



eBook Gratuit

APPRENEZ stm32

eBook gratuit non affilié créé à partir des
contributeurs de Stack Overflow.

#stm32

Table des matières

À propos.....	1
Chapitre 1: Démarrer avec stm32.....	2
Remarques.....	2
Qu'est ce que le STM32?.....	2
Serie de produits.....	2
Conseils de développement.....	2
Versions.....	3
Exemples.....	3
Première configuration avec un exemple de clignotement de la LED à l'aide de la bibliothèque.....	3
Installation IDE.....	3
Créer un projet.....	3
Blink LED application.....	6
Chapitre 2: Environnements de développement intégrés (IDE).....	10
Introduction.....	10
Remarques.....	10
Exemples.....	13
SW4STM32: Workbench système pour STM32.....	13
introduction.....	13
Installation.....	14
IAR-EWARM.....	14
introduction.....	14
Installation.....	15
Atollic - TrueSTUDIO.....	15
introduction.....	16
Installation.....	16
Coide.....	16
introduction.....	16
Installation.....	17
Chapitre 3: UART - Récepteur / émetteur asynchrone universel (communication série).....	18
Introduction.....	18

Examples.....	18
Application Echo - Bibliothèque HAL.....	18
Transmettre une grande quantité de données à l'aide de DMA et d'interruptions - Bibliothèque.....	19
Crédits.....	23

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [stm32](#)

It is an unofficial and free stm32 ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official stm32.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec stm32

Remarques

Cette section fournit une vue d'ensemble de ce que stm32 est et de la raison pour laquelle un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les sujets importants dans stm32, et établir un lien avec les sujets connexes. La documentation de stm32 étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Qu'est ce que le STM32?

STM32 est une famille de microcontrôleurs Flash 32 bits développée par ST Microelectronics. Basé sur le processeur ARM® Cortex®-M, il offre une gamme de produits 32 bits alliant des performances en temps réel, un traitement du signal numérique et un fonctionnement basse tension à faible consommation.

Une description détaillée de chaque série, des outils de développement et du décodage des numéros de pièce se trouve sur [Wikipedia](#).

Serie de produits

	Cortex-M0 / -M0 +	Cortex-M3	Cortex-M4	Cortex-M7
Haute performance:		STM32F2	STM32F4	STM32F7 , STM32H7
Courant dominant:	STM32F0	STM32F1	STM32F3	
Ultra-basse consommation:	STM32L0	STM32L1	STM32L4	

Conseils de développement

	STM32 Nucleo (mbed activé)	Kits de découverte	Cartes d'évaluation
Cas d'utilisation typique:	Prototypage flexible, communauté	Prototypage, démos créatives	Évaluation complète des fonctionnalités
Possibilités	+++	++	++

	STM32 Nucleo (mbed activé)	Kits de découverte	Cartes d'évaluation
d'extension:			
Connectivité:	Arduino TM , ST, Morpho	ST	ST

Versions

Version	Date de sortie
1.0.0	2016-11-01

Exemples

Première configuration avec un exemple de clignotement de la LED à l'aide de la bibliothèque SW4STM32 et HAL

(**Remarque:** il existe de nombreux IDE, chaînes d'outils et bibliothèques prêtes à l'emploi avec STM32. La configuration suivante nécessite peu d'efforts pour que cela fonctionne, mais ce n'est qu'un exemple parmi d'autres. N'hésitez pas à en explorer d'autres, ce n'est pas le cas. Le but de cet exemple est de forcer quiconque à utiliser les outils qui seront utilisés ici.)

Installation IDE

[System Workbench for STM32](#) : IDE gratuit sous Windows, Linux et OS X. Conçu par [AC6](#) et disponible en téléchargement après enregistrement sur le [site Web de la communauté OpenSTM32](#) .

L'EDI lui-même est basé sur Eclipse, mais est livré avec quelques extras pour le développement STM32 comme:

