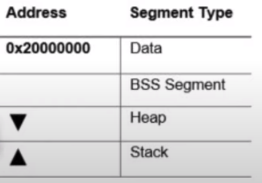
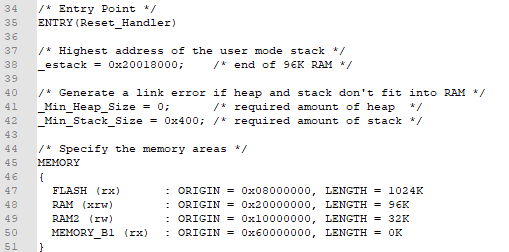
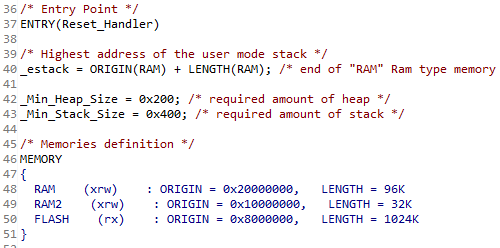
**BOOTLOADER**

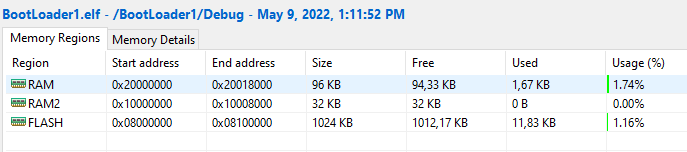
Exemple : avec CubeMXide

Créer un projet avec la bonne cible stm32.

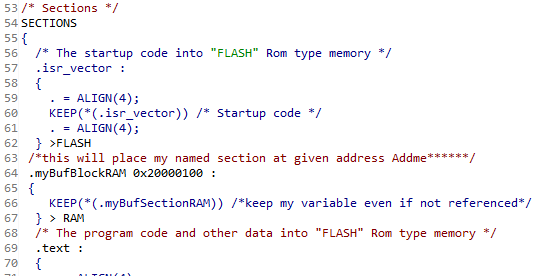
Changer la plage d’adresse projet avec Atollic application

STM32L4xx\_Flash.Id



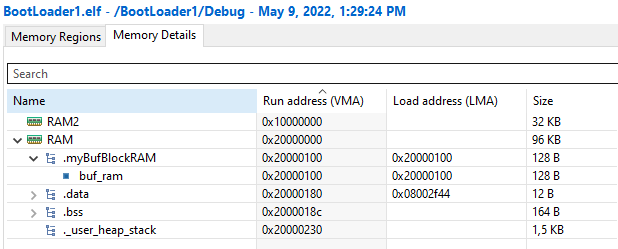


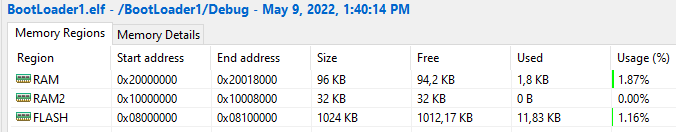
Ajouter dans la rubrique Sections Addme



Ajouter dans le main



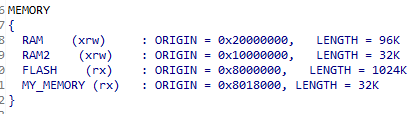




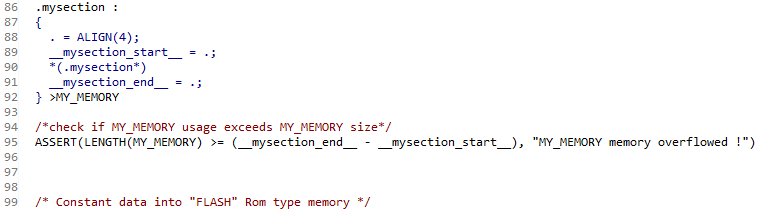
Binaries>BootLoader1.elf (click droit>show In>System Explorer) !! je n’arrive pas à faire le .bin

