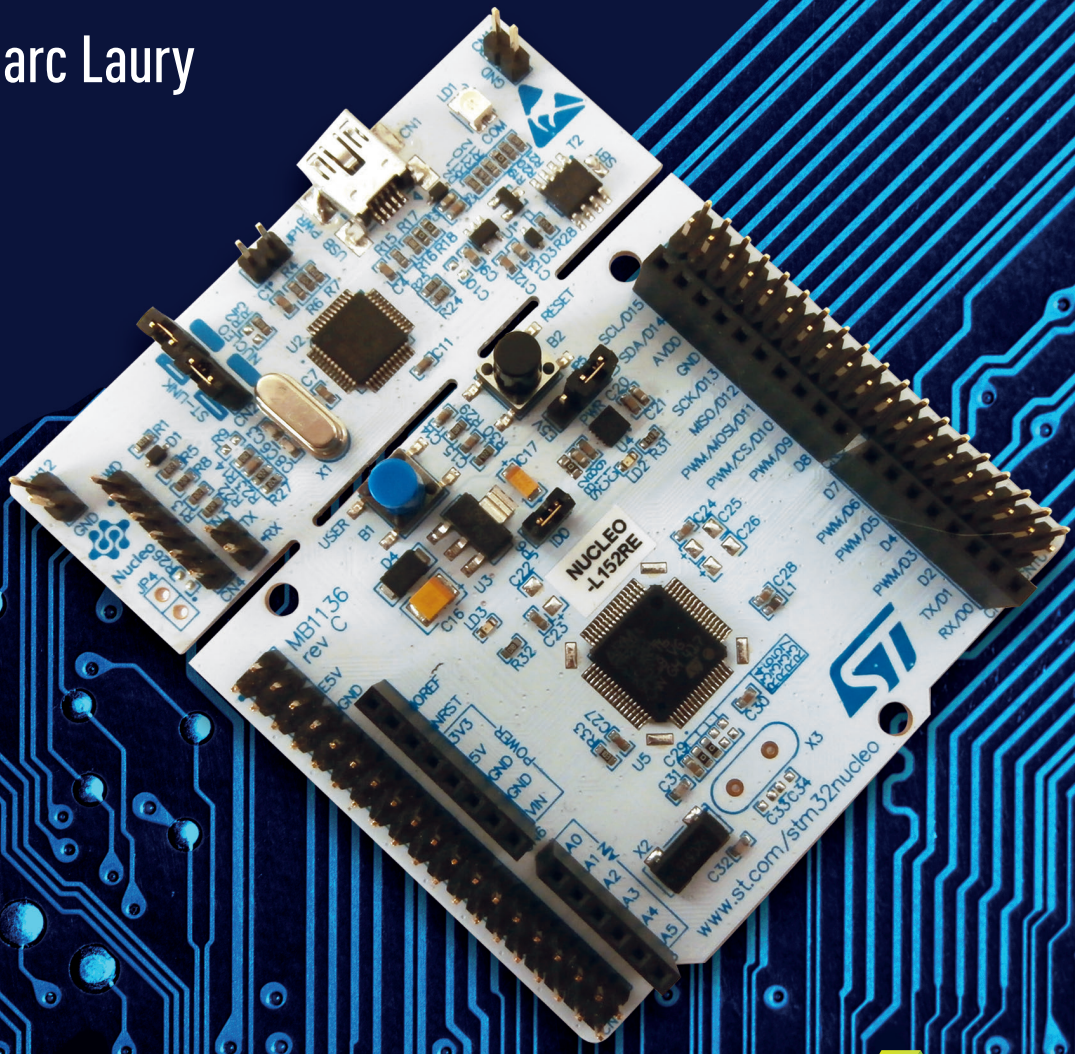
À LA DÉCOUVERTE
DES CARTES NUCLEO

Marc Laury



EYROLLES

SERIAL MAKERS

Des cartes performantes et multifonctions

Tout comme les cartes Arduino, les cartes Nucleo sont des cartes de développement dotées d'un microcontrôleur et de différents connecteurs qui leur permettent d'être facilement programmables sans connaissances préalables. La grande force de ces cartes, c'est leur microcontrôleur, le STM32, qui associe rapidité et faible consommation. Autre atout, elles peuvent être combinées à du Wi-Fi, du bluetooth ou de l'Ethernet, ce qui leur offre un champ d'action très étendu (prototypage, domotique, Internet des objets…).

Cet ouvrage se propose de vous faire découvrir ces cartes au fort potentiel à travers différents projets pratiques : allumage et extinction d'une LED, gestion d'afficheurs 7 segments et TFT, jeu de lumières, mesure de tension, capteur de température, compteur, alarme, saisie d'un texte au clavier et affichage sur écran… Pour compléter votre apprentissage, tous les exemples de programmes figurant dans le livre sont téléchargeables à l'adresse *www.editions-eyrolles.com/go/nucleo*.

À qui s'adresse ce livre ?

Aux amateurs d'électronique, étudiants, makers, ingénieurs, geeks…

Sur www.editions-eyrolles.com/go/nucleo

- Téléchargez les exemples de programmes du livre
- Dialoguez avec l'auteur

Au sommaire

Les microcontrôleurs STM32 • Les cartes Nucleo-64 • Environnement de développement pour STM32 • Mise au point d'un programme avec Keil µVision5 • Contrôleur de LED par GPIO • Port virtuel par USB • Afficheur 7 segments par liaison I2C • Jeu de lumières par liaison SPI • Mesure d'une tension par l'ADC • Capteur de température interne sous l'environnement SW4STM32 • Lumière progressive avec un compteur • Interruption par compteur • Alarme par RTC • Modes faible consommation • Application temps réel • Gestion d'un afficheur TFT • Contrôleur de carte SD • Environnement de développement MBED.