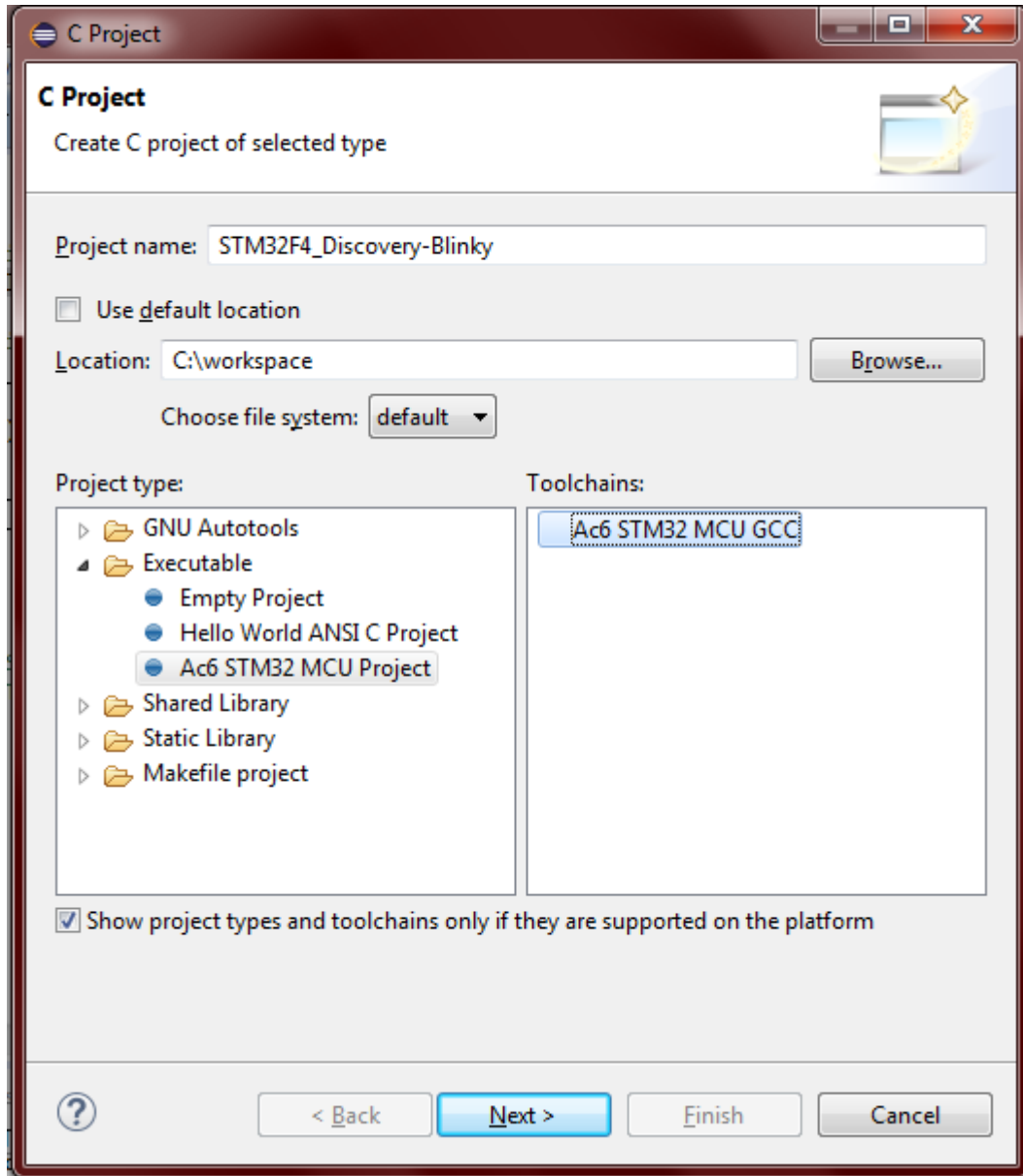
- Ac6 STM32 MCU GCC toolchain
- [OpenOCD](#) et GDB (arm-none-eabi-gdb) avec des configurations de débogage générées automatiquement en fonction de la carte cible
- Options intégrées pour programmer ou effacer les puces

Pour commencer avec STM32 avant de créer votre propre carte, il est recommandé d'expérimenter une [carte Discovery](#) , [Nucleo](#) ou [Eval](#) , fournie avec un programmeur / débogueur SWD (Serial Wire Debug) intégré appelé ST-Link.

