NSY107
Cours 1 Introduction aux microcontrôleurs

Matthias Puech

Master 1 SEMS — Cnam

6 mars 2018

Cours 1 Introduction aux microcontrôleurs

NSY107 : Thématiques abordées

Un cours pratique sur la communication embarquée

NSY107 : Thématiques abordées

Un cours pratique sur la communication embarquée

- programmation bare-metal sur microcontrôleurs
- architecture machine
- schémas de compilation
- notion de pilote de périphérique
- protocoles client-serveur bas niveau (UART, I2C, CAN...)

Infos pratiques

- Matthias Puech `matthias.puech@lecnam.net`
- Page web
  `http://cedric.cnam.fr/~puechm/ens/usrs26/`
- 15 séances, et pour chacune :
  - 1h–2h de cours
  - 1h–2h de TP

Infos pratiques

- Matthias Puech `matthias.puech@lecnam.net`
- Page web
  `http://cedric.cnam.fr/~puechm/ens/usrs26/`
- 15 séances, et pour chacune :
  - 1h–2h de cours
  - 1h–2h de TP

Evaluation

- $N = max(\text{TP noté}, \text{examen session 1})$ si $N > 10$
- $N =$ examen session 2 sinon.

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?
- qu'est-ce qu'une interruption ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?
- qu'est-ce qu'une interruption ?
- que fait l'option `-o` de GCC ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?
- qu'est-ce qu'une interruption ?
- que fait l'option `-o` de GCC ?
- qu'est-ce que le scheduler dans un OS ? un driver ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?
- qu'est-ce qu'une interruption ?
- que fait l'option `-o` de GCC ?
- qu'est-ce que le scheduler dans un OS ? un driver ?
- quel est le type d'un tableau de flottants/fonctions en C ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?
- qu'est-ce qu'une interruption ?
- que fait l'option `-o` de GCC ?
- qu'est-ce que le scheduler dans un OS ? un driver ?
- quel est le type d'un tableau de flottants/fonctions en C ?
- combien vaut `(1 << 4) | (1 << 7)` ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?
- qu'est-ce qu'une interruption ?
- que fait l'option `-o` de GCC ?
- qu'est-ce que le scheduler dans un OS ? un driver ?
- quel est le type d'un tableau de flottants/fonctions en C ?
- combien vaut (1 « 4) | (1 « 7) ?
- comment écrit-on une cible et ses dépendances dans un `Makefile` ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?
- qu'est-ce qu'une interruption ?
- que fait l'option `-o` de GCC ?
- qu'est-ce que le scheduler dans un OS ? un driver ?
- quel est le type d'un tableau de flottants/fonctions en C ?
- combien vaut (1 « 4) | (1 « 7) ?
- comment écrit-on une cible et ses dépendances dans un `Makefile` ?
- qu'est-ce que UART ? I2C ? CAN ?

Pour commencer : quiz !

- citer 2 compilateurs natifs ? 2 machines virtuelles ?
- qu'est-ce un ARM ? un RISC ? un registre ?
- qu'est-ce qu'une interruption ?
- que fait l'option `-o` de GCC ?
- qu'est-ce que le scheduler dans un OS ? un driver ?
- quel est le type d'un tableau de flottants/fonctions en C ?
- combien vaut (1 « 4) | (1 « 7) ?
- comment écrit-on une cible et ses dépendances dans un `Makefile` ?
- qu'est-ce que UART ? I2C ? CAN ?
- qu'est-ce que ASK, FSK, PSK, QPSK ?

Cours 1 Introduction aux microcontrôleurs

Introduction

Présentation des microcontrôleurs

Outils de développement

Documentation et code disponible

Cours 1 Introduction aux microcontrôleurs

Qu’est-ce qu’un microcontrôleur ?

Attention

Aujourd’hui, séance principalement consacrée au *name-dropping*.

Qu'est-ce qu'un microcontrôleur ?

Attention

Aujourd'hui, séance principalement consacrée au *name-dropping*.

Microcontrôleur (μC, uC, MCU)

Circuit intégré rassemblant les éléments essentiels d'un ordinateur

- unité de calcul et de contrôle (le coeur)
- mémoire vive
- périphériques
  (ex : port série, timers, DMA, flash...)
- entrées/sorties
  (ex : convertisseurs analogique $\longleftrightarrow$ numérique (ADC/DAC))

Exemples d'utilisation

Electroménager & Internet of Things

- montres connectées, bracelets santé
- capteurs domotiques, four, machine à laver, thermostat…
- smartphones, tablettes

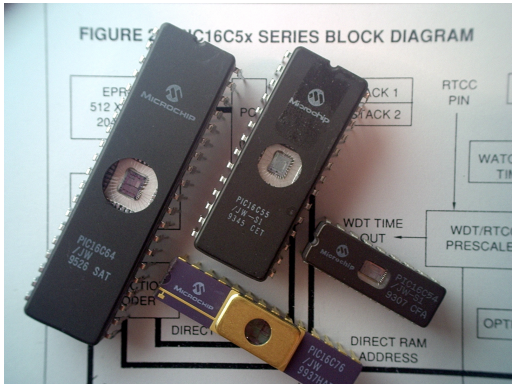
Automobile, Aeronautique

- une voiture moderne intègre ≈30 MCUs
- logiciel critique dans l'avionique (cf. reste du master)