Passionné depuis son plus jeune âge par l'électronique et l'informatique, **Marc Laury** travaille depuis 20 ans dans la société STMicroelectronics, spécialisée dans la fabrication de semi-conducteurs. Après un premier ouvrage sur le microcontrôleur ST623X (éditions Dunod) et de nombreux articles dans des revues d'électronique, il a entrepris d'écrire ce livre pour faire partager son expérience des cartes Nucleo.

SERIAL MAKERS
www.serialmakers.com

www.editions-eyrolles.com

À LA DÉCOUVERTE
DES CARTES NUCLEO

Dans la collection « Serial Makers »

C. Platt. – **L'électronique en pratique (2e édition).**
N° 14425, 2016, 328 pages.

C. Platt. – **L'électronique en pratique 2.**
N° 14179, 2015, 336 pages.

S. Monk. – **Mouvement, lumière et son avec Arduino et Raspberry Pi.**
N° 11807, 2016, 352 pages.

E. Bartmann. – **Le grand livre d'Arduino (2e édition).**
N° 14117, 2015, 612 pages.

E. De Keyser. – **Filmer et photographier avec un drone (2e édition).**
N° 67435, 2017, 224 pages.

F. Botton. – **Les drones de loisir (3e édition).**
N° 67444, 2017, 224 pages.

R. Jobard. – **Les drones (3e édition).**
N° 67434, 2017, 200 pages.

C. Bosqué, O. Noor et L. Ricard. – **FabLabs, etc.** *Les nouveaux lieux de fabrication numérique.*
N° 13938, 2015, 216 pages.

J. Boyer. – **Réparez vous-même vos appareils électroniques.**
N° 13936, 2014, 384 pages.

A. Banks, MacUSer et iFixit. – **Réparez vous-même votre Apple.**
N° 14251, 2015, 146 pages.

Marc Laury

À LA DÉCOUVERTE DES CARTES NUCLEO

EYROLLES

SERIAL MAKERS

ÉDITIONS EYROLLES
61 bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

REMERCIEMENTS

Je tiens tout d'abord à remercier STMicroelectronics, société pour laquelle je travaille depuis vingt ans, et sans laquelle ce projet n'aurait pas été rendu possible.

Merci à Caroline, mon épouse, pour sa patience durant ces longs week-ends d'écriture et pour son soutien sans faille.

Enfin, un grand merci à Liliane, Béatrice et André d'avoir accepté de jouer les néophytes sur les exercices proposés dans ce livre. Leurs retours très pertinents m'ont été indispensables.

AVANT-PROPOS

Les cartes Nucleo, conçues autour de la famille des microcontrôleurs STM32, existent depuis plus de trois ans maintenant et sont utilisées par de nombreux électroniciens, amateurs comme professionnels. Elles font aussi leur apparition dans les universités. L'intérêt de ces cartes est de présenter un facteur de forme unique quel que soit le STM32 utilisé, avec des connecteurs capables de recevoir des cartes d'extension compatibles Arduino. De plus, les outils de développement sont gratuits et simples d'utilisation, ce qui facilite le codage des applications avec les cartes Nucleo.

Pourquoi ce livre ?

Il existe de nombreux documents traitant des cartes Nucleo et de la famille des microcontrôleurs STM32, écrits pour la plupart par la société STMicroelectronics qui a conçu ces deux produits. Par ailleurs, sur Internet, de nombreux forums et sites amateurs donnent des conseils et aident à résoudre certains problèmes rencontrés sur le sujet. Néanmoins, il n'existait, jusqu'à aujourd'hui, aucun livre qui donne une vue globale sur le sujet et qui explique comment développer étape par étape une application du début jusqu'à la fin.

Cet ouvrage fait le tour complet des possibilités offertes par les cartes Nucleo et le STM32, sans prétendre tout détailler mais en donnant au lecteur une idée de toutes les possibilités offertes par cette plate-forme afin qu'il puisse approfondir tel ou tel sujet s'il le souhaite.

À qui s'adresse-t-il ?

Ce livre s'adresse avant tout aux personnes qui souhaitent mettre en œuvre des applications électroniques à faible coût, mais qui hésitent faute d'avoir un mode d'emploi simple, complet et progressif. Cet ouvrage les accompagnera tout au long de leur développement et les aidera ainsi à faire leurs premiers pas avec les cartes Nucleo, bon marché, faciles à trouver chez la plupart des distributeurs et accompagnées d'outils de développement conviviaux et gratuits.

Les personnes plus avancées sur le sujet découvriront aussi certains détails qu'ils ignoraient et synthétiseront leurs connaissances souvent acquises « sur le tas ».

De plus, les électroniciens amateurs comme professionnels qui utilisent des cartes Arduino ou développent des applications pour celles-ci apprendront à passer facilement sur les cartes Nucleo, qui ont l'avantage de posséder des connecteurs compatibles tout en offrant plus de fonctionnalités et de meilleures performances.

Comment est-il structuré ?

Après une introduction sur la famille des microcontrôleurs STM32, nécessaire pour appréhender les cartes Nucleo, ce livre détaille l'architecture de celles-ci puis décrit un ensemble d'outils disponibles pour développer des applications avec de telles cartes. Les principales fonctionnalités du STM32 sont ensuite expliquées, de la simple entrée-sortie jusqu'à une application temps réel. À chaque fonctionnalité est associé un exemple simple dont le développement est expliqué pas à pas du début jusqu'à sa mise en œuvre sur une carte Nucleo.

Conventions adoptées

Tout au long de ce livre, certaines conventions ont été adoptées :

- Les termes anglais sont mis en italique et leur traduction en français est indiquée entre parenthèses ; par exemple, IDE *Integrated Development Environment* (environnement de développement).
- La traduction n'apparaît que la première fois que le terme est rencontré.
- Le mot «utilisateur» décrit la personne qui va développer une application, écrire un programme pour le STM32 de la carte Nucleo, puis le mettre au point.
- Les chapitres d'exemple sont tous structurés de la même façon :
  1. une introduction sur la fonction étudiée ;
  2. l'initialisation de cette fonction dans STM32CubeMX et la génération de son code C d'initialisation ;
  3. l'ajout du code applicatif à l'aide d'un environnement de développement ;
  4. la programmation de la carte Nucleo-L152RE et l'essai de l'application.