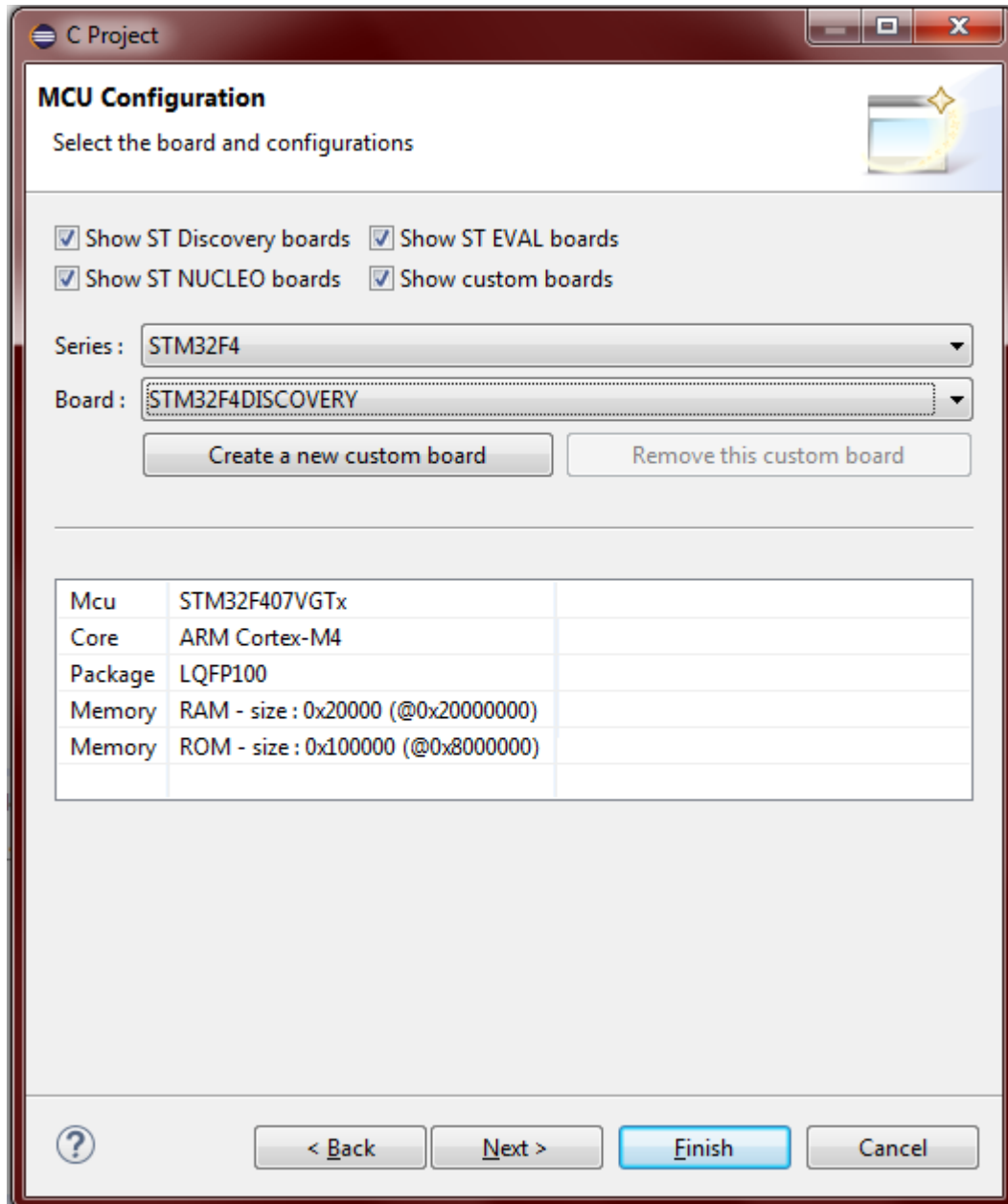
Créer un projet

Cet exemple utilisera un [kit STM32F4 Discovery](#) , qui comprend un microcontrôleur STM32F407VG. (Tout autre tableau peut également être utilisé.)

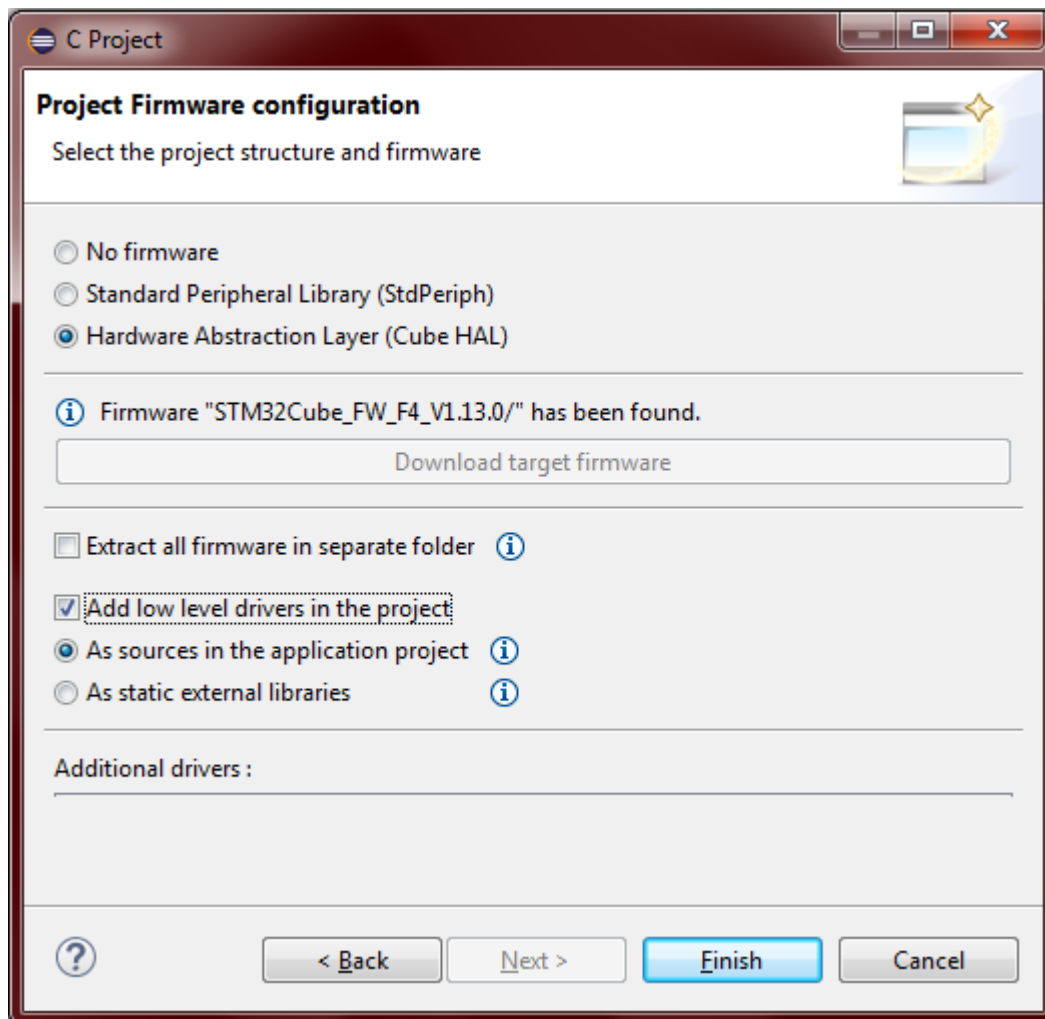
1. Ouvrez SW4STM32 et créez un nouveau projet C: **Fichier → Nouveau → Projet C**
2. Donnez-lui un nom comme "*STM32F4_Discovery-Blinky*" et dans la liste **Type de projet**, choisissez le **projet MCU Executable / Ac6 STM32** . Par défaut, la seule chaîne d'outils disponible est **Ac6 STM32 MCU GCC** . Cliquez sur Suivant.



