

Table des matières

1	Mémo	2
1.1	Atollic.....	2
1.2	HAL	2
1.3	CubeMx	3
1.4	Créer un projet from scratch pour une carte d'éval Nucleo-L152RE	3
2	Ajouter la fonction.....	4
2.1	External IT	4
2.2	TIM.....	5
2.3	SPI	5
2.4	I2C.....	6
2.5	UART	6
2.6	ADC.....	7
2.7	Printf_UART	7
2.8	Printf_ITM.....	8

1 Mémo

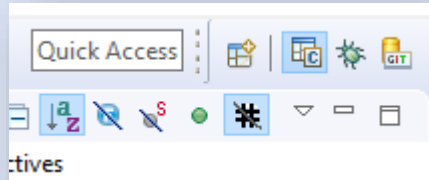
1.1 Atollic

Version 9.1, attention la 9.2 génère des erreurs avec les ST-Link (probe not connected...)

1.1.1 Shortcuts

Ctrl+Shift+A	Sélection carrée
Ctrl+Maj+ /	Comment / Uncomment
F3	Go to Definition
Ctrl+ LClic	Go to Definition
Alt + →	History backward
Ctrl+Alt+ ↓	Duplique la dernière ligne
Ctrl + F	Rechercher dans le document
Ctrl + H	(Onglet File Search) Rechercher dans le projet entier. Astuce 1, seul le file Search fonctionne correctement, aller dans 'Customize...' et désélectionner les autres onglets. Astuce 2, mettre *.c, *.h, *.s dans le file name pattern

1.1.2 Actions

Ouvrir un projet	File→open project from File system
Augmenter le niveau de debug	Project→Build Settings→Tool Settings puis 1) C Compiler→Opimization→ Optimization level→Optimize for debugging 2) C Compiler→Debugging→Debug Level →Maximum (-g3)
Changer de perspective	Icônes en haut à droite. Le scarabée pass en perspective debug, la fenêtre avec un C revient en code C. 
Réinitialiser les perspectives	En cas de fausse manipulation sur les vues, il est possible de réinitialiser la perspective en cours : Windows→Perspective→reset perspective...

1.2 HAL

1.2.1 Sémantique

LLL = Library (HAL, TIM, RCC, GPIO ; UART, I2C, ...)

FUNC = Fonction

Fonction	HAL_LLL_FUNC() - LLL = library (TIM, UART, SPI, ...) - FUNC = function (Init, Start_IT, ...)
----------	--

Handle	Utilisé par SPI, UART, TIM, ... pour manipuler les objets initialisés.
	LLL_HandleTypeDef

1.2.2 Fonctions clés

Nombre de ms depuis le démarrage	HAL_GetTick()
Fréquence du core	HAL_RCC_GetSysClockFreq()
Pause (fonction bloquante)	Hal_Delay(nb_ms)
Générer une erreur bloquante	Error_Handler

1.3 CubeMx

1.3.1 Actions

Mises à jour des librairies	Help → Manage Embedded Software packages
Pinout, sélection d'un mode	Left click sur la pin
Pinout, changement du label	Right click sur la pin, puis enter user label
Pin, Changement de pin sur un alternate	Ctrl + Left click on pin

1.4 Créer un projet from scratch pour une carte d'éval Nucleo-L152RE

1.4.1 Sélection de la board

- 1) Home → New Project → Acces to board selector
- 2) Dans la loupe '152RE' puis select
- 3) Click dans une cellule du tableau puis
- 4) Start project
- 5) Initialize all peripherals with... → Yes

1.4.2 Configuration de l'horloge 8Mhz venant du ST Link

- 1) Onglet 'Clock configuration'
- 2) HSE → PLL source mux
- 3) PLL MUL = x12
- 4) Check HCLK (MHz) = 32

1.4.3 Configuration de la génération du projet




- 1) Onglet 'Project Manager' → Sous-Onglet Project
- 2) Remplir Project Name et Project Location
- 3) ToolChain = TrueStudio
- 4) Onglet 'Project Manager' → Sous-Onglet 'Code Generator'
- 5) Check 'Copy only the necessary files'

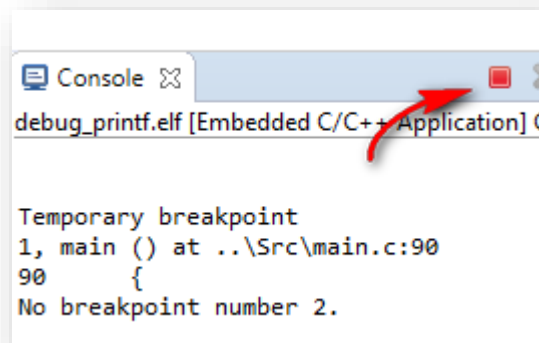
1.4.4 Pinout & Configuration

- Mettre à jour le debug pour la nucleo System → SYS → Debug → Trace Asynchronous Sw
- Basculer RCC_OSC_OUT en Reset_State
- Remplir toutes les fonctions désirées.