Pour aller plus loin

Sur *www.editions-eyrolles.com/go/nucleo*, les lecteurs peuvent télécharger tous les programmes donnés en exemples dans le livre. S'ils ont des remarques ou des questions à transmettre à l'auteur, ils peuvent lui écrire à l'adresse *nucleo.laury@free.fr*.

TABLE DES MATIÈRES

INTRODUCTION

Cet ouvrage consacré aux cartes Nucleo est composé de 18 chapitres. Les quatre premiers décrivent la famille STM32, les cartes Nucleo et les différents outils de développement disponibles. Les fonctionnalités du STM32 sont ensuite détaillées tout au long du livre et mises en œuvre sur la carte Nucleo au travers d'exemples pratiques.

Il est recommandé de lire ces quatre premiers chapitres avant de s'intéresser aux suivants, afin d'acquérir les connaissances nécessaires pour la mise en œuvre des applications proposées. En revanche, le lecteur pourra ensuite se reporter directement au chapitre traitant de la fonctionnalité qui l'intéresse indépendamment des autres chapitres. Lorsque certaines procédures se retrouvent à plusieurs endroits, le lecteur trouvera toujours un résumé avec la référence au chapitre détaillant le sujet en question.

Afin de mieux comprendre l'architecture de la carte Nucleo, il est conseillé de posséder des connaissances élémentaires sur le composant qui en est le cœur, à savoir le microcontrôleur STM32. Le chapitre 1 présente la famille STM32 et les séries qui existent actuellement. Il décrit sa structure interne dont le cœur est le Cortex-M commercialisé par la société ARM, ses fonctionnalités, ses périphériques, avant de détailler le STM32L152RET6 qui est le microcontrôleur implanté sur la carte Nucleo-L152RE utilisée dans les exemples de ce livre.

Le chapitre 2 expose l'architecture de la carte Nucleo-64, qui est commune quel que soit le STM32 qu'elle possède. Ses caractéristiques générales et toutes ses fonctionnalités sont décrites, ainsi que ses connecteurs compatibles avec les cartes d'extension Arduino. Ce chapitre se termine enfin par une présentation de la carte Nucleo-L152RE utilisée dans ce livre.

Les architectures matérielles du STM32 et de la carte Nucleo étant acquises, le lecteur pourra ensuite découvrir dans le chapitre 3 l'environnement de développement utilisé. Le logiciel gratuit STM32CubeMX y est décrit, ainsi que sa puissante bibliothèque STM32Cube qui permet de configurer graphiquement une carte Nucleo pour l'application souhaitée, le logiciel générant ensuite automatiquement le code C d'initialisation dans plusieurs environnements de développement intégrés.

Le code d'initialisation de l'application étant généré, l'utilisateur peut ensuite ajouter son code spécifique au travers d'un environnement de développement intégré. Le chapitre 4 présente l'un d'eux : Keil µVision5, qui est un IDE (*Integrated Development Environment,* environnement de développement intégré) utilisé dans la suite de ce livre. Ce logiciel est gratuit pour une utilisation jusqu'à 32 Ko de programme en mémoire, ce qui suffit pour la majorité des applications. Après avoir décrit comment configurer Keil µVision5, ce chapitre détaille les outils disponibles pour la mise au point d'un programme à base de STM32 en s'appuyant sur un exemple fourni par cet IDE pour la carte Nucleo-L152RE.

La suite du livre décrit une fonctionnalité du STM32 par chapitre et la met à chaque fois en œuvre au travers d'un exemple pratique. La première fonctionnalité, détaillée dans le chapitre 5, est la plus simple. Il s'agit de la fonction d'entrée-sortie sur les GPIO du STM32. Le chapitre 5 commence par une introduction sur les caractéristiques d'un GPIO sur le STM32, puis utilise STM32CubeMX pour configurer la carte Nucleo-L152RE afin d'utiliser cette fonctionnalité. Ensuite, un exemple simple est réalisé, qui consiste à allumer la LED LD2 présente sur la carte grâce à un GPIO du STM32L152RET6.

Une fonctionnalité intéressante dans le développement de montages électroniques est la possibilité de communiquer avec la carte. Le chapitre 6 explique comment envoyer des données à l'aide d'un clavier d'ordinateur et en recevoir sur l'écran de ce même ordinateur. Après une brève introduction sur la liaison série qui est le protocole utilisé pour cette communication, une application est développée à travers STM32CubeMX pour permettre à un ordinateur d'échanger des données avec le STM32L152RET6 de la carte Nucleo-L152RE via une liaison USB. Cette communication sera d'ailleurs souvent utilisée dans les exemples des chapitres suivants afin de les rendre plus interactifs.

Le protocole I2C est très ancien, mais on le retrouve encore souvent pour la communication entre plusieurs composants électroniques. Le chapitre 7 commence par décrire ce protocole. Puis, toujours à l'aide de STM32CubeMX, cette fonctionnalité est initialisée dans le STM32L152RET6 avant de développer une petite application qui affiche des chiffres sur des afficheurs à 7 segments se trouvant sur une carte d'extension connectée à la carte Nucleo-L152RE grâce aux connecteurs compatibles Arduino. La communication entre ces deux cartes se fait par liaison I2C.

Le protocole étudié dans le chapitre 8 est le SPI. Contrairement à l'I2C, le SPI permet d'envoyer et de recevoir des données en même temps. En contrepartie, il nécessite plus de lignes de communication. Malgré cette contrainte, il est largement utilisé de nos jours dans les applications électroniques. Après une description du protocole, ce chapitre développe l'exemple d'un chenillard programmable sur quatre LED multicolores, placées sur une carte d'extension reliée à la carte Nucleo-L152RE. Comme pour la majorité des exemples de ce livre, le logiciel graphique STM32CubeMX crée le code pour initialiser la fonction SPI du STM32L152RET6, puis le code utilisateur est développé dans l'environnement Keil µVision5.