3. La prochaine étape est les *paramètres de débogage / libération* , qui peuvent maintenant être ignorés en cliquant sur Suivant.
4. *Sélection du conseil* Les cartes existantes peuvent être sélectionnées comme dans cet exemple, la découverte STM32F4 ou de nouvelles cartes personnalisées peuvent être ajoutées.

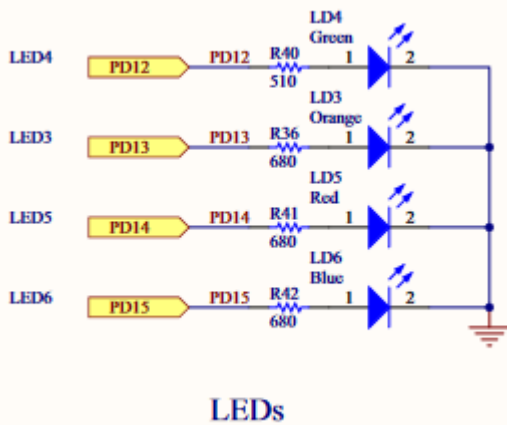


5. La prochaine étape est la *configuration du micrologiciel du projet* . Choisissez entre **Pas de microprogramme** , **Standard Peripheral Library (SPL)** ou **Hardware Abstraction Layer (HAL)** . On se demande laquelle est la plus adaptée au développement, mais cette question est hors de propos dans cet exemple. Cet exemple utilisera la bibliothèque HAL car elle est actuellement prise en charge par ST Microelectronics. [STM32CubeMX](#) est un outil logiciel supplémentaire disponible pour HAL, qui est un générateur de code d'initialisation. Plusieurs exemples d'applications sont également disponibles par les [progiciels STM32CubeFx](#) ou [STM32CubeLx](#). Téléchargez le micrologiciel cible s'il manque et il est recommandé de sélectionner les options "**Ajouter des pilotes de bas niveau dans le projet**" et "**Comme sources dans l'application**" . Enfin, cliquez sur Terminer.



Blink LED application

Comme ce projet a été créé avec une découverte STM32F4, il existe déjà plusieurs fonctions prêtes à l'emploi sous le **dossier / STM32F4_Discovery-Blinky / Utilities / STM32F4-Discovery** / pouvant être utilisées pour interfacer les périphériques du kit Discovery (accéléromètre, audio). , LED, bouton poussoir). Dans cet exemple, les fonctions `void BSP_LED_Init(Led_TypeDef Led)` et `void BSP_LED_Toggle(Led_TypeDef Led)` seront utilisées à partir du fichier `stm32f4_discovery.c` pour faire clignoter la LED verte, qui est `LED4` . Pour décider quelle est la LED qui utilise les schémas du [kit Discovery](#) .



Les noms de broches et de ports réels sont déjà masqués par certains `#define` et `enum`, utilisez *Ctrl* + *Click* pour les suivre.

1. Dans la `main`, appelez la fonction `HAL_Init()` qui réinitialise tous les périphériques, initialise l'interface Flash et le SysTick. (SysTick sera utilisé pour générer un délai pour le clignotement.)
2. L'horloge du système doit être configurée. Cela peut être fait en utilisant la [fonction de configuration d'horloge STM32CubeMX](#) ou par le manuel de référence. Dans cet exemple, l'horloge système est alimentée par la boucle PLL (Phase Locked Loop) interne, alimentée par un oscillateur à cristal externe (HSE) de 8 MHz. Des pré-gradateurs ont été configurés pour atteindre la fréquence maximale disponible, qui est de 168 MHz dans le cas de la découverte F4.
3. Initialisation des périphériques, dans ce cas une broche GPIO.
4. Dans une boucle sans fin, appelez la bascule LED et la fonction `HAL_Delay()`. `HAL_Delay()` utilise `SysTick` et génère un délai en millisecondes.