- Cocher NVIC pour toutes les IT désirée. Attention, pour chaque type de NVIC coché (GPIO, TIM, ADC, ...), il faudra aller chercher le callback correspondant dans le HAL
- Cliquer sur 'Generate Code'
- 'Open Project...'

1.4.5 Atollic

- Aller chercher les Callbacks des NVIC dans Drivers/STM32L1xx_HAL_Driver/Src
- Build  & Debug 
 - Si le debug ne marche pas, Debug configuration () puis Debugger, passer 'Debug Probe' sur ST-Link
 - Si au lancement du debug un message apparait disant que le debug est déjà en route, aller manuellement dans la perspective debug (scarabé en haut à droite) et appuyer sur stop **dans la console** :



2 Ajouter la fonction...

Cette section décrit les opérations spécifiques à mener pour implémenter une nouvelle fonction. Ces opérations sont à effectuer en plus de la création d'un projet *'from scratch'*.

2.1 External IT

But : déclencher une interruption sur changement d'état d'une pin.

Le changement peut être :

- Une transition de l'état haut à l'état bas (falling)
- Une transition de l'état bas à l'état haut (rising)

L'interruption est associée au pin number

2.1.1.1 Cube

- 1) Configurer les entrées concernées en GPIO_EXTIxx (xx = pin number).
- 2) Dans System Core -> GPIO -> NVIC, activer les EXTI line correspondante

Ces opérations configurent le projet jusqu'à l'appel du HAL_GPIO_EXTI_IRQHandler dans le fichier stm32xxx_it.c, il restera à câbler l'appel de Callback dans le main.

2.1.1.2 *Projet C*

Ajouter la fonction void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) dans le main (aller la chercher dans stm32l1xx_hal_gpio.c en enlevant le __weak).

2.2 TIM

2.2.1 Timer interne

But : déclencher une IT à interval régulier. Pas de pin de sortie.

2.2.1.1 *Cube*

- 1) Configurer un timer Pinout & Configuration ->Timers-> TIMx
- 2) L'activer en sélectionnant 'Internal Clock' dans 'Clock Source'
- 3) Dans NVIC Settings, activer l'interruption correspondante
- 4) Configurer le timer dans parameters settings ($F_{out} = F_{sys}/(Prescaler * CounterPeriod)$)

Ces opérations configurent le projet jusqu'à l'appel du HAL_TIM_IRQHandler dans le fichier stm32xxx_it.c, il restera à câbler l'appel du Callback dans le main.

2.2.1.2 *Projet C*

- 1) Ajouter la fonction void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) dans le main (aller la chercher dans stm32l1xx_tim.c en enlevant le __weak).
- 2) Activer le timer après sa configuration en ajoutant la fonction HAL_TIM_Base_Start_IT(&htimx) où x est le numéro de timer

2.2.2 PWM

But : sortir un signal périodique de rapport cyclique variable.

Il n'y a pas d'interruption dans ce mode, qui ne fait pas appel au CPU pour générer le signal.

2.2.2.1 *Cube*

- 1) Sélectionner la sortie désirée en la configurant en TIMx_CHy. La pin passe en orange
- 2) Activer le timer correspondant dans Timers->TIMx en selectionnat 'Internal Clock' dans 'Clock source'
- 3) Au même endroit, activer le channel y en selectionnant 'Output Compare CHx' dans 'Channely', la pin passe en vert
- 4) Configurer le timer dans parameters settings ($F_{out} = F_{sys}/(Prescaler * CounterPeriod)$)
- 5) Configurer le ratio dans Output compare Channel 2->Pulse

2.2.2.2 *Projet C*

Lancer le timer en ajoutant la fonction HAL_TIM_PWM_Init(&htimx) après sa configuration (x = numéro de timer).

2.3 SPI

2.3.1 *Cube*

Il y a 3 ou 4 pin à configurer :

- MISO – Master Input Slave Output
- MOSI – Master Output Slave Output
- SCK – Clock
- CS_N – Chip Select

Attention ! En master only, la min MOSI n'existe pas.

Attention ! Si le CS_N est logiciel (pin incompatible avec le layout de l'alternate), c'est une GPIO_output qui sera basculée avant et après accès SPI.

