Électronique numérique

J.-M Friedt

FEMTO-ST/département temps-fréquence

jmfriedt@femto-st.fr

transparents à `jmfriedt.free.fr`

22 novembre 2020

Plan des interventions :

7 cours/TP d'introduction au STM32 en *C baremetal* :

1. Électronique numérique et conception du circuit
   L3 : aspects analogiques, consommation électrique, lecture de datasheet
   Survol des divers périphériques qui seront abordés (RS232, SPI, timer, ADC)
   représentation des données (tailles/encodage), masques, architecture
   Rappels sur Atmega32U4 (Makefile, compilation, masques ...)
2. Premiers pas sur le STM32, adresses des périphériques, architecture
3. Fonctionnement de gcc et optimisations :
   préprocess–compilateur–assembleur–linker, passage C à assembleur, pointeurs
4. bibliothèques et séparation algorithme/matériel, simulateurs
   libopencm3, newlib & stubs, ressources requises par les bibliothèques
5. Bus de communication série, synchrone/asynchrone
6. arithmétique sur systèmes embarqués
   entiers, flottants, convertir un algorithme exprimé en nombres à virgules vers des entiers, timers
7. interruptions et acquisition de données analogiques, fréquence d'échantillonnage
   vecteurs d'interruption, gestion des horloges du STM32, ADC

Matériel pour travaux pratiques

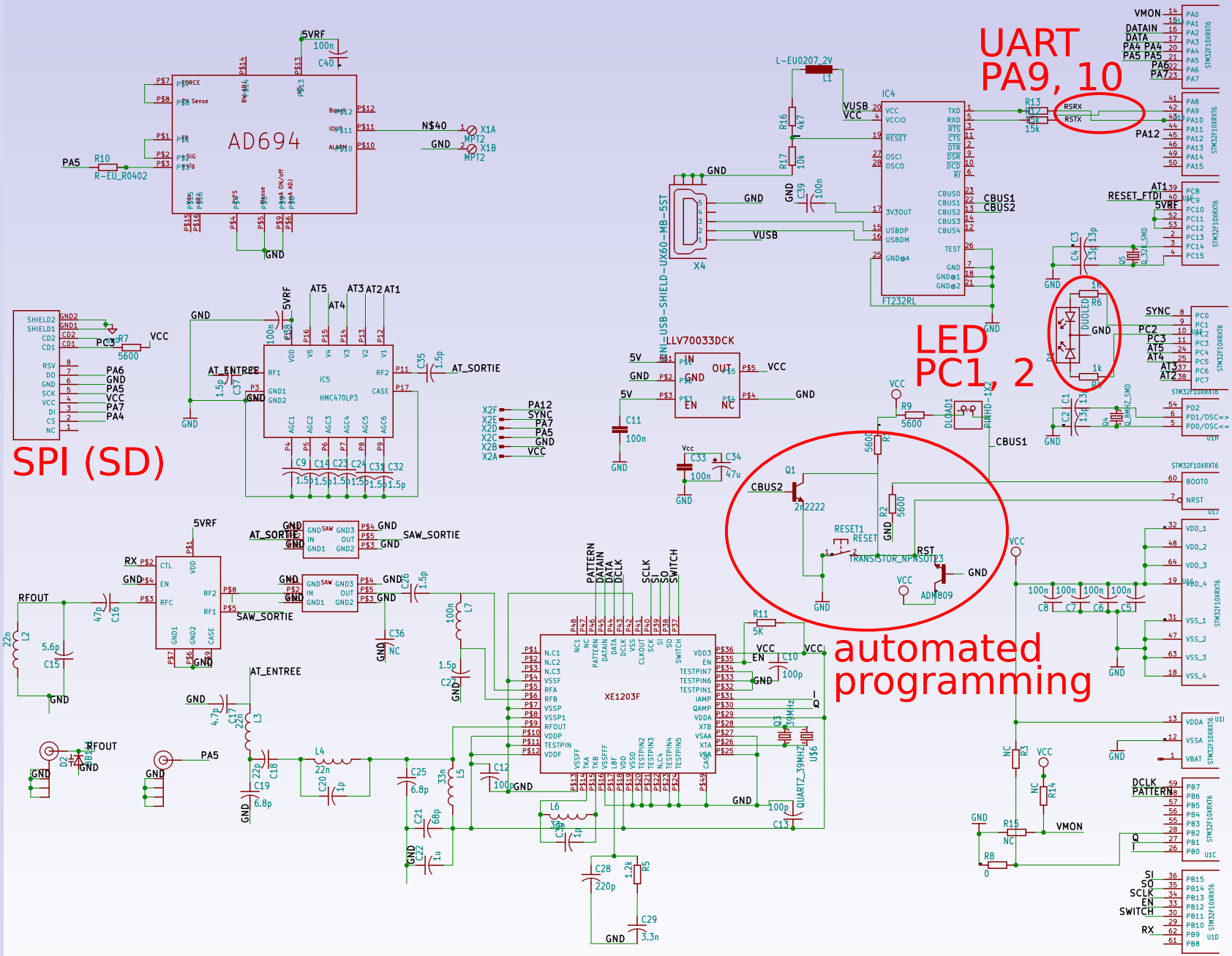
Carte dédiée incluant :