En plus des protocoles série, I2C et SPI, la famille STM32 possède des fonctions de conversion analogique vers numérique (ADC) et inversement (DAC). Le STM32L152RET6 qui se trouve sur la carte Nucleo-L152RE ne fait pas exception à cette règle. Le chapitre 9 donne donc les bases de la conversion analogique-numérique. Ensuite, une application est développée avec STM32CubeMX et Keil µVision5 pour convertir une tension analogique en entrée sur une des broches du microcontrôleur en une valeur numérique et l'afficher sur l'écran de l'ordinateur.

Le STM32L152RET6 possède nativement un capteur qui indique à tout moment sa température interne, donnée essentielle dans les applications qui fonctionnent en continu ou dans des environnements dont les températures sont élevées. En effet, comme c'est le cas pour beaucoup de composants électroniques, une augmentation trop importante de la température interne peut l'endommager de façon irrémédiable, c'est pourquoi il est très important de prévoir tout dysfonctionnement thermique du composant. Le chapitre 10 décrit la mise en œuvre du capteur de température interne. L'IDE gratuit SW4STM32 sera utilisé dans l'exemple de ce chapitre.

La famille STM32 contient de nombreux *timers* (compteurs), qui sont regroupés par fonction : le comptage bien sûr, mais aussi le décomptage, la mesure et la génération de signaux, l'alarme. Le chapitre 11 s'intéresse à la génération d'un signal PWM, c'est-à-dire dont le rapport cyclique varie à chacune de ses périodes qui, elles, sont de durée fixe. Après une explication du fonctionnement interne de ce compteur, un exemple est mis en œuvre qui éclaire la LED située sur la carte Nucleo-L152RE en fonction du signal modulé qui vient d'une broche en sortie du STM32L152RET6.

Le chapitre 12 reprend la fonction *timer* décrite au chapitre précédent, mais pour expliquer le mode interruption. En effet, il est important de connaître son fonctionnement car il peut être utilisé par toutes les fonctions de la famille STM32. De plus, ce mécanisme permet au microcontrôleur d'exécuter d'autres tâches en attendant l'interruption. L'application exemple est simple. Elle consiste à allumer la LED située sur la carte Nucleo-L152RE à chaque interruption générée pas le comptage du *timer*.

En plus des *timers*, la famille STM32 possède un RTC *Real Time Clock* (horloge temps réel) qui fournit une horloge calendaire très précise permettant entre autres de dater les événements qui se produisent dans une application. Le chapitre 13 présente son architecture, puis propose de réaliser une alarme. Lors de la génération du code C d'initialisation avec STM32CubeMX, la configuration graphique de l'arbre d'horloge du STM32L152RET6 grâce à ce logiciel est expliquée. En général, il n'est pas nécessaire de modifier cet arbre d'horloge, mais il est toujours intéressant de savoir l'utiliser si l'application nécessite des fréquences particulières.

Le chapitre 14 s'intéresse aux modes faible consommation. Cette caractéristique est très intéressante car, de nos jours, ces microcontrôleurs sont en grande majorité embarqués dans des appareils mobiles où la puissance consommée est un facteur essentiel. Ce chapitre commence par expliquer les différents modes de consommation, avant de développer un exemple qui mesure, sur la carte Nucleo-L152RE, la consommation du STM32L152RET6 en mode RUN, puis en mode SLEEP. Pour terminer ce chapitre, une simulation de ce changement de mode est effectuée avec STM32CubeMX, avec sa fonction PCC *Power Consumption Calculator* (calculateur de la puissance consommée).

Bien que réservée à des applications spécifiques, la fonctionnalité temps réel du STM32 est abordée au chapitre 15. La famille STM32 permet en effet de développer des applications temps réel en utilisant FreeRTOS. Un exemple temps réel est ensuite développé, mettant en œuvre deux tâches indépendantes qui gèrent toutes les deux la même LED avec un accès à cette dernière à des instants très précis.

Contrairement aux exemples étudiés dans les chapitres précédents, où une application est créée entièrement, le chapitre 16 part de la démonstration fournie avec la carte Nucleo-L152RE en expliquant sa structure afin que l'utilisateur puisse repartir de cet exemple plus complexe pour l'adapter à ses besoins. La démonstration utilise la carte d'extension Adafruit 1,8" TFT qui est connectée à la carte Nucleo. L'application lit des fichiers images contenus sur une carte SD située sur la carte d'extension pour les afficher sur l'écran TFT de cette même carte.

Repartant du montage du chapitre précédent, le chapitre 17 explique en détail la gestion des fichiers sur une carte SD en mode SPI à partir du STM32L152RET6. Parmi toutes les fonctions disponibles, l'application développée prend l'exemple de la création et de l'écriture d'un fichier sur la carte SD.

Enfin, le chapitre 18 présente l'environnement de développement en ligne MBED. Cet environnement est pratique car il n'y a pas de logiciel à installer sur l'ordinateur. En contrepartie, la mise au point n'est possible qu'en utilisant la fonction `printf`. La gestion d'un projet sous cet IDE est expliquée à partir d'exemples également disponibles sur le site de cet environnement.

CHAPITRE 1

LES MICROCONTRÔLEURS STM32

La famille des STM32, développée par la société STMicroelectronics, a été mise au point pour offrir de nouveaux degrés de liberté à ses utilisateurs. Ces microcontrôleurs, qui se déclinent en une gamme complète de produits 32 bits, non seulement combinent la haute performance, le temps réel, la basse tension et la faible consommation, mais offrent également une intégration complète et un développement facile.

Les microcontrôleurs STM32 sont construits autour du processeur ARM Cortex-M développé par la société ARM.