Le code entier est le suivant:

```
#include "stm32f4xx.h"
#include "stm32f4_discovery.h"

void SystemClock_Config(void);

int main(void)
{
    /* Reset of all peripherals, Initializes the Flash interface and the SysTick. */
    HAL_Init();

    /* Configure the system clock */
    SystemClock_Config();

    /* Initialize one of the LED GPIO pin */
    BSP_LED_Init(LED4);

    while(1)
    {
        BSP_LED_Toggle(LED4);
        HAL_Delay(1000);    // in milliseconds
    }
}
```

```

}

/**
 * @brief System Clock Configuration
 * The system Clock is configured as follow :
 *
 * System Clock source           = PLL (HSE)
 * SYSCLK(Hz)                     = 168000000
 * HCLK(Hz)                       = 168000000
 * AHB Prescaler                  = 1
 * APB1 Prescaler                 = 4
 * APB2 Prescaler                 = 2
 * HSE Frequency(Hz)              = HSE_VALUE
 * PLL_M                          = (HSE_VALUE/1000000u)
 * PLL_N                          = 336
 * PLL_P                          = 2
 * PLL_Q                          = 7
 * VDD(V)                         = 3.3
 * Main regulator output voltage  = Scale1 mode
 * Flash Latency(WS)              = 5
 *
 * @param None
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_ClkInitTypeDef RCC_ClkInitStruct;
    RCC_OscInitTypeDef RCC_OscInitStruct;

    // Enable Power Control clock
    __PWR_CLK_ENABLE();


    // The voltage scaling allows optimizing the power consumption when the
    // device is clocked below the maximum system frequency, to update the
    // voltage scaling value regarding system frequency refer to product
    // datasheet.
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    // Enable HSE Oscillator and activate PLL with HSE as source
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;


    // This assumes the HSE_VALUE is a multiple of 1MHz. If this is not
    // your case, you have to recompute these PLL constants.
    RCC_OscInitStruct.PLL.PLLM = (HSE_VALUE/1000000u);
    RCC_OscInitStruct.PLL.PLLN = 336;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 7;
    HAL_RCC_OscConfig(&RCC_OscInitStruct);

    // Select PLL as system clock source and configure the HCLK, PCLK1 and PCLK2
    // clocks dividers
    RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
        | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
    HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);
}

```

Construire avec le marteau  et téléchargez l'application en cliquant avec le bouton droit de la souris sur le dossier du projet et en sélectionnant l'option **Cible → Program chip**

Une autre façon de télécharger consiste à utiliser le *débogage* . Pour ce faire, cliquez sur la flèche

à côté de l'icône du bug  dans la barre d'outils et ouvrez le menu **Configuration du débogage** . Créez une nouvelle configuration de **débogage Ac6 STM32** et, si le champ **Application C / C ++** est vide, remplissez les champs suivants:

Debug \ STM32F4_Discovery-Blinky.elf

D'autres paramètres de débogage tels que le fichier de configuration OpenOCD et les ports Telnet et GDB utilisés sont automatiquement générés et complétés par la structure. Enfin, cliquez sur le bouton Déboguer.

Lire Démarrer avec stm32 en ligne: <https://riptutorial.com/fr/stm32/topic/7617/demarrer-avec-stm32>

Chapitre 2: Environnements de développement intégrés (IDE)

Introduction

L'objectif de cette rubrique est de répertorier tous les environnements de développement intégrés (IDE) pouvant être utilisés pour développer des logiciels pour les microcontrôleurs STM32. Les exemples doivent contenir: 1. Liste des principales fonctionnalités de l'EDI. 2. Liste des systèmes d'exploitation pris en charge par l'EDI. 3. Processus d'installation 4. Étapes de configuration supplémentaires (le cas échéant).

Remarques

Les IDE répertoriés par ST Microelectronics:

Numéro d'article	Description générale	État du marketing	Fournisseur	Type de logiciel
Coide	CooCox CoIDE, un environnement de développement logiciel gratuit et hautement intégré pour les MCU Cortex ARM	actif	CooCox	Suites de développement SW
Cosmicide	Outils de développement croisé Cosmic ARM / Cortex "M" pour microcontrôleur STM32	actif	Cosmique	Suites de développement SW
CrossWorks	Rowley Associates CrossWorks, environnement de développement intégré avec téléchargement et débogage JTAG Flash	actif	Rowley	Suites de développement SW
DS-5	ARM Development Studio 5 (DS-5) fournit les meilleurs outils pour la plus large	actif	BRAS	Suites de développement SW

Numéro d'article	Description générale	État du marketing	Fournisseur	Type de logiciel
	gamme de plates-formes basées sur un processeur ARM			
EMP-Thunder	Emprog ThunderBench, des outils de développement C / C ++ entièrement intégrés et bien conçus pour ARM Cortex	actif	Emprog	Firmware
Hitop5	Interface utilisateur universelle, IDE et débogueur pour tous les outils de développement Hitex	actif	Hitex	Suites de développement SW
IAR-EWARM	Environnement de développement intégré IAR et optimisation du compilateur C / C ++ pour ARM Cortex-M	actif	IAR	Suites de développement SW
MDK-ARM-STM32	Environnement de développement logiciel MDK-ARM pour les MCU basés sur Cortex-M	actif	Keil	Suites de développement SW
MULTI	Environnement de développement et de débogage intégré GreenHills pour les applications intégrées utilisant C et C ++	actif	Logiciel GreenHills	Suites de développement SW
Men-Nucleus-SF	Nucleus SmartFit pour STM32	actif	Mentor Graphics	Firmware
PER-traçeur	Analyseur de trace d'exécution Percepio pour STM32 MCU	actif	Percepio	
PLSUDE-STM32	Plate-forme de débogage et	actif	Pls	Suites de développement