Blink LED dans un bootLoader

STM32L4xx\_Flash.id



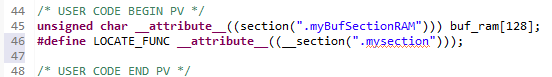
+

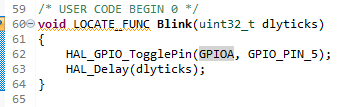


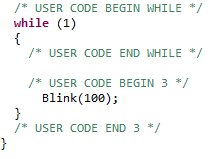
A LACOMPILATION NOUS DEVONS AVOIR UNE REGION MEMOIRE : MY\_MEMORY

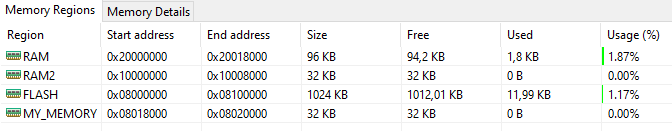
Main.c

Ajout du #define

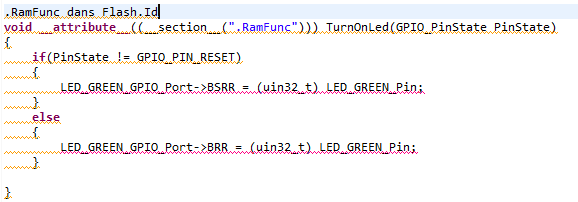








Si on met



Le programme ajoute TurnOnLed dans la RAM et pas dans MEMORY

0x8000000 96k 0110 0000 0x60 avant 128k 1000 0000 0x80

0x8018000 32k 0010 0000 0x32

0110 0000

0010 0000 +

1000 0000 =

0x081000000

0x08000000

1000 0000 0000 0000 0000 0000 0000

STM32L476RGTX\_FLASH.Id

Créer le projet

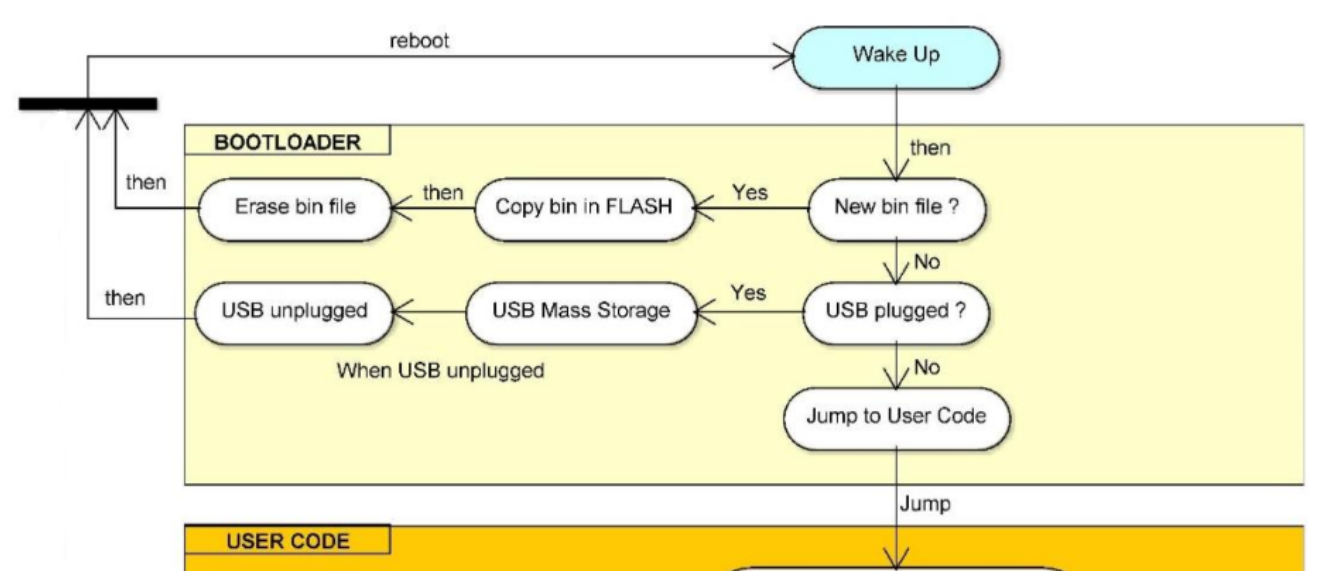
|  |  |
| --- | --- |
| Bootloader | Application |
| STM32L4xx\_FLASH.Id    Drivers/  {BSP (Components[accelero, audio, camera, epd, gyro,  hsensor, idd, io, lcd, magneto, psensor, ts,  tsensor{h}]},  {stmpe1600, stmpe811}  {STM32L476G\_EVAL[stm32l476g\_eval…{c,h}}  {CMSIS, STM32L4xx\_HAL\_DRIVER}  Inc/  [flash\_if, IAP, main, stm32l4xx\_hal\_conf.h, stm32l4xx\_it, usbd\_conf, usbd\_desc, usbd\_storage {h}]}  Middlewares/  {FatFs, STM32\_USB\_Device\_Library}  Scritps/  {GetRevision.class , getrevision.java}  Src/  [flash\_if, IAP, proc.c, proc.h, main, stm32l4xx\_it, usbd\_conf, usbd\_desc, usbd\_storage, version.h {c}]}  Main.c | STM32L4xx\_FLASH.Id |