ARM Cortex-M

PRÉSENTATION

Les processeurs ARM Cortex se déclinent en une large gamme d'architectures et de cœurs très populaires dans le monde de l'embarqué. Ils sont regroupés en trois familles principales :

- Le Cortex-**A**, pour ***A****pplication* (application), est une gamme de processeurs mise au point pour effectuer des tâches nécessitant des calculs complexes. Cette gamme supporte les jeux d'instructions ARM, Thumb et Thumb-2.
- Le Cortex-**M**, pour *e****M****bedded* (embarqué), est une gamme destinée aux microcontrôleurs qui nécessitent un bon rendement énergétique et qui visent un marché à bas prix de vente. Cette gamme supporte seulement le jeu d'instructions Thumb-2.
- Le Cortex-**R**, pour ***R****eal-Time* (temps réel), est une gamme de processeurs offrant de hautes performances pour les systèmes embarqués temps réel dans lesquels la fiabilité, la maintenance et le déterminisme sont essentiels. Cette gamme supporte les jeux d'instructions ARM, Thumb et Thumb-2.

La gamme ARM Cortex-M appartient à une génération de processeurs fournissant une architecture matérielle commune pour une large palette de demandes technologiques. Contrairement aux gammes Cortex-A et Cortex-R, la Cortex-M est un cœur complet devenu un standard dans le domaine des processeurs. Il s'appuie sur une architecture système bien définie qui comprend une gestion des interruptions, une horloge système, un système de mise au point et un espace d'adressage mémoire.

L'espace mémoire est divisé en régions distinctes prédéfinies pour le code, les données, les périphériques et le système. S'appuyant sur une architecture Harvard, le Cortex-M possède ainsi plusieurs bus, ce qui lui permet d'effectuer des opérations en parallèle. Le mode d'accès désaligné sur les données lui assure une meilleure utilisation de la mémoire et un accès plus efficace aux registres des périphériques. Le Cortex-M possède en outre des instructions RISC (*Reduce Instruction Set*, jeu d'instructions réduit) ainsi que des cycles d'instructions bien spécifiques.

Les processeurs développés par la société ARM ne sont pas des composants autonomes mais seulement des IP (*Intellectual Properties*, propriétés intellectuelles). Ce sont les fabricants, tels que STMicroelectronics, qui réalisent des composants autonomes tels que le STM32 en ajoutant au Cortex-M de la mémoire et des périphériques après en avoir acheté la licence auprès de ARM.

Un des composants clés du Cortex-M est son NVIC (*Nested Vector Interrupt Controller*, contrôleur de vecteurs d'interruptions rapprochés), qui présente une structure standard de gestion des interruptions et fournit jusqu'à 240 sources périphériques qui peuvent être chacune dédiée à une interruption. Le temps entre la réception de l'interruption et le début de l'exécution du code correspondant est optimisé en partie grâce à une gestion automatique de la pile dans le microcode du Cortex-M. À première vue, les périphériques semblent des classiques des microcontrôleurs : un double convertisseur analogique-numérique, des compteurs d'usage général, I2C, SPI, CAN, USB et un compteur temps réel. Cependant, chacun de ces périphériques présente des fonctionnalités supplémentaires. Par exemple, le convertisseur analogique-numérique possède un capteur de température et de multiples modes de conversion. De même, chacun des compteurs possède quatre unités de capture/comparaison et chaque bloc de compteur peut se combiner avec les autres pour construire des tableaux de compteurs plus sophistiqués. Un autre compteur peut contrôler un moteur, avec des sorties PWM (*Pulse Width Modulation*, modulation de largeurs d'impulsion). Le STM32 inclut aussi un contrôleur DMA (*Direct Memory Access*, accès direct mémoire) avec douze canaux. Chaque canal peut être utilisé pour transférer des données vers ou à partir d'un registre périphérique en mémoire sur 8, 16 ou 32 bits.

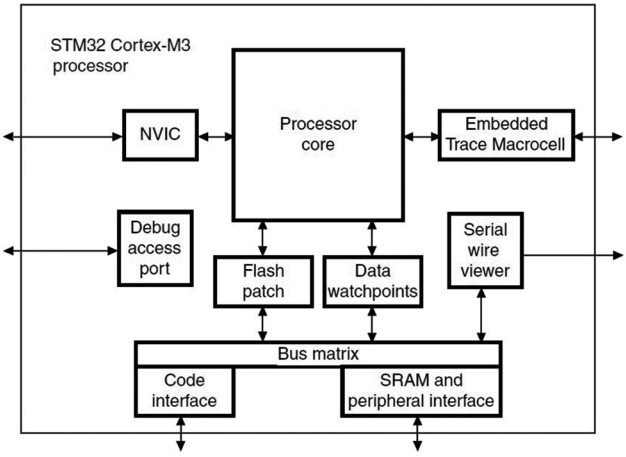


Figure 1-1. Un STM32 basé sur un Cortex-M3

PIPELINE

Le Cortex-M peut exécuter la plupart des instructions en un seul cycle d'horloge grâce à un *pipeline* (tuyau) à trois étages. Ce processeur possède également un mécanisme de branchements prédictifs afin d'éviter de vider inutilement le *pipeline*.

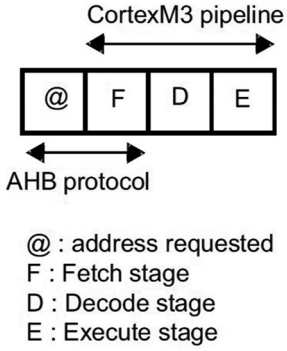


Figure 1-2. Fonctionnement du pipeline du Cortex-M

Comme le montre la figure 1-2, pendant qu'une instruction s'exécute dans le premier étage du *pipeline*, la suivante est décodée dans le second étage et une troisième est copiée de la mémoire vers le troisième étage. Ce travail convient très bien pour un code linéaire mais, quand un branchement survient, le *pipeline* doit se vider et se remplir avec les nouvelles instructions avant que le code puisse continuer à s'exécuter. Grâce à son mécanisme de branchements prédictifs, lorsqu'une instruction de branchement conditionnel est atteinte, le Cortex-M anticipe l'instruction à dupliquer de la mémoire vers le *pipeline*, de manière à ce que les deux destinations de l'instruction de condition soient disponibles sans affecter les performances du programme. Ce mécanisme de branchements prédictifs n'est cependant pas infaillible, par exemple dans certains cas de branchements indirects dans lesquels le Cortex-M ne peut anticiper l'instruction à dupliquer et doit vider son *pipeline* pour pouvoir exécuter la suite du code.