Numéro d'article	Description générale	État du marketing	Fournisseur	Type de logiciel
	d'émulateur avec prise en charge optimisée de la trace et de la mémoire Flash pour les MCU basés sur STM32 Cortex-M par les outils de développement PLS			SW
RIDE-STM32	Environnement de développement intégré de marque Raisonance pour les MCU STM32	actif	Raisonance	Suites de développement SW
SOMN-DRT-IDE	SOMNIUM DRT Cortex-M IDE	actif	SOMNIUM	Suites de développement SW
SW4STM32	System Workbench pour STM32: IDE gratuit sous Windows, Linux et OS X	actif	AC6	Suites de développement SW
TASKINGVX-STM32	Les outils de compilation et de débogage C / C ++ d'Altium pour les MCU basés sur ARM	actif	TACHER	Firmware
TrueSTUDIO	Le premier outil de développement C / C ++ pour le développement de STM32, avec ses fonctionnalités inégalées et son intégration sans précédent	actif	Atollique	Suites de développement SW
iSYS-winIDEAOpen	La plate-forme de développement de logiciels illimitée et gratuite d'iSYSTEM pour tous les appareils basés sur STM32	actif	iSYSTEM	Suites de développement SW

Numéro d'article	Description générale	État du marketing	Fournisseur	Type de logiciel
	Cortex-M			
MikroBasicPRO	MikroElektronika, le compilateur de base complet qui rend le développement STM32 adapté à tous	actif	Mikroelectronika	Suites de développement SW
MikroCPRO	MikroElektronika, le compilateur complet ANSI C pour les appareils STM32. Il dispose d'un IDE intuitif, puissant compilateur avec optimisations avancées	actif	Mikroelectronika	Suites de développement SW
MikroPascalPRO	MikroElektronika compilateur Pascal complet pour les périphériques STM32. Il dispose d'un IDE intuitif avec prise en charge de la station d'accueil, riche en fonctionnalités, éditeur de texte avancé, de nombreux outils, bibliothèques et exemples disponibles.	actif	Mikroelectronika	Suites de développement SW
winIDEA-STM32	Solution complète de développement logiciel et de test d'iSYSTEM pour les MCU STM32	actif	iSYSTEM	Firmware

Exemples

SW4STM32: Workbench système pour STM32

introduction

System Workbench for STM32 est un IDE gratuit sous Windows, Linux et OS X. Description de [ST Microelectronics](#) :

La chaîne d'outils System Workbench, appelée SW4STM32, est un environnement de développement logiciel multi-OS gratuit basé sur Eclipse, qui prend en charge la gamme complète des microcontrôleurs STM32 et des cartes associées.

La chaîne d'outils SW4STM32 peut être obtenue sur le site Web www.openstm32.org, qui comprend des forums, des blogs et des formations pour le support technique. Une fois inscrits sur ce site, les utilisateurs recevront les instructions d'installation sur la page Documentation> System Workbench pour procéder au téléchargement de la chaîne d'outils gratuite.

La chaîne d'outils System Workbench et son site Web collaboratif ont été conçus par AC6, une société de services spécialisée dans la formation et le conseil sur les systèmes embarqués.

Ce produit est fourni par un tiers non affilié à ST. Pour les dernières informations sur le cahier des charges, consultez le site internet du tiers: www.ac6.fr.

Principales caractéristiques

- Prise en charge complète des microcontrôleurs STM32, des cartes Nucleo STM32, des kits Discovery et des cartes d'évaluation, ainsi que du microprogramme STM32 (bibliothèque de périphériques standard ou STM32Cube HAL)
- Compilateur GCC C / C ++
- Débogueur basé sur GDB
- Eclipse IDE avec gestion du travail en équipe
- Compatible avec les plug-ins Eclipse
- Support ST-LINK
- Aucune limite de taille de code
- Prise en charge de plusieurs systèmes d'exploitation: Windows®, Linux et OS X®

Installation

1. Allez à: <http://www.openstm32.org/HomePage> .
2. Inscrivez-vous et connectez-vous au site.
3. Accédez à:
<http://www.openstm32.org/Downloading+the+System+Workbench+for+STM32+installer> .
4. Téléchargez la dernière version pour votre système d'exploitation.
5. Exécutez le programme d'installation téléchargé.

IAR-EWARM

introduction

IAR-EWARM est une suite de développement de logiciels fournie avec des fichiers de configuration de périphérique, des chargeurs flash et 4300 exemples de projets prêts à l'emploi. IAR Embedded Workbench est compatible avec les autres compilateurs compatibles ARM®EABI et prend en charge les scores ARM® suivants pour STM32:

Principales caractéristiques

1. Composants clés:

- Environnement de développement intégré avec outils de gestion de projet et éditeur
- Optimisation du compilateur C et C ++ pour ARM®
- Vérification automatique des règles MISRA C (MISRA C: 2004)
- Conformité ARM® EABI et CMSIS
- Prise en charge complète du système cible de matériel
- Sondes de débogage en circuit optionnel J-jet et JTAGjet™ -Trace
- Débogage de l'alimentation pour visualiser la consommation d'énergie en corrélation avec le code source
- Bibliothèques d'exécution, y compris le code source
- Déplacement de l'assembleur ARM®
- Outils de liaison et de bibliothécaire
- Débogueur C-SPY® avec simulateur ARM®, prise en charge de JTAG et prise en charge du débogage compatible RTOS sur le matériel
- Plugins RTOS disponibles auprès des fournisseurs IAR Systems et RTOS
- Plus de 3100 exemples de projets pour les cartes d'évaluation de nombreux fabricants différents
- Guides d'utilisation et de référence au format PDF
- Aide en ligne contextuelle