BootLoader Chargeur de démarrage s’occupe de

Mettre à jour le micro logiciel applicatif.

Présenter le nœud comme un disque USB si le câble USB est connecté.

Lancer l’application (firmware.bin)



/Env/private

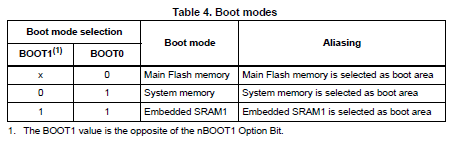
Contient des fichiers de référence pour détecter les modifications du fichier de configuration.

+ état des capteurs pour sauvegarder des valeurs entre les remises à zéro (endormissement)

/proc (info sur le système)

Bootloader

Dans le stm32L47x 3 modes de démarrage différentes peuvent être sélectionnés via la Broche BOOT0 et bit nBOOT1 dans l’octet d’option utilisateur comme indiqué dans le tableau :



La valeur BOOT1 est l’opposé du bit d’option nBOOT1

Pour définir nBOOT1 et BOOT0 pour sélectionner le mode de démarrage requis.

La broche BOOT0 et le bit nBOOT1 sont également rééchantillonnés lors de la sortie du mode veille.

Par conséquent, ils doivent être conservés dans la configuration requise du mode Boot en mode Standby.

Une fois ce délai de démarrage écoulé, la CPU récupère la valeur de haut de pile à partir de l'adresse

0x0000 0000, puis démarre l'exécution du code à partir de la mémoire de démarrage à 0x0000 0004.

Selon le mode de démarrage sélectionné, la mémoire flash principale, la mémoire système ou SRAM1 est accessibles comme suit :

• **Boot from main Flash memory** : la mémoire Flash principale est aliasée dans la mémoire de démarrage

(0x0000 0000), mais toujours accessible depuis son espace mémoire d'origine

(0x0800 0000). En d'autres termes, le contenu de la mémoire Flash est accessible à partir de

de l'adresse 0x0000 0000 ou 0x0800 0000.

• **Boot from system memory** : la mémoire système est aliasée dans l'espace mémoire de démarrage

(0x0000 0000), mais toujours accessible depuis son espace mémoire d'origine (0x1FFF 0000).

• **Démarrage à partir de la SRAM1 intégrée** : la SRAM1 est aliasée dans l'espace mémoire de démarrage

(0x0000 0000), mais il reste accessible depuis son espace mémoire d'origine (0x2000 0000).