1. microcontrôleur STM32F103RCT6 (horloge 72 MHz, 48 KB RAM, 256 KB flash, suffisant pour **FreeRTOS** [1]/NuttX [2]/ZephyrOS [3])
2. PC9 est GPIO pour lire alimentation V_{USB} qui polarise aussi le Reset du convertisseur USB-RS232 FTDI $\Rightarrow$ **garder en entrée** (configuration par défaut)
3. communication par USART1 (USART2 est connecté au radiomodem)
4. LED bicolore sur PC1 & PC2
5. configuration dédiée du FT232RL pour commander les signaux de Reset et de Boot : pas de bouton poussoir à manipuler

Alternative à ce circuit : émulateur qemu supportant le STM32F103

- `https://github.com/beckus/qemu_stm32`
- même carte mémoire et périphériques mais LED sur PC1 uniquement.

1. `https://www.freertos.org/`
2. `https://nuttx.apache.org/`
3. `https://docs.zephyrproject.org/latest/boards/arm/stm32_min_dev/doc/index.html`


UART
PA9, 10
LED
PC1, 2
SPI (SD)
automated
programming
AD694
HMC470LP3
LLV70033DCK
FT232RL
XE1203F
ADM809
IC4
IC5
5VRF
VCC
GND
PA5
PA6
PA7
PA4
PA12
SYNC
RESET_FTDI
AT_ENTREE
AT_SORTIE
SAW_SORTIE
RFOUT
RX
PATTERN
DATAIN
DATA
DCLK
SCLK
SI
SO
SWITCH
VMON
CBUS1
CBUS2
RESET1
RST
QUARTZ_39MHz

Code ASCII

Comment afficher le contenu d'une variable 16 bits à l'écran

Table ASCII : correspondance entre un nombre $\in [0..127]$ et un symbole

```
ASCII(7)        Linux Programmer's Manual       ASCII(7)

NAME
ascii - the ASCII character set encoded in octal, decimal,
and hexadecimal

     2 3 4 5 6 7       30 40 50 60 70 80 90 100 110 120
    -------------     ---------------------------------
  0:   0 @ P ` p    0:    (  2  <  F  P  Z  d   n   x
  1: ! 1 A Q a q    1:    )  3  =  G  Q  [  e   o   y
  2: " 2 B R b r    2:    *  4  >  H  R  \  f   p   z
  3: # 3 C S c s    3: !  +  5  ?  I  S  ]  g   q   {
  4: $ 4 D T d t    4: "  ,  6  @  J  T  ^  h   r   |
  5: % 5 E U e u    5: #  -  7  A  K  U  _  i   s   }
  6: & 6 F V f v    6: $  .  8  B  L  V  `  j   t   ~
  7:   7 G W g w    7: %  /  9  C  M  W  a  k   u  DEL
  8: ( 8 H X h x    8: &  0  :  D  N  X  b  l   v
  9: ) 9 I Y i y    9: '  1  ;  E  O  Y  c  m   w
  A: * : J Z j z
  B: + ; K [ k {
  C: , < L \ l |
  D: - = M ] m }
  E: . > N ^ n ~
  F: / ? O _ o DEL
```

- Découper chaque quartet en symboles individuels ⇒ hexadécimal
  Exercice
- Effectuer les divisions successives ⇒ décimal
  Exercice

Un microcontrôleur ne sait pas ce qu'est `printf()` (vers quel périphérique écrire ?) ⇒ utiliser une fonction `putchar`.