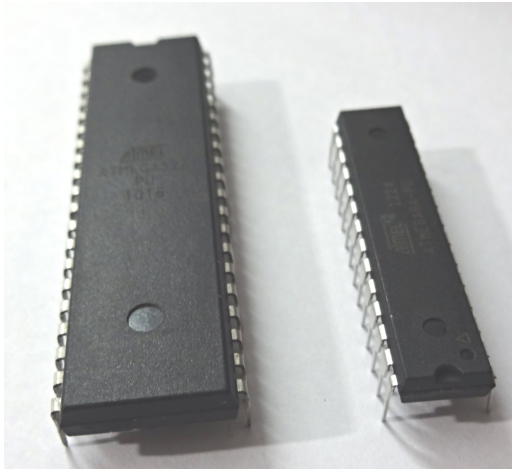
Un foyer moyen possède 4 CPUs et 30 MCUs.

Quelques MCUs fameux dans leur habitat naturel

- Microchip PIC16 (1990, 8 bits, 1MHz, quelques registres)

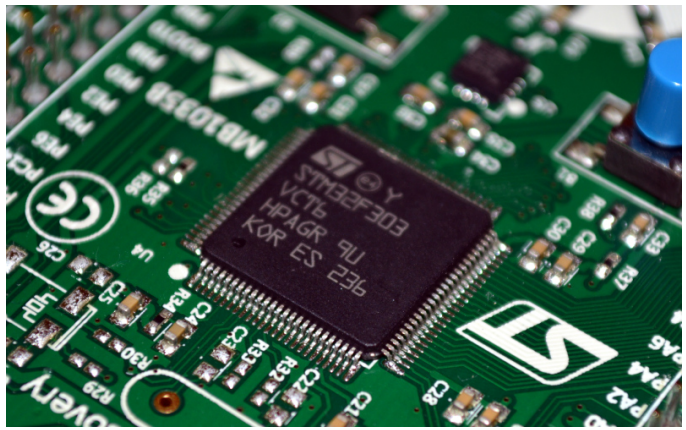


- Microchip PIC12 (1990, très utilisé en DIY)
- Atmel AVR (2000, 8/16 bits, base des Arduino)

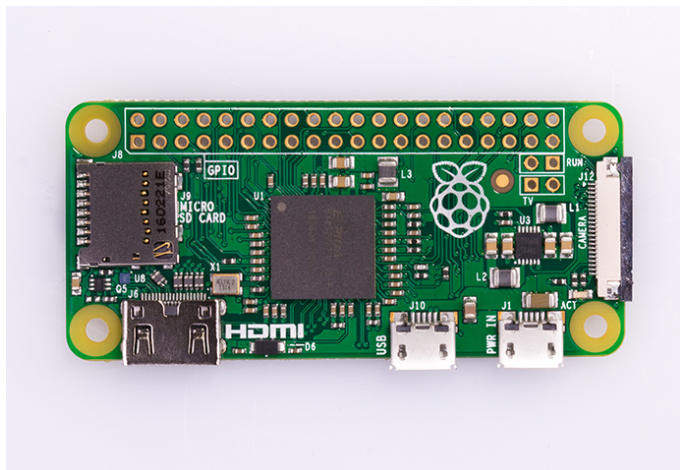


Quelques MCUs fameux dans leur habitat naturel

- ARM (2000) : Coeurs communs, différents constructeurs (ST, Texas Instruments, Microchip…)



- SoC Freescale, Broadcom, TI etc. (2010, Raspberry Pi)



1GHz, 64 bits, RAM 512 Mo

Le MCU, un compromis

L'intérêt

- forte intégration (une puce intègre toutes les fonctions)
- faible consommation électrique (1–500mW)
- faible coût (0.10–10€)
- généricité (par rapport au silicium dédié)

Le MCU, un compromis

L'intérêt

- forte intégration (une puce intègre toutes les fonctions)
- faible consommation électrique (1–500mW)
- faible coût (0.10–10€)
- généricité (par rapport au silicium dédié)

Les limitations

- peu de capacité de calcul (1–200 MHz)
- *très* faible stockage (1–512 Ko RAM, 1K–1M flash)

Un MCU n'est *pas* :

- un CPU (*Central Processing Unit*)
  (ex : Intel i7, Apple A11, ...)
  - puissance de calcul supérieure (GHz)
  - pas de mémoire embarquée (quelques registres)
  - pas de périphériques embarqués (bus externes)
- un FPGA (*Field-Programmable Gate Array*)
  (ex : Xilinx, Altera...)
  - circuit intégré reconfigurable
  - grand nombre de portes logiques généralistes
  - "programmation" en Verilog/VHDL

Protocoles de communication embarqués

Toute communication entre le coeur et ses périphériques s'établit selon un *protocole* (une langue commune)

Interne

bus de données : unité de calcul $\longleftrightarrow$ RAM $\longleftrightarrow$ DMA,

Externe

communication avec puces externes ; communication inter-MCU
(ex : réseau de capteurs)

Architectures

symétrique, en réseaux, client-serveur...
en général : couches de nombreux protocoles

Exemple

UART, SPI, I2C, CAN, USB...