Remarque : Lorsque l'appareil démarre à partir de la SRAM, dans le code d'initialisation de l'application, vous devez déplacer la table vectorielle dans la SRAM à l'aide de la table d'exception NVIC et du registre de décalage.

Lors du démarrage à partir de la mémoire flash principale, le logiciel d'application peut soit démarrer à partir de banque 1 ou à partir de la banque 2. Par défaut, le démarrage à partir de la banque 1 est sélectionné.

Pour sélectionner le démarrage à partir de la banque de mémoire flash 2, définissez le bit BFB2 dans les octets d'option utilisateur. Lorsque ce bit est défini et les broches de démarrage sont dans la configuration "démarrage à partir de la mémoire Flash principale", l'appareil démarre à partir de la mémoire système et le chargeur de démarrage saute pour exécuter l'utilisateur application programmée dans la banque de mémoire Flash 2. La mémoire système reste aliasée l'espace mémoire de démarrage (0x0000 0000). Pour plus de détails, veuillez-vous référer à AN2606.

Remarque : lors du démarrage à partir de la banque 2, le chargeur de démarrage permutera les banques de mémoire Flash. Par conséquent, dans le code d'initialisation de l'application, vous devez déplacer la table des vecteurs vers adresse de base permutée de la banque 2 (0x0800 0000) à l'aide de la table d'exception NVIC et de l'offset S'inscrire.

Remap physique

Une fois les broches de démarrage sélectionnées, le logiciel d'application peut modifier la mémoire accessible dans la zone de code (de cette façon, le code peut être exécuté via le bus ICode dans place du bus système). Cette modification s'effectue en programmant le SYSCFG registre de remappage de la mémoire (SYSCFG\_MEMRMP) dans le contrôleur SYSCFG.

Les mémoires suivantes peuvent ainsi être remappées :

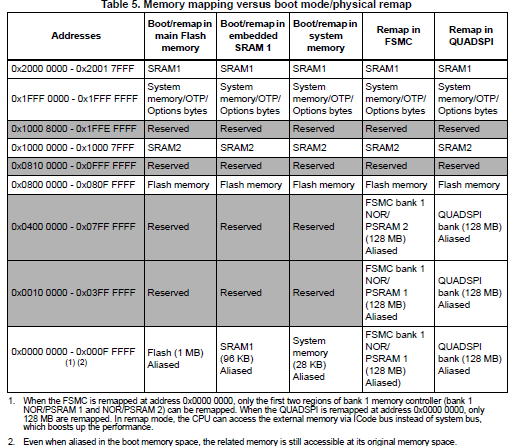
• Mémoire flash principale

• Mémoire système

• SRAM1 intégrée (96 Ko)

• Banque FSMC 1 (NOR/PSRAM 1 et 2)

• Mémoire Quad-SPI



Les appareils STM32L476xx comportent jusqu'à 1 Mo de mémoire Flash intégrée disponible pour stocker les programmes et les données. La mémoire Flash est divisée en deux banques permettant des opérations de lecture pendant l'écriture. Cette fonction permet d'effectuer une opération de lecture à partir d'une banque pendant qu'une opération d'effacement ou de programmation est effectuée sur l'autre banque. Le démarrage à double banque est également pris en charge. Chaque banque contient 256 pages de 2 Ko.

Des protections flexibles peuvent être configurées grâce aux octets d'option :

• Protection contre la lecture (RDP) pour protéger toute la mémoire. Trois niveaux sont disponibles :

– Niveau 0 : pas de protection contre la lecture

– Niveau 1 : protection contre la lecture de la mémoire : la mémoire Flash ne peut pas être lue ou écrit si l'une ou l'autre des fonctionnalités de débogage est connectée, le démarrage dans la RAM ou le chargeur de démarrage est choisi

– Niveau 2 : protection contre la lecture des puces : fonctionnalités de débogage (Cortex-M4 JTAG et série fil), le démarrage en RAM et la sélection du bootloader sont désactivés (fusible JTAG). Cette la sélection est irréversible.