Code ASCII

Comment afficher le contenu d'une variable 16 bits à l'écran

Table ASCII : correspondance entre un nombre $\in [0..127]$ et un symbole

```
ASCII(7)         Linux Programmer's Manual         ASCII(7)

NAME
ascii - the ASCII character set encoded in octal, decimal,
and hexadecimal

     2 3 4 5 6 7       30 40 50 60 70 80 90 100 110 120
   -------------     --------------------------------
  0:   0 @ P ` p     0:    (  2  <  F  P  Z  d   n   x
  1: ! 1 A Q a q     1:    )  3  =  G  Q  [  e   o   y
  2: " 2 B R b r     2:    *  4  >  H  R  \  f   p   z
  3: # 3 C S c s     3: !  +  5  ?  I  S  ]  g   q   {
  4: $ 4 D T d t     4: "  ,  6  @  J  T  ^  h   r   |
  5: % 5 E U e u     5: #  -  7  A  K  U  _  i   s   }
  6: & 6 F V f v     6: $  .  8  B  L  V  `  j   t   ~
  7:   7 G W g w     7: %  /  9  C  M  W  a  k   u  DEL
  8: ( 8 H X h x     8: &  0  :  D  N  X  b  l   v
  9: ) 9 I Y i y     9: '  1  ;  E  O  Y  c  m   w
  A: * : J Z j z
  B: + ; K [ k {
  C: , < L \ l |
  D: - = M ] m }
  E: . > N ^ n ~
  F: / ? O _ o DEL
```

- Découper chaque quartet en symboles individuels ⇒ hexadécimal

```
void affiche(short s)
{int k;char c;
 for (k=3;k>=0;k--)
   {c=s>>(4*k)&0x0f;
    if (c<10) putchar(c+'0');
       else putchar(c+'A'-10);
   }
}
```

- Effectuer les divisions successives ⇒ décimal

Exercice

Code ASCII

Comment afficher le contenu d'une variable 16 bits à l'écran

Table ASCII : correspondance entre un nombre $\in [0..127]$ et un symbole

```
ASCII(7)         Linux Programmer's Manual         ASCII(7)

NAME
ascii - the ASCII character set encoded in octal, decimal,
and hexadecimal

     2 3 4 5 6 7        30 40 50 60 70 80 90 100 110 120
   -------------      --------------------------------
  0:   0 @ P ` p     0:    (  2  <  F  P  Z  d   n   x
  1: ! 1 A Q a q     1:    )  3  =  G  Q  [  e   o   y
  2: " 2 B R b r     2:    *  4  >  H  R  \  f   p   z
  3: # 3 C S c s     3: !  +  5  ?  I  S  ]  g   q   {
  4: $ 4 D T d t     4: "  ,  6  @  J  T  ^  h   r   |
  5: % 5 E U e u     5: #  -  7  A  K  U  _  i   s   }
  6: & 6 F V f v     6: $  .  8  B  L  V  `  j   t   ~
  7:   7 G W g w     7: %  /  9  C  M  W  a  k   u  DEL
  8: ( 8 H X h x     8: &  0  :  D  N  X  b  l   v
  9: ) 9 I Y i y     9: '  1  ;  E  O  Y  c  m   w
  A: * : J Z j z
  B: + ; K [ k {
  C: , < L \ l |
  D: - = M ] m }
  E: . > N ^ n ~
  F: / ? O _ o DEL
```