La famille des ARM Cortex

ARM architecture qui spécifie (notamment) un jeu d'instructions RISC, l'organisation de la mémoire, modèle d'exécution...
Différentes versions du standard : ARMv*X*-M

La famille des ARM Cortex

ARM architecture qui spécifie (notamment) un jeu d'instructions RISC, l'organisation de la mémoire, modèle d'exécution...
Différentes versions du standard : ARMv*X*-M

Cortex Famille de coeurs physiques
vendus aux fabricants de silicone

- **Cortex-A** pour Application (ex : smartphones)
- **Cortex-R** pour Real-time
  (temps d'exécution déterministe)
- **Cortex-M** pour eMbedded (faible consommation)

La famille des ARM Cortex

ARM architecture qui spécifie (notamment) un jeu d'instructions RISC, l'organisation de la mémoire, modèle d'exécution...
Différentes versions du standard : ARMv*X*-M

Cortex Famille de coeurs physiques
vendus aux fabricants de silicone

- **Cortex-A** pour Application (ex : smartphones)
- **Cortex-R** pour Real-time
  (temps d'exécution déterministe)
- **Cortex-M** pour eMbedded (faible consommation)
  - **M0** ARMv6-M, petit et pas cher
  - **M3** ARMv7-M, *,/ hardware, plus d'instructions
  - **M4** ARMv7E-M, FPU, DSP

La famille des ARM Cortex

ARM architecture qui spécifie (notamment) un jeu d'instructions RISC, l'organisation de la mémoire, modèle d'exécution...
Différentes versions du standard : ARMv*X*-M

Cortex Famille de coeurs physiques
vendus aux fabricants de silicone

- **Cortex-A** pour Application (ex : smartphones)
- **Cortex-R** pour Real-time
  (temps d'exécution déterministe)
- **Cortex-M** pour eMbedded (faible consommation)
  - **M0** ARMv6-M, petit et pas cher
  - **M3** ARMv7-M, *,/ hardware, plus d'instructions
  - **M4** ARMv7E-M, FPU, DSP

Licences Cortex-M4

Atmel, STMicroelectronics, NXP, Texas Instruments

La famille des MCU STMicroelectronics STM32

STM32L0	Cortex M0+, 32MHz, 8Ko SRAM, 32-64Ko flash
STM32F1	Cortex M1, 24-72MHz, 4-96Ko SRAM, 16-1024Ko
STM32F3	Cortex M4 + FPU, 72MHz, 16-40Ko SRAM, 64-256Ko
...STM32F7	ARM Cortex-M7F, 216MHz, 512-1024Ko RAM, ...

La famille des MCU STMicroelectronics STM32

STM32L0 Cortex M0+, 32MHz, 8Ko SRAM, 32-64Ko flash

STM32F1 Cortex M1, 24-72MHz, 4-96Ko SRAM, 16-1024Ko

STM32F3 Cortex M4 + FPU, 72MHz, 16-40Ko SRAM, 64-256Ko

...STM32F7 ARM Cortex-M7F, 216MHz, 512-1024Ko RAM, ...

Chaque famille contient divers périphériques (E/S, calcul, ...) :

- 2xADC multiplexé, 2xDAC 12/16 bits
- USART, SDIO, I2C, SPI, USB, ...
- EEPROM, flash, ROM...
- DMA, timers, watchdogs, RTC, RNG...

La famille des MCU STMicroelectronics STM32

STM32L0 Cortex M0+, 32MHz, 8Ko SRAM, 32-64Ko flash

STM32F1 Cortex M1, 24-72MHz, 4-96Ko SRAM, 16-1024Ko

STM32F3 Cortex M4 + FPU, 72MHz, 16-40Ko SRAM, 64-256Ko

...STM32F7 ARM Cortex-M7F, 216MHz, 512-1024Ko RAM, ...

Chaque famille contient divers périphériques (E/S, calcul, ...) :

- 2xADC multiplexé, 2xDAC 12/16 bits
- USART, SDIO, I2C, SPI, USB, ...
- EEPROM, flash, ROM...
- DMA, timers, watchdogs, RTC, RNG...

Applications

- DSP audio (ADC/DAC)
- contrôle de moteur, prototypage généraliste
  (ex : Arduino, cartes Nucleo/Discovery)

La série des STM32F3

STM32F3 plusieurs sous-familles :

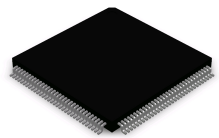
- STM32F301, STM32F302, STM32F303 généralistes, différents périphériques analogiques (contrôle de moteur)
- STM32F334 timer haute résolution (217 picosecondes)
- STM32F373 16-bit sigma-delta ADC et ampli-ops intégrés
- STM32F3x8 marche à 1.8V au lieu de 3.3V

La série des STM32F3

STM32F3 plusieurs sous-familles :

- STM32F301, STM32F302, STM32F303 généralistes, différents périphériques analogiques (contrôle de moteur)
- STM32F334 timer haute résolution (217 picosecondes)
- STM32F373 16-bit sigma-delta ADC et ampli-ops intégrés
- STM32F3x8 marche à 1.8V au lieu de 3.3V

