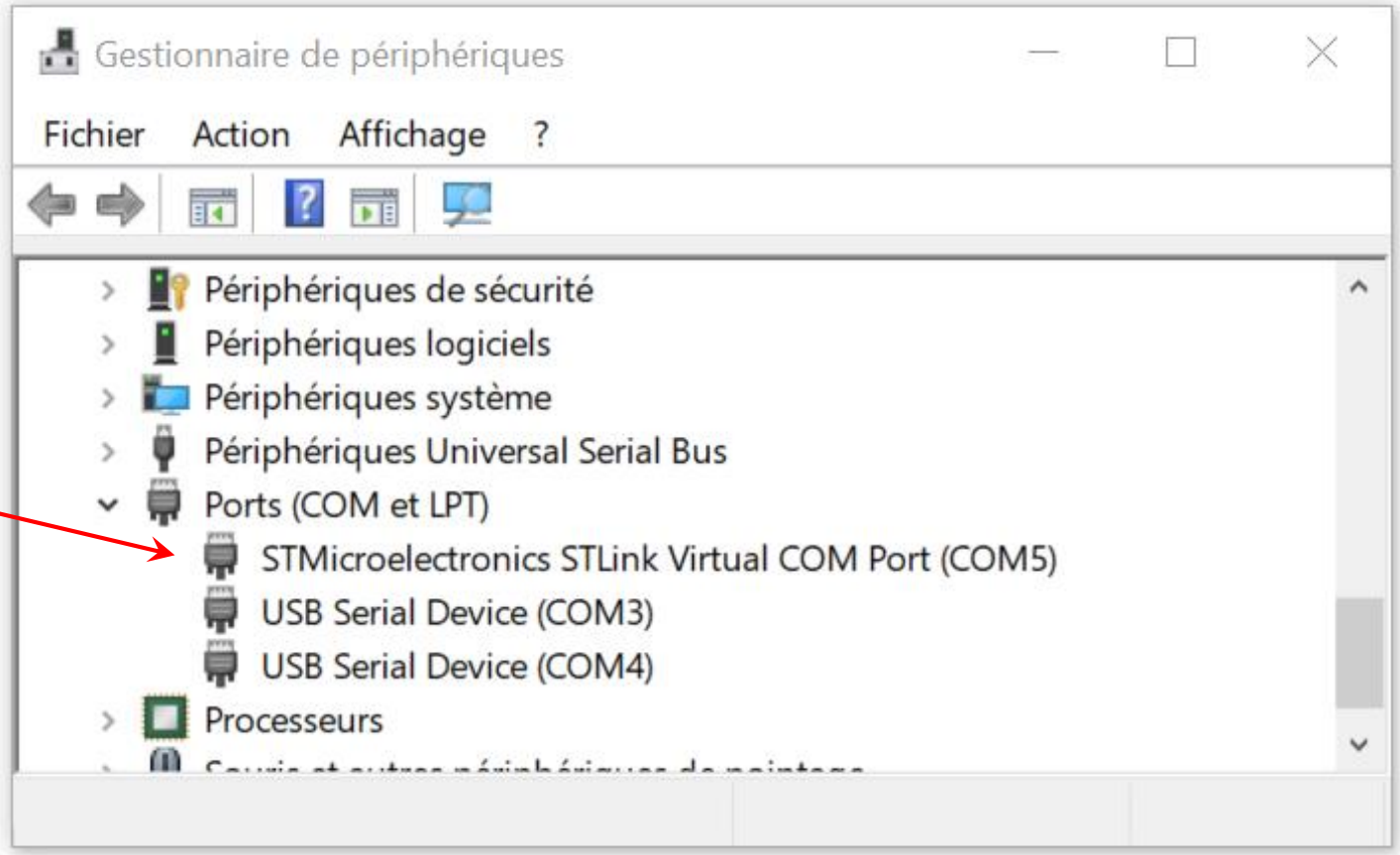
```
while (1)
{
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1,20);
lux=HAL_ADC_GetValue(&hadc1);
sprintf(msg,"Light : %hu \r\n",lux);
HAL_UART_Transmit(&huart2,(uint8_t *)msg, strlen(msg),HAL_MAX_DELAY);
if (lux<2000) {
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
}else {
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
}
HAL_Delay(200);
}
```