Modes de démarrage

Au démarrage, la broche BOOT0 et le bit d'option BOOT1 sont utilisés pour sélectionner l'une des trois options de démarrage :

• Démarrage à partir de l'utilisateur Flash

• Démarrage à partir de la mémoire système

• Démarrage à partir de la SRAM intégrée

Le chargeur de démarrage est situé dans la mémoire système. Il est utilisé pour reprogrammer la mémoire Flash en utilisant USART, I2C, SPI, CAN ou USB OTG FS en mode Device via DFU (mise à jour du firmware de l'appareil).

SD-CARTE

Entrée/sortie numérique sécurisée et interface MultiMediaCards

(SDMMC)

L'interface hôte de la carte (SDMMC) fournit une interface entre le bus périphérique APB et les cartes multimédias (MMC), les cartes mémoire SD et les cartes SDIO.

Les fonctionnalités SDMMC incluent les éléments suivants :

• Conformité totale avec la version 4.2 des spécifications du système MultiMediaCard. Prise en charge des cartes pour trois modes de bus de données différents : 1 bit (par défaut), 4 bits et 8 bits

• Compatibilité totale avec les versions précédentes de MultiMediaCards (compatibilité ascendante)

• Conformité totale avec les spécifications de la carte mémoire SD Version 2.0

• Conformité totale avec la version 2.0 de la spécification de carte d'E/S SD : prise en charge de la carte pour deux différents modes de bus de données : 1 bit (par défaut) et 4 bits

• Transfert de données jusqu'à 48 MHz pour le mode 8 bits

• Écriture et lecture de données avec capacité DMA

USB

Universal serial bus on-the-go-speed (OTG\_FS)

Les appareils intègrent un appareil/hôte/périphérique OTG USB OTG pleine vitesse avec des émetteurs-récepteurs intégrés. Le périphérique USB OTG FS est conforme à la spécification USB 2.0 et à la spécification OTG 2.0. Il dispose d'un paramètre de point de terminaison configurable par logiciel et prend en charge la suspension/reprise. Le contrôleur USB OTG nécessite une horloge dédiée de 48 MHz qui peut être fournie par l'oscillateur multivitesse interne (MSI) ajusté automatiquement par l'oscillateur externe (LSE) de 32,768 kHz. Cela permet d'utiliser le périphérique USB sans cristal externe haute vitesse (HSE).

Les principales caractéristiques sont :

• Taille combinée Rx et Tx FIFO de 1,25 Ko avec dimensionnement dynamique FIFO

• Prend en charge le protocole de demande de session (SRP) et le protocole de négociation d'hôte (HNP)

• 1 point de terminaison de contrôle bidirectionnel + 5 points de terminaison IN + 5 points de terminaison OUT

• 12 canaux hôtes avec prise en charge OUT périodique

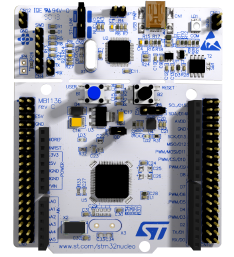
• HNP/SNP/IP à l'intérieur (pas besoin de résistance externe)

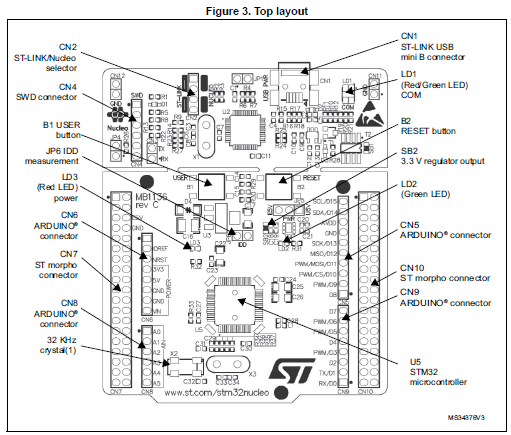
• Prise en charge USB 2.0 LPM (gestion de l'alimentation par liaison)