Cette gestion du *pipeline*, intégralement prise en charge par le Cortex-M, est totalement transparente pour le programmeur.

ARCHITECTURE RISC

Le Cortex-M possède un jeu d'instructions RISC avec chargement et enregistrement. En effet, lors du traitement des instructions sur les données, les opérandes sont chargés dans les registres centraux, l'opération est ensuite effectuée sur ces registres et les résultats sont finalement enregistrés en retour dans la mémoire.

La figure 1-3 illustre ce mécanisme avec l'addition de deux opérandes : ceux-ci sont d'abord chargés dans les registres R1 et R2, le résultat de l'addition est ensuite placé dans le registre R4, puis ce dernier est finalement sauvegardé en mémoire.

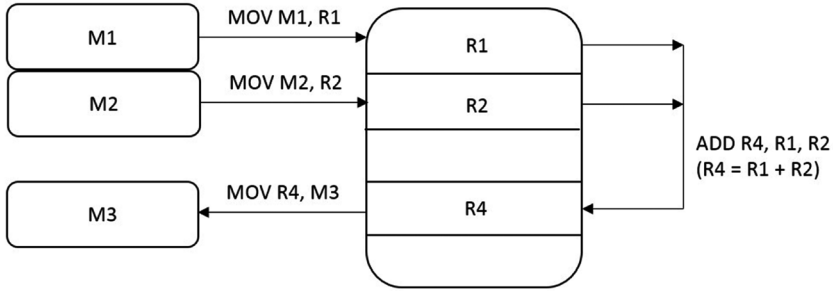


Figure 1-3. Utilisation des registres pour additionner deux opérandes

Le Cortex-M peut se retrouver dans les deux modes suivants :

- *Thread Mode* : le Cortex-M passe dans ce mode lorsqu'il exécute le programme de l'application et après une initialisation.
- *Handler Mode* : le Cortex-M passe dans ce mode pour traiter une exception. Il retourne ensuite dans le *Thread Mode* après le traitement de l'exception.

> **Remarque**
>
> Les interruptions et les exceptions sont deux causes de rupture de l'exécution normale d'un programme. Les interruptions surviennent de façon asynchrone et sont commandées par le matériel ou le système. Les exceptions sont déclenchées par des incidents lors de l'exécution du programme, par exemple un débordement arithmétique, une erreur bus ou encore l'utilisation d'une instruction réservée.

Le Cortex-M possède aussi des niveaux de privilèges distincts lors de l'exécution d'un programme :

- Le niveau *Unprivileged* (non privilégié) donne les droits suivants au programme :
  - il a un accès limité aux instructions `MSR` et `MRS` et ne peut pas utiliser l'instruction `CPS` ;
  - il ne peut pas accéder au compteur système, au NVIC ni au bloc de contrôle du système ;
  - il peut avoir un accès restreint à la mémoire et aux périphériques.
- Le niveau *Privileged* (privilégié) laisse le programme accéder à toutes les instructions et à toutes les ressources.

Les registres R0 à R12 sont à usage général, pour les opérations sur des données.

Les registres R13 à R15 ont des fonctions spéciales à l'intérieur du Cortex-M :

- Le registre R13 est le SP (*Stack Pointer*, pointeur de pile). Lorsque le Cortex-M se trouve en *Thread Mode*, c'est le bit 1 du registre de contrôle qui indique le pointeur de pile à utiliser :
  - 0 pour MSP (*Main Stack Pointer*, pointeur de pile principal) : c'est le pointeur après une initialisation du Cortex-M ;
  - 1 pour PSP (*Process Stack Pointer*, pointeur de pile de traitement).
- Le registre R14 est le LR (*Link Register*, registre de lien). Il mémorise l'information de retour pour les sous-routines, les appels de fonction et les exceptions.
- Le dernier registre, R15, est le PC (*Program Counter*, compteur programme). Il contient l'adresse courante du registre.

Le PSR (*Program Status Register*, registre d'état du programme), représenté à la figure 1-5, combine trois registres :

- APSR (*Application Program Status Register*, registre d'état du programme d'application) ;
- IPSR (*Interrupt Program Status Register*, registre d'état du programme d'interruption) ;
- EPSR (*Execution Program Status Register*, registre d'état du programme d'exécution).