- Découper chaque quartet en symboles individuels ⇒ hexadécimal

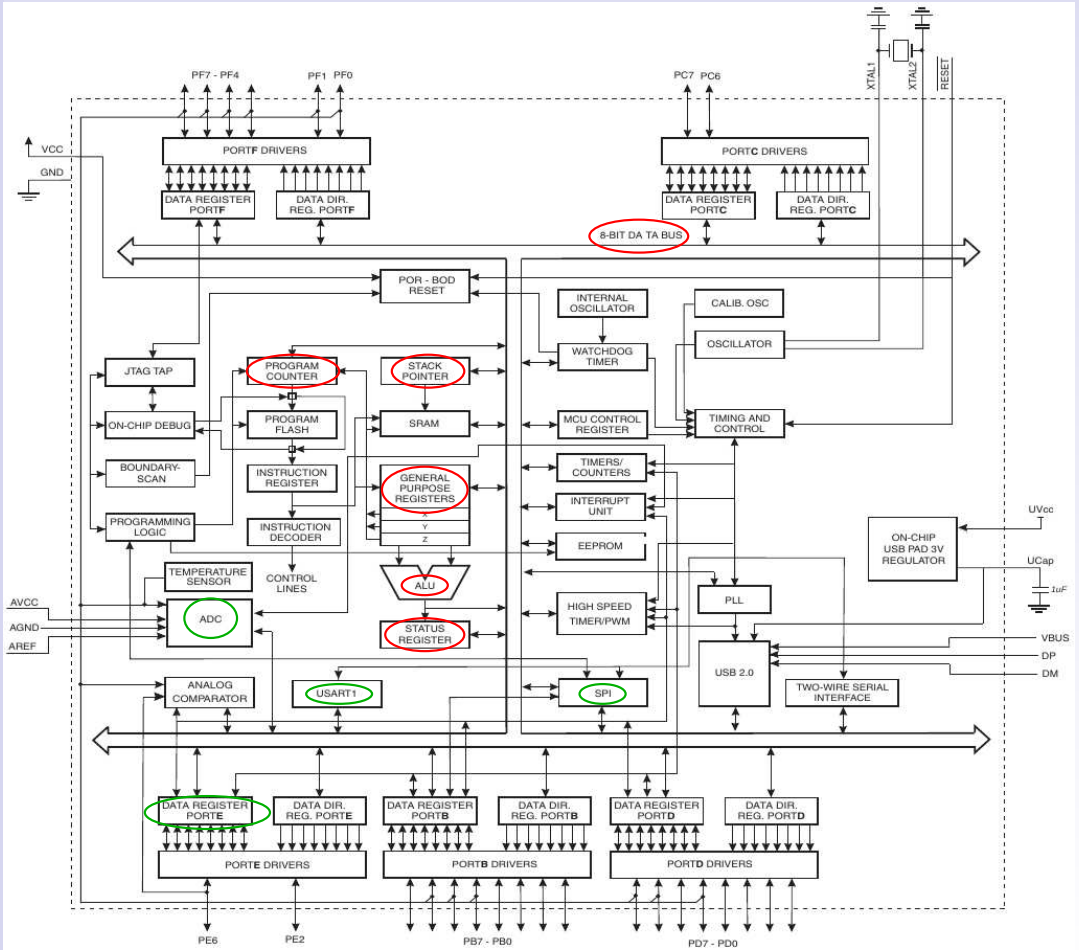
```
void affiche(unsigned short s)
{int k;char c;
 for (k=3;k>=0;k--)
   {c=s>>(4*k)&0x0f;
    if (c<10) putchar(c+'0');
       else putchar(c+'A'-10);
   }
}
```

- Effectuer les divisions successives ⇒ décimal

```
void affiche(unsigned short s)
{int k=10000;char c; // k<65536
 while (k>0)
   {c=s/k;putchar(c+'0');
    s=s-c*k;
    k/=10;
   }
}
```