Transmission de la donnée convertie par l'ADC en utilisant l'USART

4

Vous devez connaître le port COM du bus USB sur lequel est connectée la carte NUCLEO
Pour cela, ouvrer le gestionnaire de périphériques et cliquer sur Ports (COM et LPT)

ICI la carte est connectée sur le COM5



Transmission de la donnée convertie par l'ADC en utilisant l'USART

5

Télécharger et Installer l'utilitaire PuTTy (https://putty.org/)

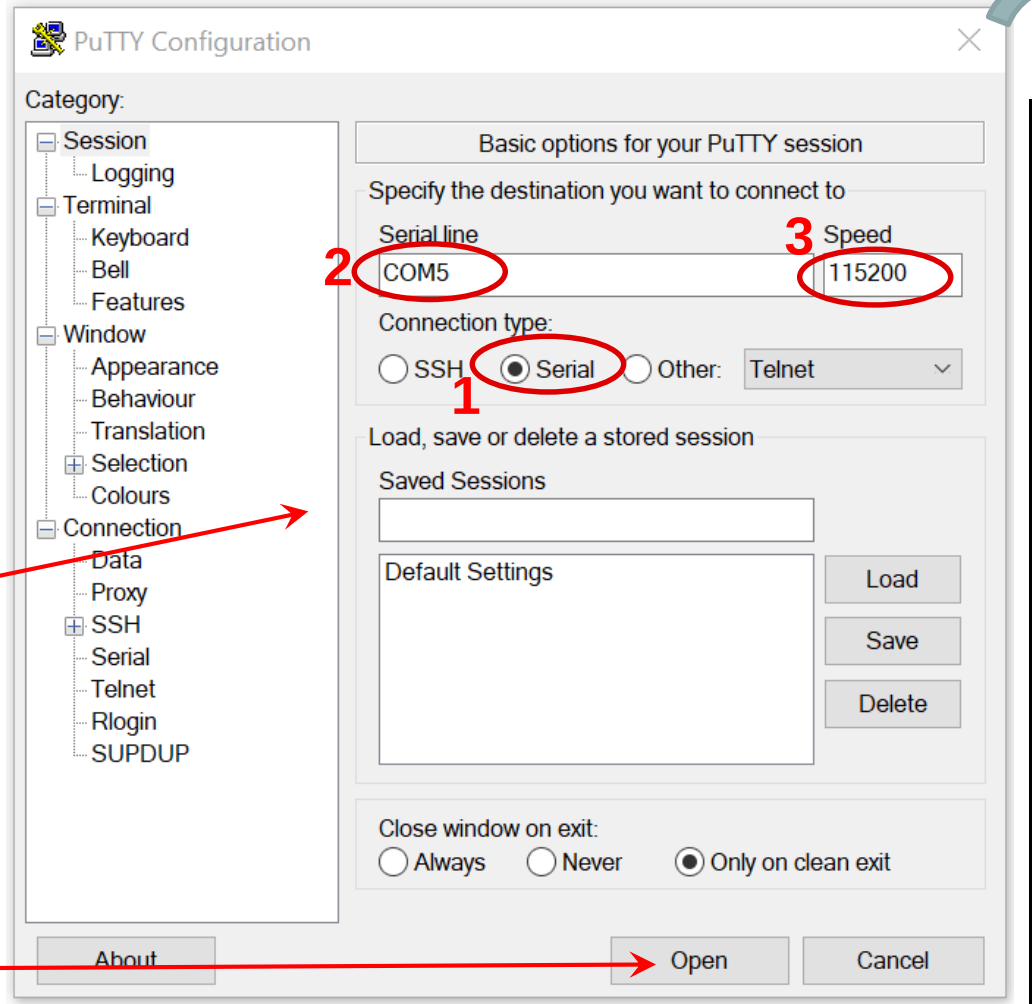
6

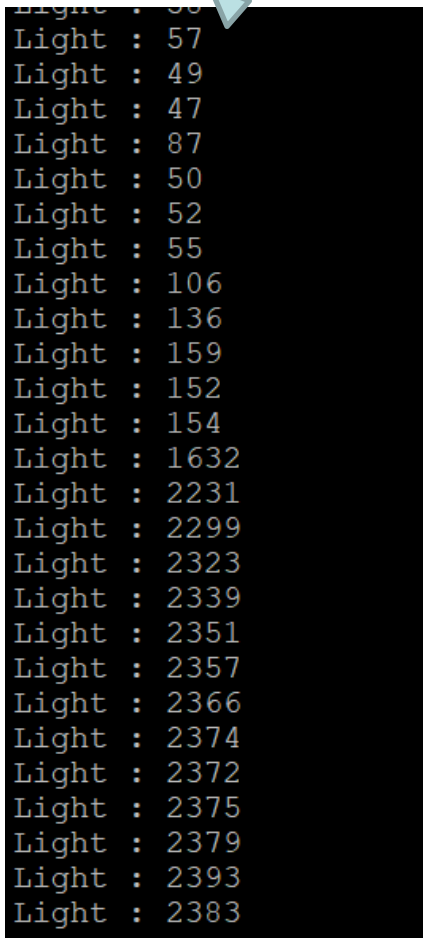
Lancer PuTTY et configurer:

1) **Connectivity type: Serial**
2) **Serial line: COM5 (celui détecté dans le gestionnaire de périphériques)**
3) **Speed: 115200 (celui programmé dans l'USART)**

7