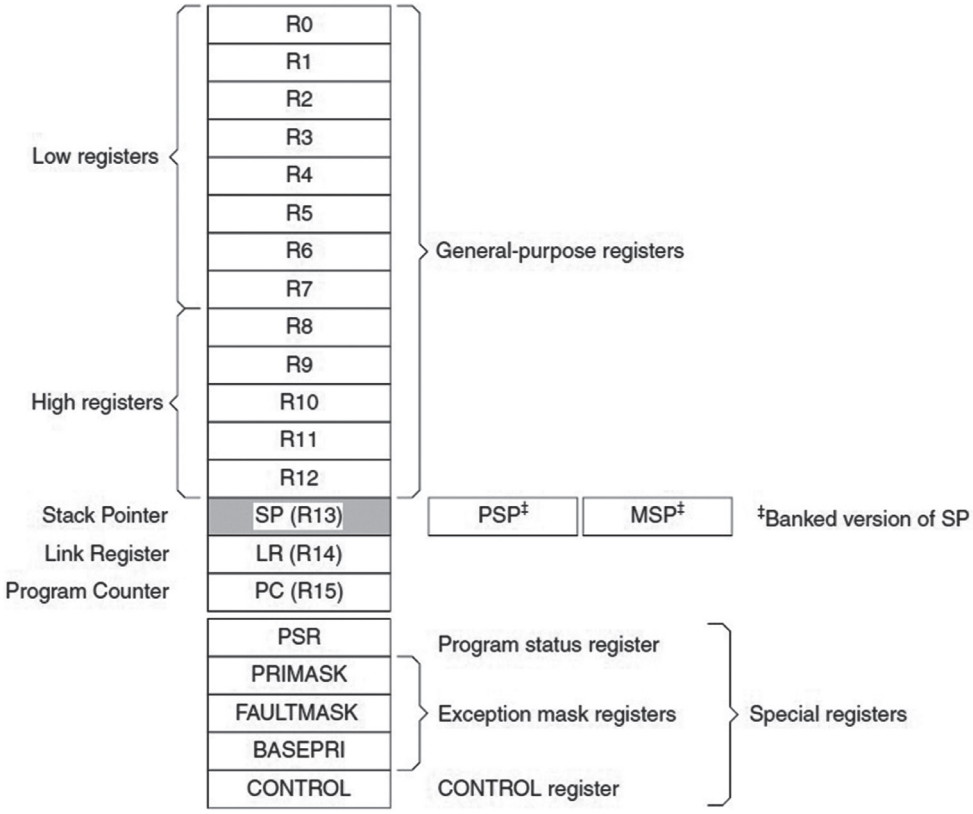


Figure 1-4. Registres centraux du processeur ARM Cortex-M3

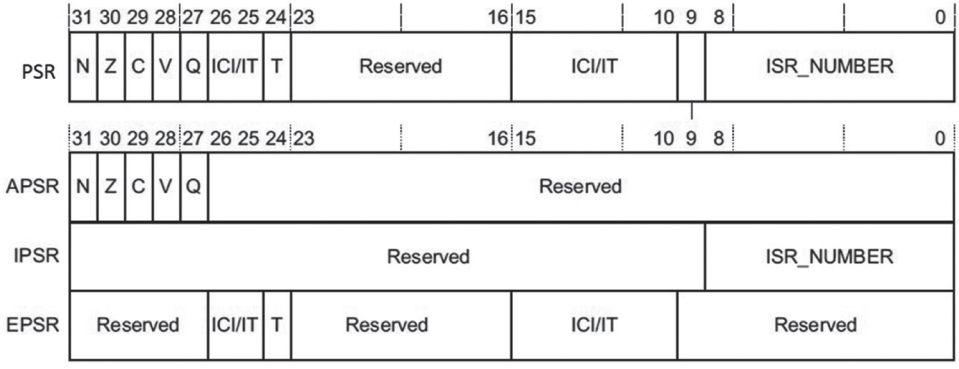


Figure 1-5. Représentation du PSR, le registre d'état du programme

L'accès à ces registres, individuellement ou en les combinant, s'effectue en utilisant le nom du registre désiré comme argument des fonctions `MSR` ou `MRS`.

Les quatre premiers bits sont des drapeaux de code condition, appelés N (*Negative*, négatif), Z (*Zero*, zéro), C (*Carry*, retenue) et V (*oVerflow*, dépassement) et positionnés suivant le résultat de l'instruction de traitement de l'opération. Le bit Q est utilisé par les instructions mathématiques pour indiquer qu'une variable a atteint sa valeur maximale ou minimale. Les champs 26-10 sont utilisés dans les branchements conditionnels et les champs 7-0 pour les interruptions.

Le PRIMASK (*PRIority MASK register*, registre de masque de priorité) contrôle l'activation des exceptions grâce à des niveaux de priorité configurables.

Le FAULTMASK (*FAULT MASK register*, registre de masque de faute) empêche l'activation de toutes les exceptions sauf NMI (*Non-Maskable Interrupt*, interruption non masquable).

Le BASEPRI (*BASE PRIority mask register*, registre de masque de base) définit la priorité minimale pour le traitement d'une exception.

Le CONTROL *register* (registre de contrôle) détermine la pile utilisée et le niveau de privilège pendant l'exécution du programme en *Thread Mode*.

ORGANISATION MÉMOIRE

Le Cortex-M voit sa mémoire comme une suite linéaire d'octets numérotés par ordre croissant en partant de zéro. Ces octets sont codés en mémoire dans le format *little endian*. Dans ce format, dont un exemple est représenté à la figure 1-6, le processeur enregistre l'octet de poids le plus faible d'un mot en premier et l'octet de poids le plus fort en dernier.

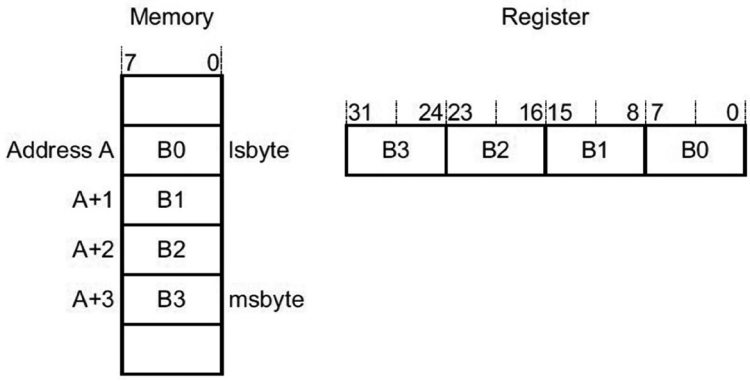


Figure 1-6. Exemple de stockage au format little endian

MANIPULATION DE BITS

Concernant les manipulations sur les bits en mémoires SRAM et périphériques, les processeurs sont en majorité seulement capables d'utiliser des opérations `AND` et `OR` ainsi que des masques sur les bits. Ceci nécessite une opération de lecture-modification-écriture qui est gourmande en nombre de cycles processeur.