- 1) Configurer les pins dans le Pinout View
- 2) Dans connectivity SPIx, configurer
 - Le Mode
 - i. Transmit Only Master pour un envoi simple
 - ii. Full Duplex Master pour un e envoi avec lecture de l'esclave
 - Le NSS Signal. C'est le CS_N, si CS_N software, le basculer sur Disable
 - La Data Size (8 ou 16 bits)
 - Le Prescaler (division sur la fréquence du bus APB)
 - Pas de NVIC
- 3) Si CS_N software
 - Dans System Core->GPIO, configurer le GPIOOutput level du CS_N à High

2.3.2 Projet C

Utiliser HAL_SPI_Transmit / HAL_SPI_Receive. *Attention ! La taille des données dépend de la DataSize configurée dans le SPI.*

Si CS_N software, il faut basculer la pin CS_N à 0 avec l'instruction HAL_SPI_xxx, et a rebasculer à 1 après avec HAL_GPIO_WritePin.

2.4 I2C

2.4.1 Cube

Il y a deux pin à configurer : SCL et SDA.

- 1) Configurer l'I2C connecté au périphérique (s)
 - Basculer la pin SCL (clock) en I2Cx_SCL
 - Basculer la pin SDA (data) en I2Cx_SDA
- 2) Dans la section *connectivity*, configurer la vitesse en standard (100kHz)
- 3) Pas de NVIC par défaut

2.4.2 Projet C

- 1) Utiliser HAL_I2C_Mem_Write/ HAL_I2C_Mem_Read pour un accès avec registre I²C interne. *Attention, il y a deux tailles dans ces fonctions, celle de l'adresse interne du device et celle des données !*

2.5 UART

Configurer une communication asynchrone sur 2 pin.

USART = UART si le mode est configuré sur asynchrone.

2.5.1.1 Cube

- 1) Dans le pinout view, configurer sur les pins de TX et de RX sur le même UART. Par exemple PA2 sur USART2_TX et PA3 sur USART2_RX, c'est le bus mappé en COM virtuel dans le STLINK.
- 2) Dans Connectivity->USARTx, basculer le mode de Disabled à Asynchronous
- 3) Dans les parameters settings, configurer Baud Rate, Parity et Stop Bits



2.5.1.2 *Projet C*

Utiliser les fonctions HAL_UART_Transmit et HAL_UART_Receive.

2.6 ADC

But : convertir à interval régulier de 1 à n entrées analogiques.

2.6.1.1 *Cube*

2.6.1.2 *Projet C*

2.7 Printf_UART

2.7.1 *Cube*

- 1) Configurer l'UART connecté au ST_LINK (USART2)

- 115200 bps,
- parity none,
- 1 stop bit,
- Disable hardware flow control
- Pas de NVIC (envoi seulement)

2.7.2 *Projet C*

- 1) Dans main.c, inclure stdio.h pour activer le printf
- 2) Dans main.c, redéfinir le __write() pour envoyer les messages de printf vers l'uart :

```
/* USER CODE BEGIN 4 */
int __write(intfile, char*ptr, intlen)
{
    HAL_UART_Transmit(&uart2, (uint8_t *)ptr, len, 10);
    return len;
}
/* USER CODE END 4 */
```

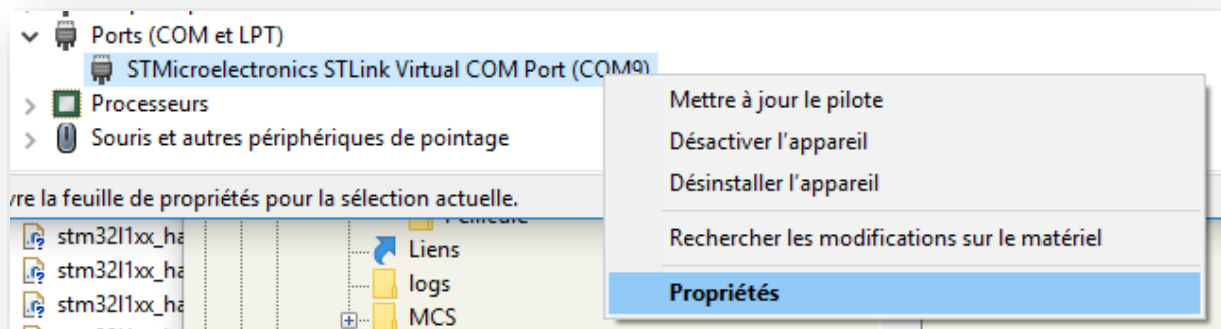
- 3) Utiliser ensuite le printf :

```
/* USER CODE BEGIN 2 */
printf ("test\r\n");
/* USER CODE END 2 */
```