Architecture interne d'un microcontrôleur

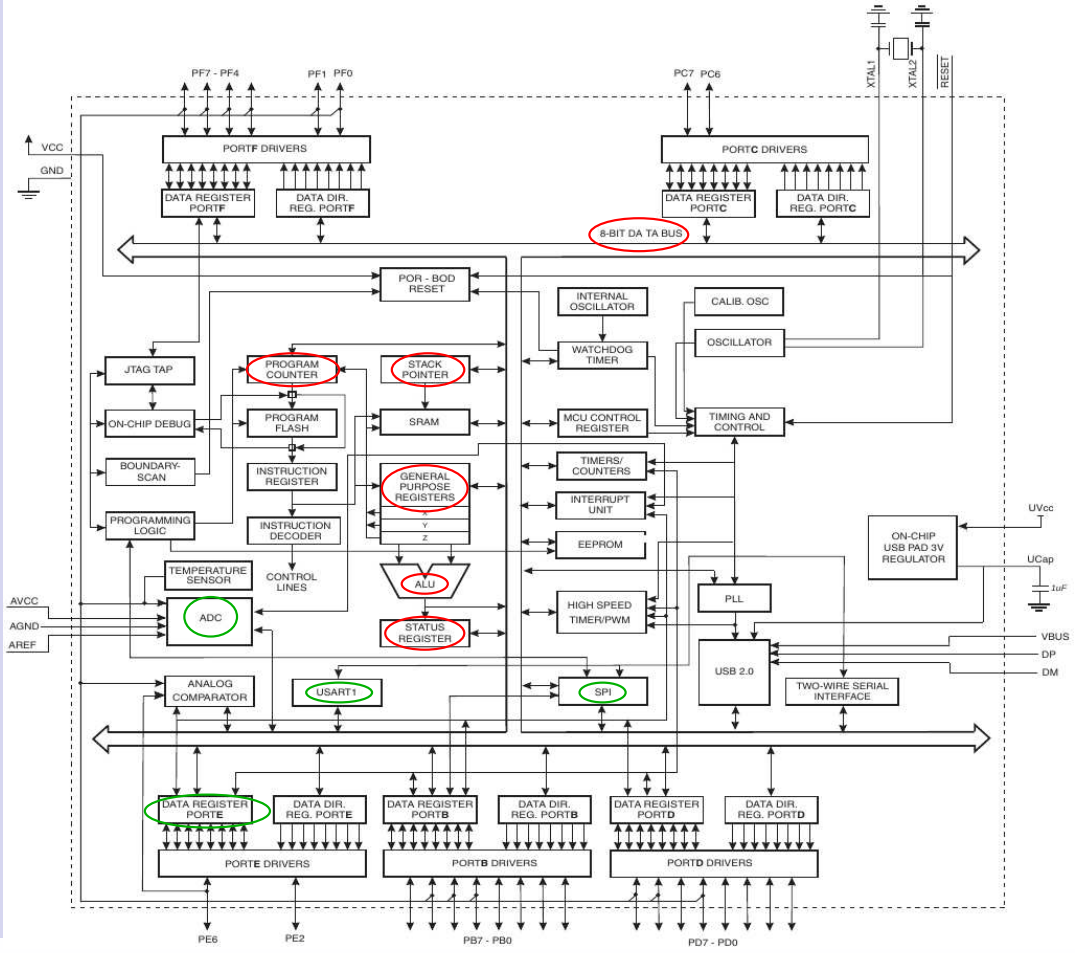
- Lien entre le matériel et le logiciel : bus d'adresse, de données et de contrôle (où, quoi, comment)
- Bus = ensemble de fils reliant deux périphériques (liaison parallèle)



- ALU : Unité Arithmérique et Logique
- GPIO
- timer
- ADC
- PC
- SP

Architecture interne d'un microcontrôleur

- Lien entre le matériel et le logiciel : bus d'adresse, de données et de contrôle (où, quoi, comment)
- Bus = ensemble de fils reliant deux périphériques (liaison parallèle)



6.4 GPIO registers

This section gives a detailed description of the GPIO registers.
For a summary of register bits, register address offsets and reset values, refer to *Table 27*.

The GPIO registers can be accessed by byte (8 bits), half-words (16 bits) or words (32 bits).

6.4.1 GPIO port mode register (GPIOx_MODER) (x = A..C and H)

Address offset: 0x00

Reset values:

- 0x0C00 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

6.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..C and H)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