Le Cortex-M, quant à lui, introduit nativement la technique de *bit banding* (alignement des bits). Comme le montre la figure 1-7, cela consiste à prendre une région mémoire du processeur adressable par bit de 1 Mo et une région d'alias pour l'alignement des bits qui utilise 32 Mo et adressable par mot de 32 bits. Cet alignement affecte chaque bit de la région de 1 Mo à une adresse mémoire de la région d'alias de 32 Mo. Ainsi, la manipulation bit par bit s'effectue directement en référençant une adresse mémoire de la région d'alias correspondant au bit désiré.

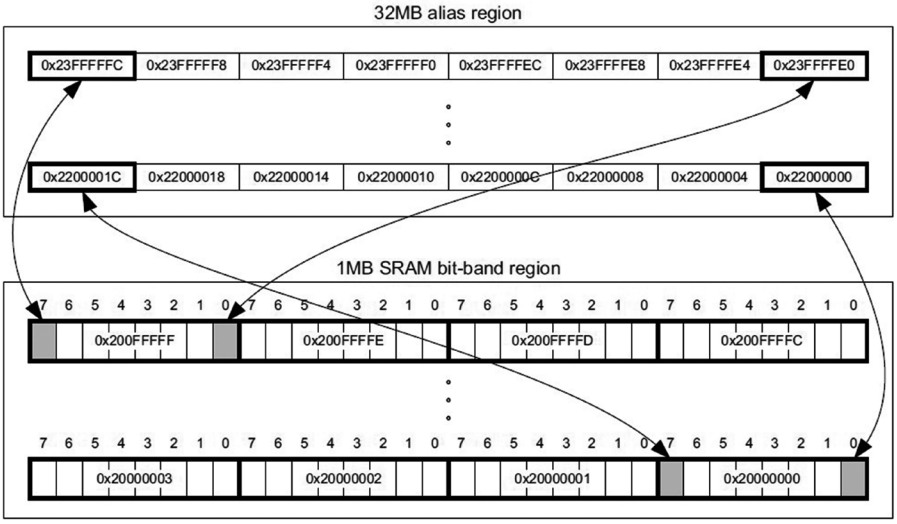


Figure 1-7. Technique de l'alignement des bits

LA GAMME CORTEX-M

La gamme des Cortex-M se décline en différents modèles :

- Le Cortex-M0/M0+/M1 traite des données d'ordre général ainsi que les tâches d'entrée-sortie standard.
- Le Cortex-M3 s'occupe de traitements avancés sur les données et de la manipulation des champs de bits.
- Le Cortex-M4 comprend un DSP (*Digital Signal Processing*, traitement numérique des signaux) et un FPU (*Flotting Point Unit*, unité à virgule flottante).
- Le récent Cortex-M7 délivre un très haut niveau de performance.

Le jeu d'instructions correspondant à chaque modèle de la gamme Cortex-M est ascendant compatible (figure 1-8).

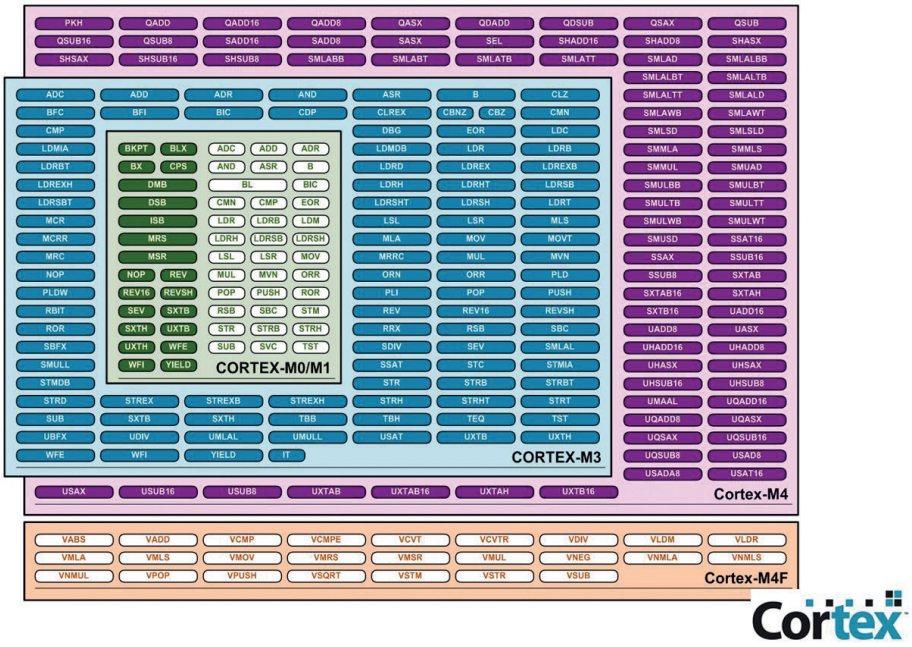


Figure 1-8. Jeu d'instructions de chaque modèle de Cortex-M

À ce processeur sont rattachés de nombreux périphériques ainsi que de la mémoire, pour former le microcontrôleur STM32 capable de gérer à lui tout seul une application complète, ce que ne pourrait pas faire un processeur seul. Il existe toute une gamme de STM32 suivant le processeur Cortex-M qui lui est associé, ses périphériques et sa vitesse de fonctionnement.

La famille STM32

Actuellement, la famille STM32 se divise en deux grandes catégories :

- Les STM32Fxx, **F** pour ***F**ast* (rapide), sont utilisés dans les applications qui donnent priorité à la vitesse d'exécution et dont la consommation en énergie n'est pas une exigence première.
- Les STM32Lxx, **L** pour ***L**ow power* (faible consommation), sont utilisés dans des domaines où la faible consommation en énergie est prioritaire par rapport à la vitesse d'exécution de l'application.

En effet, pour aller plus vite on doit consommer plus et, inversement, moins on veut consommer moins il faut aller vite. Il est donc très difficile, voire impossible, d'avoir ces deux qualités, vitesse d'exécution et faible consommation, dans un même microcontrôleur.