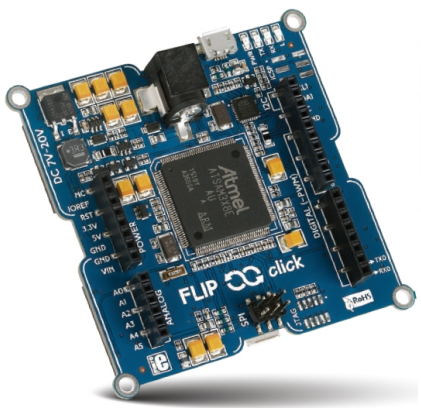
. . .chacuns dans différents choix de RAM, flash et packages :

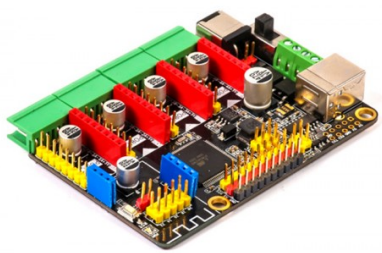
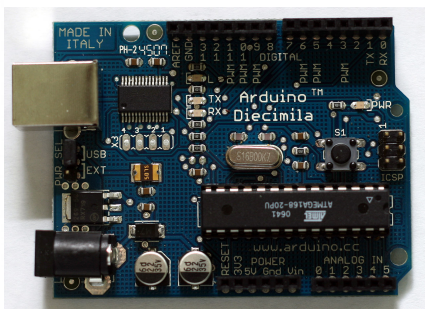
LQFP32  , LQFP100  , UFBGA100  . . .

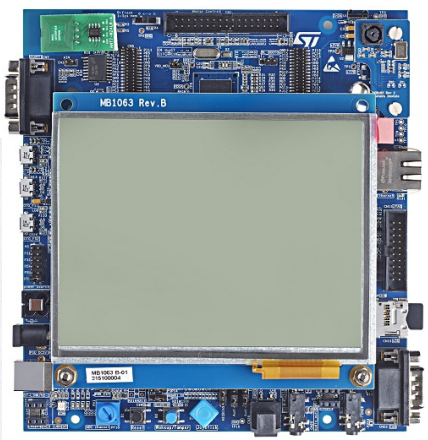
Les cartes d'essai

Si vous voulez étudier les MCUs ou faire du prototypage rapide, pas besoin de concevoir une carte à chaque fois, il existe de nombreuses *cartes d'essai*.

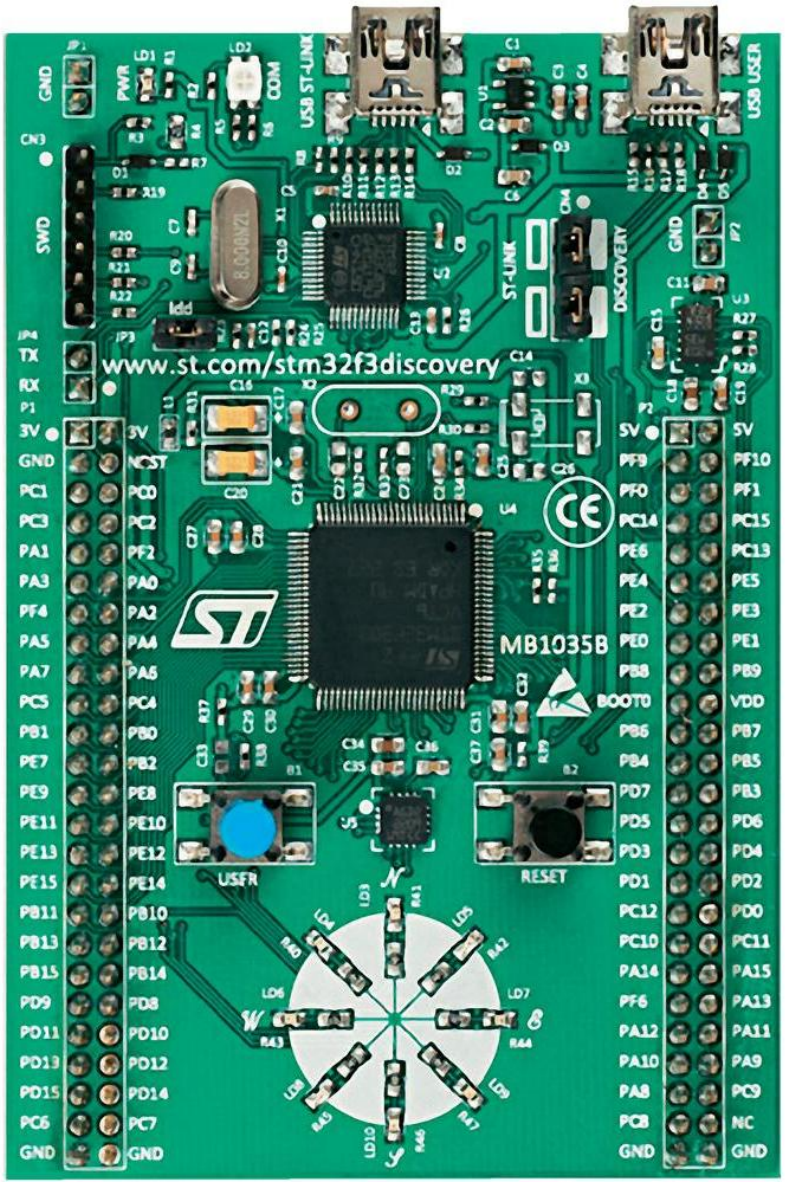
- produits d'appel pour les fabricants, donc très bon marché (≈20€)
- divers périphériques embarqués : capteurs, écrans, réseau, audio…
- programmation facile (interface USB)