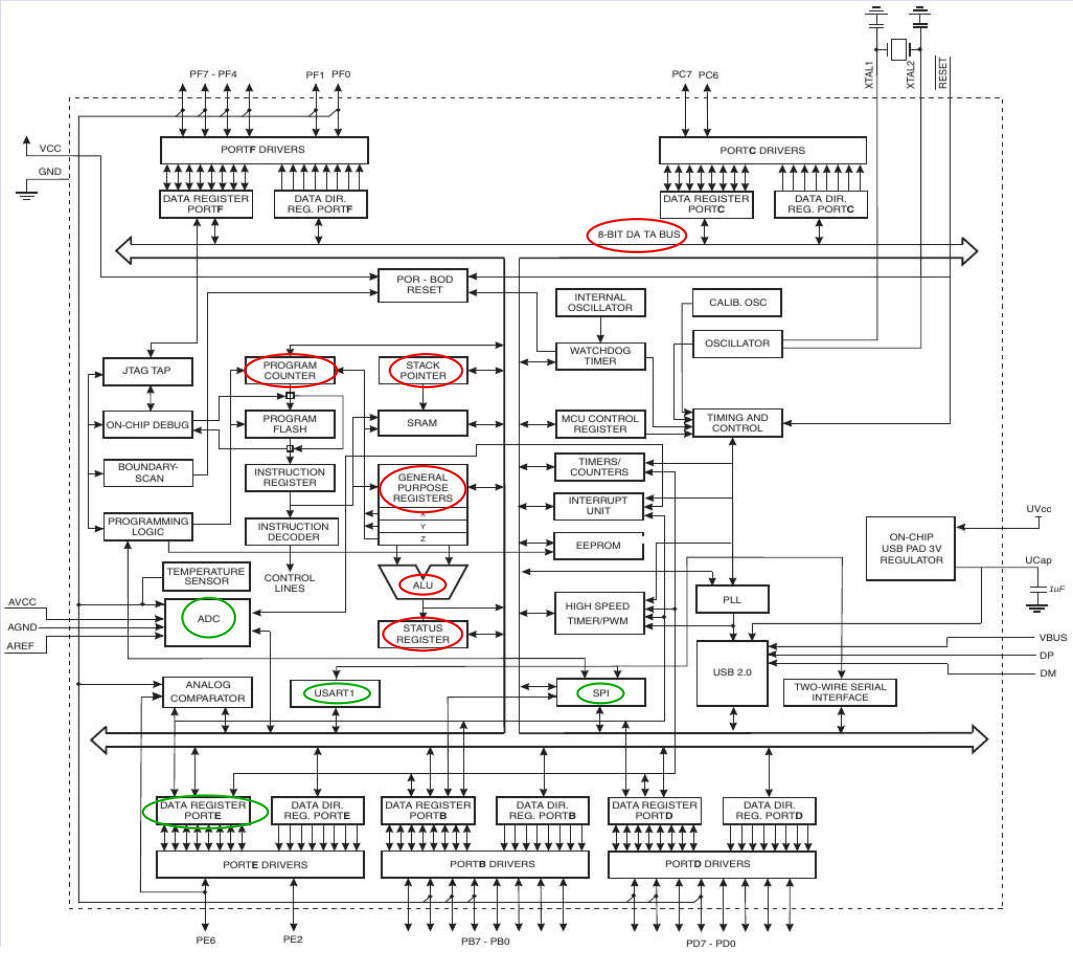
Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy**: Port x configuration bits (y = 0..15)
These bits are written by software to configure the output type of the I/O port.
0: Output push-pull (reset state)
1: Output open-drain

RM0401 : datasheet STM32F410

Architecture interne d'un microcontrôleur

- Lien entre le matériel et le logiciel : bus d'adresse, de données et de contrôle (où, quoi, comment)
- Bus = ensemble de fils reliant deux périphériques (liaison parallèle)



Memory map and register boundary addresses

See the datasheet corresponding to your device for a comprehensive diagram of the memory map.

The following table gives the boundary addresses of the peripherals available in the devices.