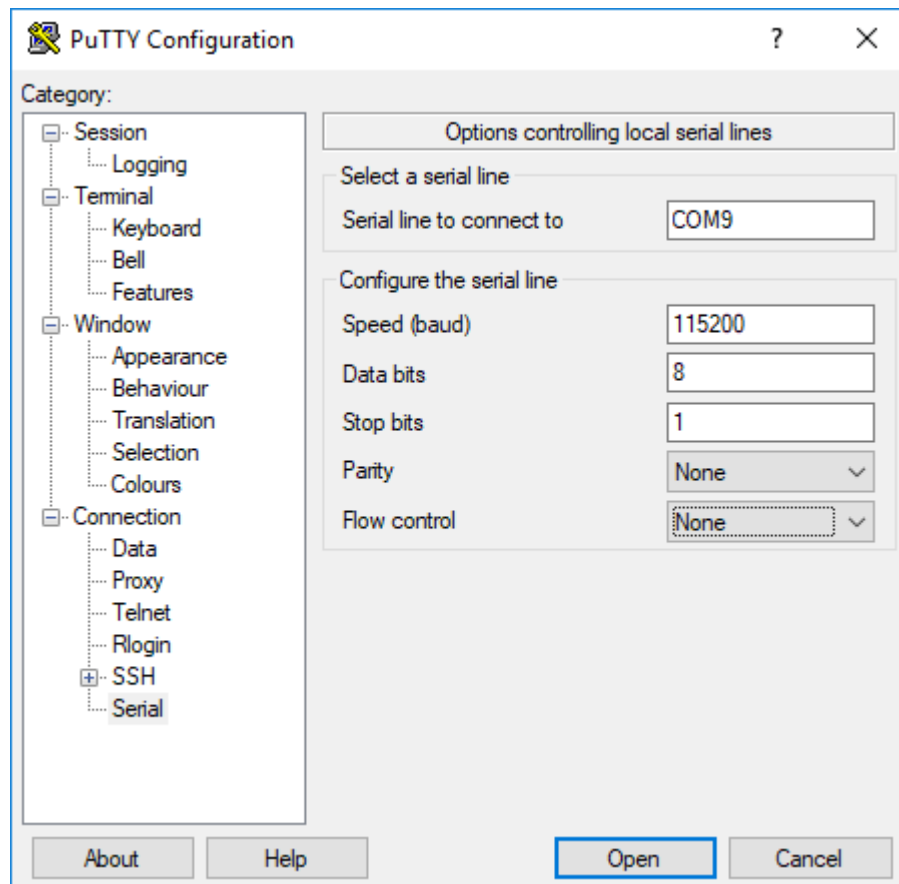
- 4) Pour visualiser les traces :

- Configurer dans Windows le port série du STM avec les mêmes paramètres (propriétés -> paramètres du port).





- Ouvrir un terminal putty sur ce port :



- 5) Lancer le code.

2.8 Printf_ITM

L'ITM est une interface de debug hardware intégrée au Cortex. Elle est plus rapide que l'UART et utilise la pin SWO du µC. Pour utiliser cette sortie, :

- 1) Configurer la sortir SWO dans le cube (SYS->Debug->Trace Asynchronous Sx)
- 2) Dans main.c, inclure stdio.h pour activer le printf
- 3) Dans main.c, redéfinir le __write() pour envoyer les messages de printf vers l'ITM :

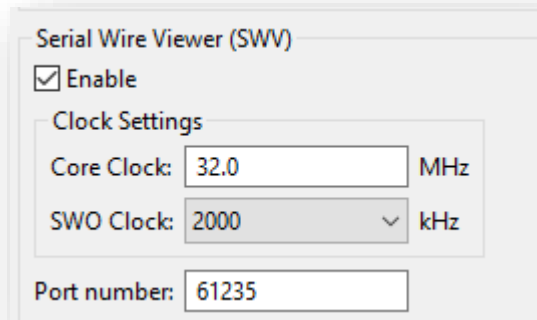

```
/* USER CODE BEGIN 4 */  
int _write(intfile, char*ptr, intlen)  
{  
    int DataIdx;  
    for(DataIdx=0; DataIdx<len; DataIdx++)  
        ITM_SendChar(*ptr++);  
}  
/* USER CODE END 4 */
```

4) Utiliser ensuite le printf :

```
/* USER CODE BEGIN 2 */  
printf ("test\r\n");  
/* USER CODE END 2 */
```

5) Pour visualiser les traces :

- Activer le SWV dans les parametres du ST-LINK
 - Run -> Debug configuration->Debugger(Tab)



- Ouvrir dans Atollic la fenetre SWV (Windows -> Show View -> Other -> SWV Console)
- **Lancer le debug**
 - Dans la fenêtre SWV Console, ouvrir les paramètres (🔧) et cocher la case 0



- Cliquer sur Start trace (🔴)