Cliquer ensuite sur « Open »





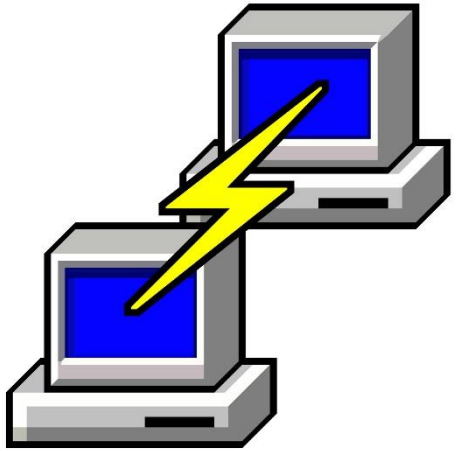
Un mot sur PuTTY

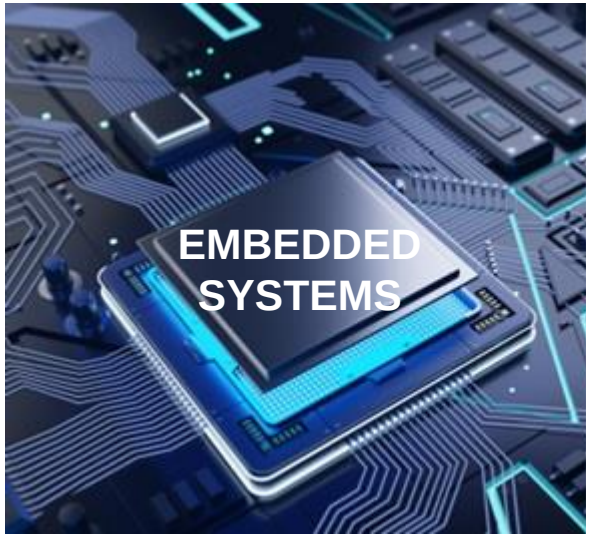
PuTTY est un programme client qui permet d'accéder à des systèmes distants, en particulier à des serveurs via le protocole SSH (Secure Shell) pour une connexion sécurisée.

Le nom PuTTY est souvent utilisé pour désigner à la fois le client SSH et les autres outils associés qui font partie du projet PuTTY.

Voici quelques-unes des principales caractéristiques de PuTTY :

1. **Connexion SSH**
2. **Connexion Telnet**
3. **Transfert de fichiers SCP et SFTP**
4. **Connexions série**
5. **Terminal émulateur**
6. **Interface graphique et ligne de commande**





FIN !
Questions ?