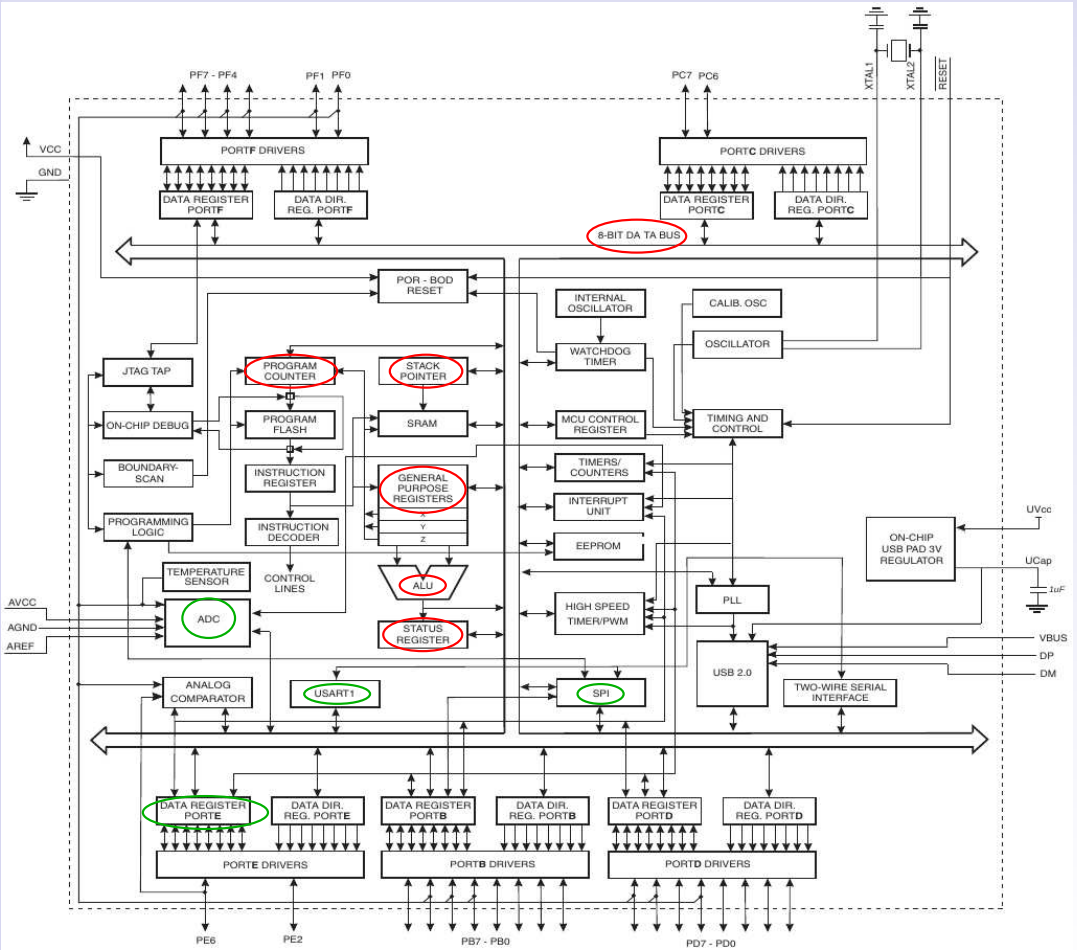
Table 1. Register boundary addresses

Bus	Boundary address	Peripheral
	0xE010 0000 - 0xFFFF FFFF	Reserved
Cortex®-M4	0xE000 0000 - 0xE00F FFFF	Cortex-M4 internal peripherals
	0x5000 0000 - 0xDFFF FFFF	Reserved
AHB1	0x4008 0400 - 0x4FFF FFFF	Reserved
	0x4008 0000 - 0x4008 03FF	RNG
	0x4002 6800 - 0x4007 FFFF	Reserved
	0x4002 6400 - 0x4002 67FF	DMA2
	0x4002 6000 - 0x4002 63FF	DMA1
	0x4002 5000 - 0x4002 4FFF	Reserved
	0x4002 3C00 - 0x4002 3FFF	Flash interface register
	0x4002 3800 - 0x4002 3BFF	RCC
	0x4002 3400 - 0x4002 37FF	Reserved
	0x4002 3000 - 0x4002 33FF	CRC
	0x4002 2800 - 0x4002 2FFF	Reserved
	0x4002 2400 - 0x4002 27FF	LPTIM1
	0x4002 2000 - 0x4002 23FF	Reserved
	0x4002 1C00 - 0x4002 1FFF	GPIOH
	0x4002 0C00 - 0x4002 1BFF	Reserved
	0x4002 0800 - 0x4002 0BFF	GPIOC
	0x4002 0400 - 0x4002 07FF	GPIOB
	0x4002 0000 - 0x4002 03FF	GPIOA

RM0401 : datasheet STM32F410

Architecture interne d'un microcontrôleur

- Lien entre le matériel et le logiciel : bus d'adresse, de données et de contrôle (où, quoi, comment)
- Bus = ensemble de fils reliant deux périphériques (liaison parallèle)



/libopencm3/stm32/common/gpio_common_f234.h : #define GPIOC GPIO_PORT_C_BASE

/libopencm3/stm32/f4/memorymap.h : #define GPIO_PORT_C_BASE (PERIPH_BASE_AHB1 + 0x0800)

/libopencm3/stm32/f4/memorymap.h : #define PERIPH_BASE_AHB1 (PERIPH_BASE + 0x20000)

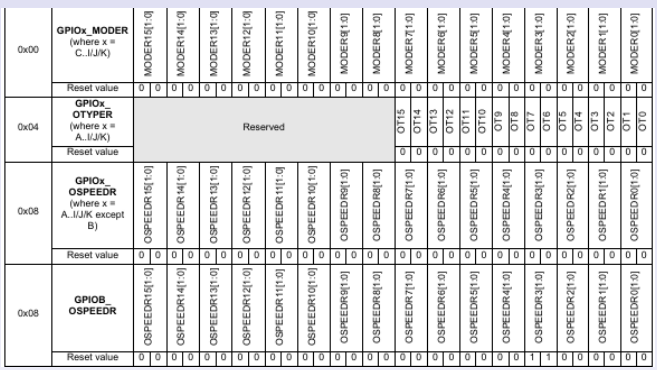
/libopencm3/stm32/f4/memorymap.h : #define PERIPH_BASE (0x40000000U)

$\Rightarrow$ GPIOC = 0x40020800

Comment connaître l'adresse d'un périphérique ...