2. Support spécifique à la puce:

- 4300 exemples de projets, y compris pour les cartes d'évaluation STMicroelectronics
- Prise en charge d'applications 4 Go en mode ARM® et Thumb®
- Chaque fonction peut être compilée en mode ARM® ou Thumb®
- Génération de code co-processeur VFP Vector Floating Point
- Prise en charge de NEON™ intrinsèque

3. Prise en charge du débogage matériel:

- STMicroelectronics ST-LINK V2: prend en charge les périphériques STM32
- STMicroelectronics ST-LINK: prend en charge les périphériques STM32

4. Support RTOS: consultez le site Web de l'IAR <http://www.iar.com>

5. Appareils pris en charge: consultez le site Web d'IAR <http://www.iar.com>

Installation

Atollic - TrueSTUDIO

introduction

C / C ++ IDE pour le développement d'ARM.

Atollic TrueSTUDIO® est testé et vérifié sur les systèmes d'exploitation suivants:

- Microsoft® Windows®Vista (version 32 bits)
- Microsoft® Windows® Vista (version 64 bits)
- Microsoft® Windows® 7 (version 32 bits)
- Microsoft® Windows® 7 (version 64 bits)
- Microsoft® Windows® 8 (version 64 bits)
- Microsoft® Windows® 10 (version 64 bits)
- Prise en charge de Linux prévue fin 2016 Q4
- Prise en charge de Mac OS X attendue pour le deuxième trimestre 2017

TrueSTUDIO est uniquement disponible en tant qu'application **32 bits** .

Installation

Le produit Atollic TrueSTUDIO est livré en tant qu'installateur exécutable. Assurez-vous que le compte d'utilisateur à partir duquel le programme d'installation est lancé possède des privilèges d'administrateur. Il n'y a pas besoin d'enregistrement ou de connexion Internet pendant l'installation. Lorsque TrueSTUDIO est installé, il fonctionnera en mode Lite si aucune licence n'est détectée.

1. Allez à: <http://atollic.com/resources/downloads/> .
2. Téléchargez la dernière version stable ou la dernière version bêta.
3. Exécutez le programme d'installation.

Coide

introduction

CooCox ColDE, un environnement de développement logiciel gratuit et hautement intégré pour les MCU Cortex ARM. Description de [ST Microelectronics](#) :

ColDE est un environnement de développement de logiciels gratuits basé sur la chaîne d'outils Eclipse et GCC, qui a été personnalisé et simplifié pour permettre aux utilisateurs d'accéder facilement aux microcontrôleurs ARM® Cortex®-M.

Ce produit est fourni par un tiers non affilié à ST. Pour obtenir des informations complètes et à jour sur les spécifications et les packages des pièces achetées, consultez le site Web du tiers www.coocox.org.

Principales caractéristiques

- Prise en charge complète des microcontrôleurs STM32, des cartes Nucleo

STM32 et des bibliothèques logicielles STM32Cube.

- Compilateur GCC C / C ++.
- Débogueur basé sur GDB.
- IDE Eclipse simplifié.
- Support ST-Link.
- Prise en charge multilingue: anglais, chinois.

Installation

Lire Environnements de développement intégrés (IDE) en ligne:

<https://riptutorial.com/fr/stm32/topic/7741/envIRONNEMENTS-DE-DEVELOPPEMENT-INTEGRES--IDE->

Chapitre 3: UART - Récepteur / émetteur asynchrone universel (communication série)

Introduction

Cette rubrique concerne la communication série à l'aide du périphérique UART (Universal Asynchronous Receiver / Transmitter) des microcontrôleurs STM32.

Exemples

Application Echo - Bibliothèque HAL

Dans cet exemple, le microcontrôleur renvoie les octets reçus à l'expéditeur à l'aide de l'interruption UART RX.

```
#include "stm32f4xx.h"

UART_HandleTypeDef huart2;

/* Single byte to store input */
uint8_t byte;

void SystemClock_Config(void);

/* USART2 Interrupt Service Routine */
void USART2_IRQHandler(void)
{
    HAL_UART_IRQHandler(&huart2);
}

/* This callback is called by the HAL_UART_IRQHandler when the given number of bytes are
received */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if (huart->Instance == USART2)
    {
        /* Transmit one byte with 100 ms timeout */
        HAL_UART_Transmit(&huart2, &byte, 1, 100);

        /* Receive one byte in interrupt mode */
        HAL_UART_Receive_IT(&huart2, &byte, 1);
    }
}

void uart_gpio_init()
{
    GPIO_InitTypeDef GPIO_InitStruct;

    __GPIOA_CLK_ENABLE();

    /**USART2 GPIO Configuration
    PA2      -> USART2_TX
    */
}
```

```

PA3      -----> USART2_RX
*/
GPIO_InitStruct.Pin = GPIO_PIN_2 | GPIO_PIN_3;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}

void uart_init()
{
    __USART2_CLK_ENABLE();

    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    HAL_UART_Init(&huart2);

    /* Peripheral interrupt init*/
    HAL_NVIC_SetPriority(USART2_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(USART2_IRQn);
}

int main(void)
{
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    uart_gpio_init();
    uart_init();

    HAL_UART_Receive_IT(&huart2, &byte, 1);

    while(1)
    {

    }
}

```

Cet exemple utilise une découverte STM32F4 (STM32F407VG), les valeurs de GPIO et les autres fonctions doivent être modifiées en fonction du microcontrôleur STM32 utilisé.