La carte STM32F3-discovery



- STM32F303 : 72MHz, 48Ko RAM, 256Ko flash
- documentation et schéma
- programmeur USB intégré ST-Link
- (presque) toutes les pattes du MCU accessibles sur header
- périphériques externes :
  - port USB utilisateur
  - alimentation par USB ou externe (pile)
  - accéléromètre/boussole 3D
  - gyroscope
  - 10 LEDs
  - 1 bouton utilisateur

Cours 1 Introduction aux microcontrôleurs

Présentation

Programme C :

```
int main() {
  RCC→AHBENR |= (1 « 21);     /* enable GPIO E clock */
  GPIOE→MODER |= 0x55550000;   /* configure E8–E15 for output */
  RCC→AHBENR |= (1 « 17);     /* enable GPIO A clock */
  GPIOA→MODER |= 0x00000000;   /* configure A0 for digital input */
  while(1) {
    if (!(GPIOA→IDR & 0x00000001))
      GPIOE→ODR ^= 0x0000FF00;   /* invert pin 8–15 to 1 */
    delay();
  }
}
```

Présentation

. . .compilé vers assembleur ARMv7 :

```
main:
        push    {r7, lr}
        add     r7, sp, #0
        ldr     r2, .L9
        ldr     r3, .L9
        ldr     r3, [r3, #20]
        orr     r3, r3, #2097152
        str     r3, [r2, #20]
        ldr     r2, .L9+4
        ldr     r3, .L9+4
        ldr     r3, [r3]
        orr     r3, r3, #1426063360
        orr     r3, r3, #5570560
```

Présentation

. . .assemblé en binaire :

```
0000000: ff9f 0020 ed03 0008 3d04 0008 3d04 0008  ... ....=...=...
0000010: 3d04 0008 3d04 0008 3d04 0008 0000 0000  =...=...=.......
0000020: 0000 0000 0000 0000 0000 0000 3d04 0008  ............=...
0000030: 3d04 0008 0000 0000 3d04 0008 3d04 0008  =.......=...=...
0000040: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
0000050: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
0000060: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
0000070: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
0000080: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
0000090: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
00000a0: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
00000b0: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
00000c0: 3d04 0008 3d04 0008 3d04 0008 3d04 0008  =...=...=...=...
```

Présentation

. . .téléchargé sur la carte par SWD (Single Wire Debug) :

```
# st-flash write main.bin 0x8000000
st-flash 1.3.0
2017-02-20T15:00:47 INFO: Loading device parameters....
2017-02-20T15:00:47 INFO: Device connected is: F3 device, id 0x10036422
2017-02-20T15:00:47 INFO: SRAM size: 0xa000 bytes (40 KiB), Flash:
      0x40000 bytes (256 KiB) in pages of 2048 bytes
2017-02-20T15:00:47 INFO: Attempting to write 2540 (0x9ec) bytes to stm32
      address: 134217728 (0x8000000)
Flash page at addr: 0x08000800 erased
2017-02-20T15:00:47 INFO: Finished erasing 2 pages of 2048 (0x800) bytes
2017-02-20T15:00:47 INFO: Starting Flash write for VL/F0/F3 core id
2017-02-20T15:00:47 INFO: Successfully loaded flash loader in sram
  1/1 pages written
2017-02-20T15:00:47 INFO: Starting verification of write complete
2017-02-20T15:00:47 INFO: Flash written and verified! jolly good!
#
```

Présentation

Le système est maintenant indépendant, peut être déconnecté de l'USB, alimenté (par pile, transfo...) et interagir avec son environnement par ses broches et ses composants (LEDs etc.)

Présentation

[optionnel] déboguage :

```
# arm-none-eabi-gdb main.elf --eval-command="target remote
      localhost:4242"
GNU gdb (GNU Tools for ARM Embedded Processors) 7.10.1.20160923-cvs
Remote debugging using localhost:4242
0x080003ec in Reset_Handler ()
(gdb) br main.c:18
Breakpoint 1 at 0x800025c: file main.c, line 18.
(gdb) c
Continuing.
Breakpoint 1, main () at main.c:19
19              if (!(GPIOA->IDR & 0x00000001))
(gdb) n
20                GPIOE->ODR ^= 0x0000FF00;   /* invert pin 8-15 to 1 */
(gdb)
21              delay();
(gdb)
```