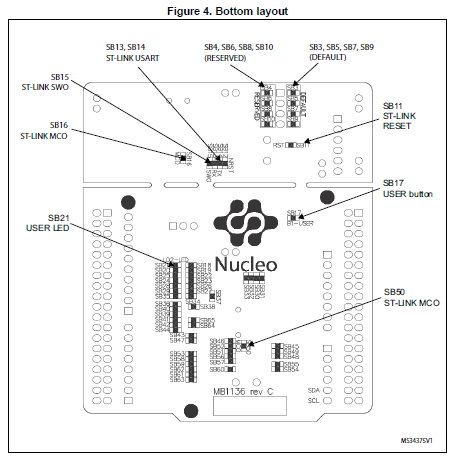
• Prise en charge de la révision 1.2 des spécifications de charge de la batterie

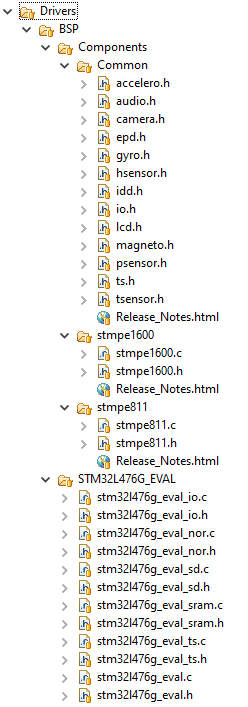
• Prise en charge interne FS OTG PHY

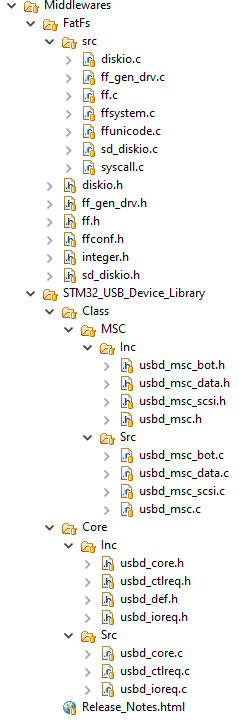
Pour les modes OTG/Host, un interrupteur d'alimentation est nécessaire dans le cas où des appareils alimentés par bus sont connectés.

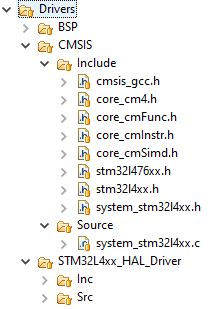


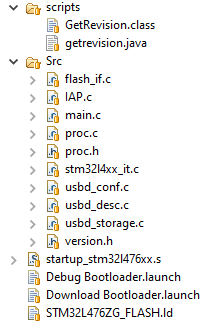


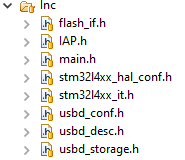


Structure du programme









FLASH.Id

Application

/\* Specify the memory areas \*/

MEMORY

{

RAM (xrw) : ORIGIN = 0x20000000, LENGTH = 96K

RAM2 (xrw) : ORIGIN = 0x10000000, LENGTH = 32K

FLASH (rx) : ORIGIN = 0x08020000, LENGTH = 896K

}

**Bootloader**

/\* Specify the memory areas \*/

MEMORY

{

FLASH (rx) : ORIGIN = 0x8000000, LENGTH = 128K

RAM (xrw) : ORIGIN = 0x20000000, LENGTH = 96K

}

Srs/flash\_if.c

/\* 1M0 flash 1 \* 1024 \* 1024 \*/

**#define** FLASH\_START\_ADRESS 0x08000000

**#define** FLASH\_PAGE\_NBPERBANK 256

**#define** FLASH\_BANK\_NUMBER 2