Transmettre une grande quantité de données à l'aide de DMA et d'interruptions - Bibliothèque HAL

Dans cet exemple, 2000 octets seront transférés à l'aide de DMA, les interruptions de **transmission semi-complète** et de **transmission complète** obtiendront les meilleures performances.

La première moitié du tampon de transmission est chargée par la CPU de nouvelles données dans le rappel d'interruption **semi-complet transmis** alors que la seconde moitié du tampon est

transmise par le DMA en arrière-plan.

Ensuite, dans la **transmission terminée**, la seconde moitié du tampon de transmission est chargée par les nouvelles données par le processeur tandis que la première moitié (mise à jour précédemment) est transmise par le DMA en arrière-plan.

```
#include "stm32f4xx.h"

uint8_t dma_buffer[2000];
volatile uint8_t toggle = 0;

UART_HandleTypeDef huart2;
DMA_HandleTypeDef hdma_usart2_tx;

void uart_gpio_init()
{
    GPIO_InitTypeDef GPIO_InitStruct;

    __GPIOA_CLK_ENABLE();

    /**USART2 GPIO Configuration
    PA2      -----> USART2_TX
    PA3      -----> USART2_RX
    */
    GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}

void uart_dma_init()
{
    /* DMA controller clock enable */
    __DMA1_CLK_ENABLE();

    /* Peripheral DMA init*/
    hdma_usart2_tx.Instance = DMA1_Stream6;
    hdma_usart2_tx.Init.Channel = DMA_CHANNEL_4;
    hdma_usart2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
    hdma_usart2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_usart2_tx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_usart2_tx.Init.PeriphDataAlignment = DMA_MDATAALIGN_BYTE;
    hdma_usart2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
    hdma_usart2_tx.Init.Mode = DMA_NORMAL;
    hdma_usart2_tx.Init.Priority = DMA_PRIORITY_LOW;
    hdma_usart2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    HAL_DMA_Init(&hdma_usart2_tx);

    __HAL_LINKDMA(&huart2, hdmactx, hdma_usart2_tx);

    /* DMA interrupt init */
    HAL_NVIC_SetPriority(DMA1_Stream6_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA1_Stream6_IRQn);
}

void uart_init()
{
    __USART2_CLK_ENABLE();
```

```

huart2.Instance = USART2;
huart2.Init.BaudRate = 115200;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
HAL_UART_Init(&huart2);

/* Peripheral interrupt init*/
HAL_NVIC_SetPriority(USART2_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(USART2_IRQn);
}

/* This function handles DMA1 stream6 global interrupt. */
void DMA1_Stream6_IRQHandler(void)
{
    HAL_DMA_IRQHandler(&hdma_usart2_tx);
}

void USART2_IRQHandler(void)
{
    HAL_UART_IRQHandler(&huart2);
}

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
    uint16_t i;
    toggle = !toggle;

    for(i = 1000; i < 1998; i++)
    {
        if(toggle)
            dma_buffer[i] = '&';
        else
            dma_buffer[i] = 'z';
    }

    dma_buffer[1998] = '\r';
    dma_buffer[1999] = '\n';
}

void HAL_UART_TxHalfCpltCallback(UART_HandleTypeDef *huart)
{
    uint16_t i;

    for(i = 0; i < 1000; i++)
    {
        if(toggle)
            dma_buffer[i] = 'y';
        else
            dma_buffer[i] = '|';
    }
}

int main(void)
{
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

```



```

uart_gpio_init();
uart_dma_init();
uart_init();

uint16_t i;

for(i = 0; i < 1998; i++)
{
    dma_buffer[i] = 'x';
}

dma_buffer[1998] = '\r';
dma_buffer[1999] = '\n';

while(1)
{
    HAL_UART_Transmit_DMA(&huart2, dma_buffer, 2000);
}
}

```

L'exemple a été écrit pour une carte STM32F4 Discovery (STM32F407VG). L'instance DMA appropriée, le canal UART-DMA, le GPIO et les autres paramètres de fonction doivent être modifiés en fonction du microcontrôleur STM32 utilisé.

Lire UART - Récepteur / émetteur asynchrone universel (communication série) en ligne:

<https://riptutorial.com/fr/stm32/topic/9707/uart---recepteur---emetteur-asynchrone-universel--communication-serie->

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec stm32	Bence Kaulics , Community
2	Environnements de développement intégrés (IDE)	Bence Kaulics
3	UART - Récepteur / émetteur asynchrone universel (communication série)	Bence Kaulics