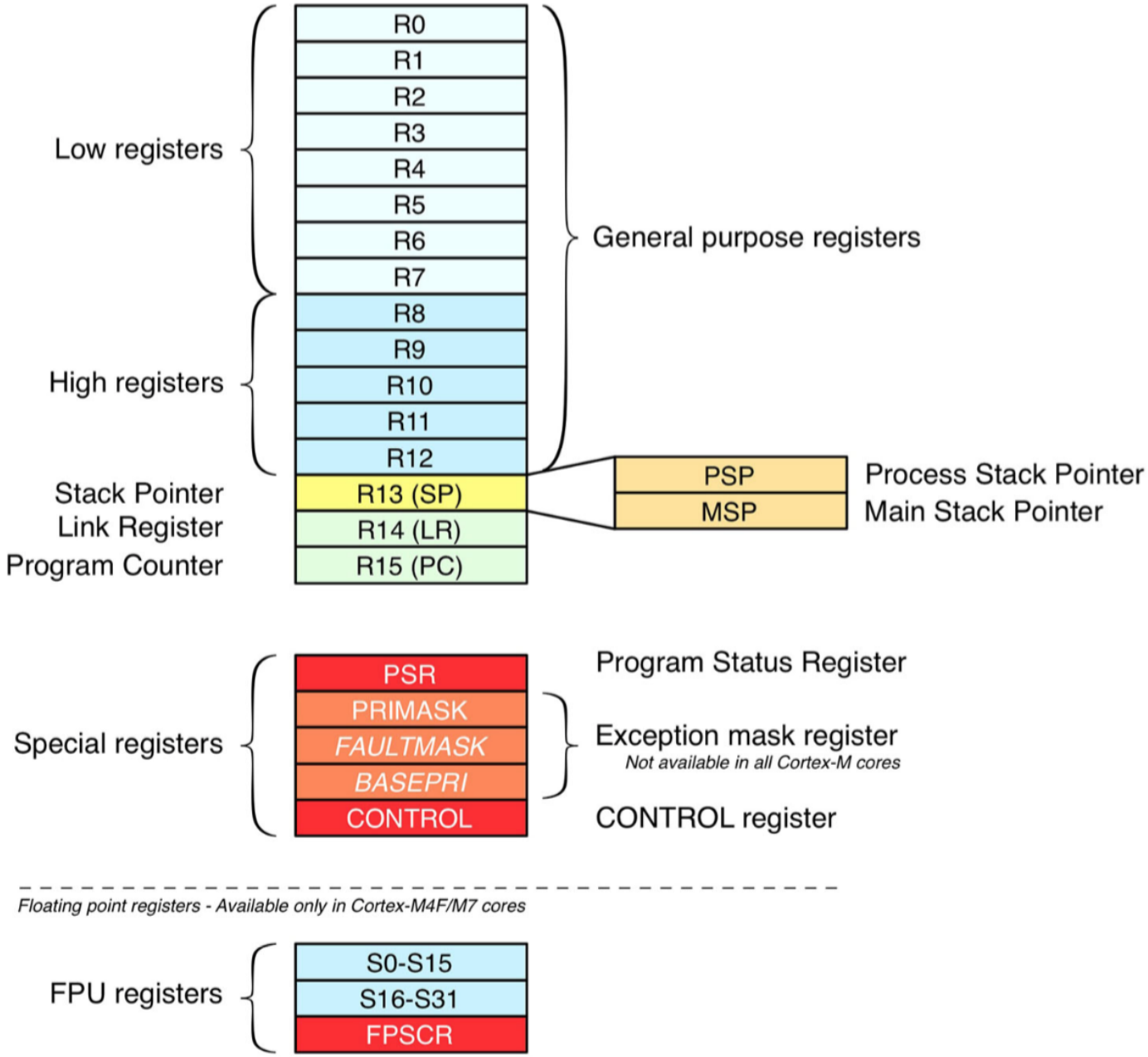
Modèle d'execution

Le coeur ARM : une machine load/store

- ≈ 30 registres de 32 bits
  (ex : *R0-R12, SP, PC*)
- cycle fetch/decode/execute (avec pipelining basique)
- des instructions qui changent l'état des registres
  (ex : *add*, *bl*)
- des instructions qui lisent/écrivent dans la mémoire
  (ex : *ldr*, *str*)
- . . .et c'est tout !

Toute l'interaction avec le monde extérieur (périphériques) se fait par écriture dans des cases de mémoire spéciales.

Registres des Cortex-M



L'espace d'adressage des Cortex-M

Il y a un niveau d'indirection entre mémoire RAM et addresses mémoires : c'est le mécanisme de *mémoire virtuelle*.

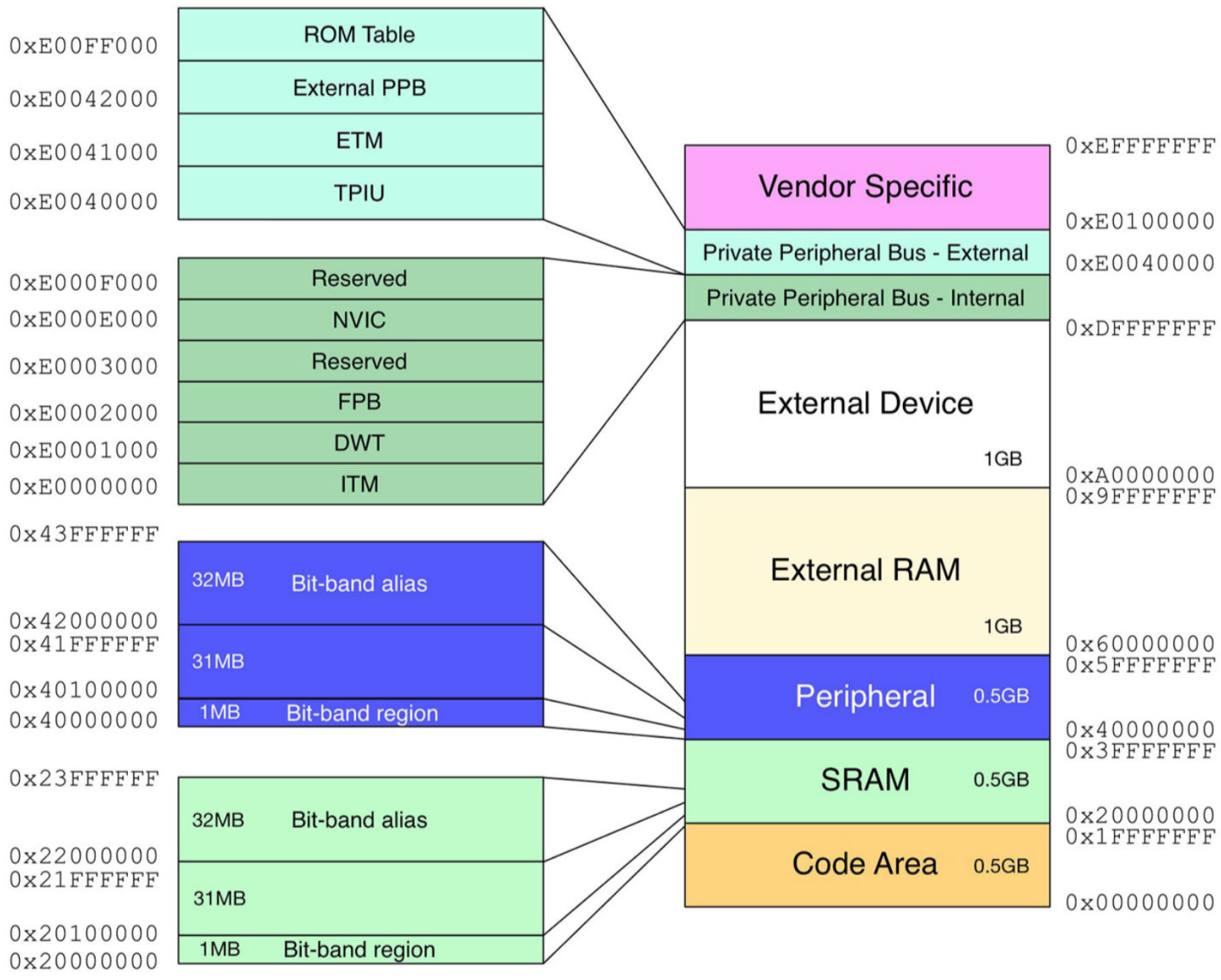
- On peut "addresser" (lire/écrire avec les instructions load/store) n'importe quelle addresse mémoire entre 0x00000000 et 0xEFFFFFFF

→ 4Go d'adresses

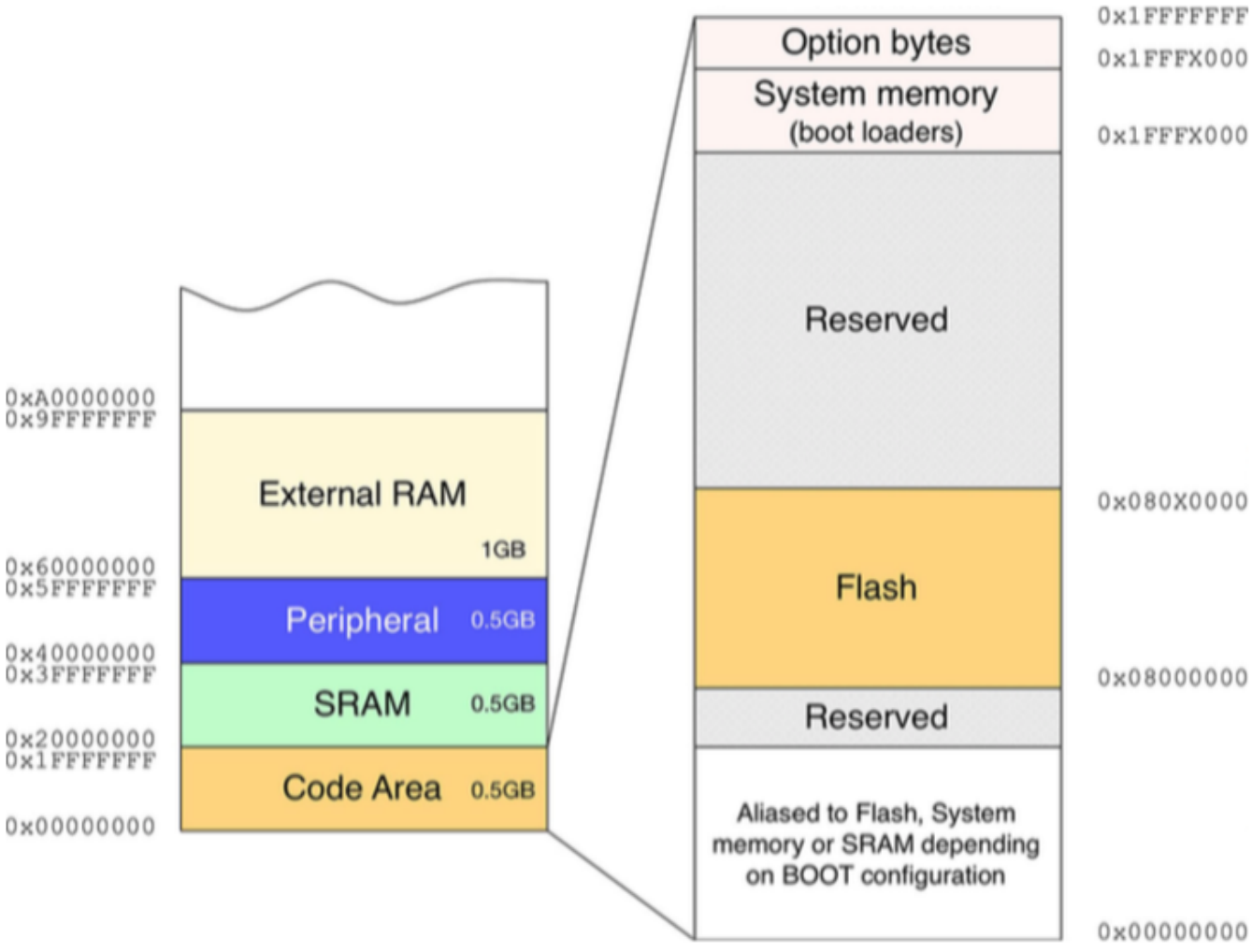
- toutes ces cases mémoire ne correspondent pas à la RAM !

→ notion d'*espace d'adressage* (spécification Cortex-M)

L’espace d’adressage des Cortex-M



L’espace d’adressage des Cortex-M



L'espace d'adressage des Cortex-M

Example

Pour allumer les LEDs de la carte, il faut :

- écrire 0x00200000 à l'adresse 0x40021014,
- écrire 0x55550000 à l'adresse 0x48000000,
- écrire 0xFFFFFFFF à l'adresse 0x48001014.

Le manuel de référence du STM32F303 décrit tous ces *registres*. [1]

1. Attention, sens de "registre" différent du dernier slide : adresse mémoire fictive correspondant à une fonction précise d'un périphérique.

L'espace d'adressage des Cortex-M

Example

Pour allumer les LEDs de la carte, il faut :

- écrire 0x00200000 à l'adresse 0x40021014,
- écrire 0x55550000 à l'adresse 0x48000000,
- écrire 0xFFFFFFFF à l'adresse 0x48001014.

Le manuel de référence du STM32F303 décrit tous ces *registres*. [1]

A retenir

- La mémoire flash commence à 0x08000000

1. Attention, sens de "registre" différent du dernier slide : adresse mémoire fictive correspondant à une fonction précise d'un périphérique.

Comment développer pour les ARM STM32 ?

Langages

C ou C++...
(ou tout langage compilant vers ARMv7)

Environnements de développement

Il en existe plusieurs (IDE, compilateur, debugger etc.) :

- Keil IDE / ArmCC (Keil),
- IAR Embedded Workbench (IAR),
- mBed (ARM)
- SW4STM/Eclipse/CubeMX/`gcc` (STMicroelectronics)
- ...
- `gcc/gdb/make/emacs` :)

Librairies

. . .et plusieurs façons d'accéder aux registres pour configurer/interagir avec les périphériques :

- CMSIS (ARM)
  (paraphrase du manuel de référence)
- HAL (STMicroelectronics) (bibliothèque d'abstraction matérielle)
- mBed (ARM)
- FreeRTOS
  (système d'exploitation embarqué temps réel)

CMSIS Cortex Microcontroller Software Interface Standard

Une surcouche très légère au dessus du matériel pour faciliter la programmation en C. Ensemble de macros qui donnent des noms :

- aux registres des périphériques
- à leurs valeurs possibles

Example

```
RCC→AHBENR |= RCC_AHBENR_GPIOEEN_Msk;
```

au lieu de :

```
0x40021014 |= 0x00200000;
```

Confère

`stm32f303xc.h` dans le code fourni

HAL Hardware Abstraction Layer

Bibliothèque de plus haut niveau (STMicroelectronics), qui cherche à s'abstraire des caractéristiques techniques de chaque MCU

- \+ code plus clair
- \+ plus portable
- − pas forcément plus concis
- − nombreux bugs !

Example

```
__HAL_RCC_GPIOE_CLK_ENABLE();
```

au lieu de :

```
RCC→AHBENR |= RCC_AHBENR_GPIOEEN_Msk;
```

Documentation de référence

- Page d'accueil du STM32F303

  `http://www.st.com/en/microcontrollers/stm32f303vc.html`

  - manuel de référence du STM32F303
    (registres, périphériques)
  - datasheet du STM32F303
    (fonctions de chaque broche)
  - manuel utilisateur de la carte d'essai STM32F3-Discovery
    (schéma de la carte)

- *Mastering STM32*, Carmine Noviello (payant)

  `http://leanpub.com/mastering-stm32`