Liste des registres dans la datasheet du microcontrôleur (STM32F4) ⇒ utilisation des masques pour définir un bit individuel

0x4002 3800 - 0x4002 3BFF	RCC	AHB1	*Section 7.3.24: RCC register map on page 265*
0x4002 3000 - 0x4002 33FF	CRC		*Section 4.4.4: CRC register map on page 115*
0x4002 2800 - 0x4002 2BFF	GPIOK		*Section 8.4.11: GPIO register map on page 287*
0x4002 2400 - 0x4002 27FF	GPIOJ		
0x4002 2000 - 0x4002 23FF	GPIOI		
0x4002 1C00 - 0x4002 1FFF	GPIOH		*Section 8.4.11: GPIO register map on page 287*
0x4002 1800 - 0x4002 1BFF	GPIOG		
0x4002 1400 - 0x4002 17FF	GPIOF		
0x4002 1000 - 0x4002 13FF	GPIOE		
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		



Extraits de datasheets du STM32F4

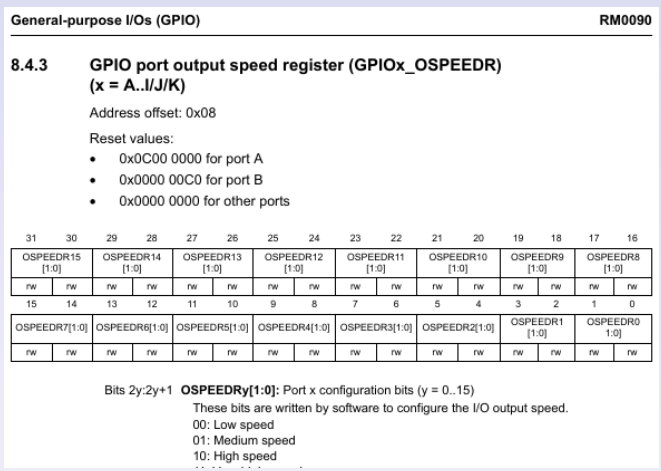
Une documentation décrivant le matériel de chaque modèle de STM32 (e.g. `www.st.com/resource/en/datasheet/dm00037051.pdf`)

Une documentation décrivant les registres de chaque modèle de STM32 (e.g. `www.st.com/resource/en/reference_manual/dm00031020.pdf`)

Comment connaître l'adresse d'un périphérique ...

Liste des registres dans la datasheet du microcontrôleur (STM32F4) ⇒ utilisation des masques pour définir un bit individuel

0x4002 3800 - 0x4002 3BFF	RCC	AHB1	Section 7.3.24: RCC register map on page 265
0x4002 3000 - 0x4002 33FF	CRC		Section 4.4.4: CRC register map on page 115
0x4002 2800 - 0x4002 2BFF	GPIOK		Section 8.4.11: GPIO register map on page 287
0x4002 2400 - 0x4002 27FF	GPIOJ		
0x4002 2000 - 0x4002 23FF	GPIOI		
0x4002 1C00 - 0x4002 1FFF	GPIOH		
0x4002 1800 - 0x4002 1BFF	GPIOG		
0x4002 1400 - 0x4002 17FF	GPIOF		
0x4002 1000 - 0x4002 13FF	GPIOE		Section 8.4.11: GPIO register map on page 287
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		



General-purpose I/Os (GPIO) RM0090

8.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..I/J/K)

Address offset: 0x08

Reset values:

- 0x0C00 0000 for port A
- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **OSPEEDRy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O output speed.
00: Low speed
01: Medium speed
10: High speed

Extraits de datasheets du STM32F4

Une documentation décrivant le matériel de chaque modèle de STM32 (e.g. `www.st.com/resource/en/datasheet/dm00037051.pdf`)

Une documentation décrivant les registres de chaque modèle de STM32 (e.g. `www.st.com/resource/en/reference_manual/dm00031020.pdf`)

Obtenir et compiler une archive

1. Versionnemment d'un projet : `cvs`, `svn`, `git`, `mercurial`
   `git clone https://github.com/jmfriedt/stm32`
2. Compilation d'un projet : `configure` (autoconf), `make`, `cmake`
3. Format d'un Makefile :
   - un objectif : `all`
   - chaque objectif dépend de conditions à résoudre
     `mon_programme: objet1.o objet2.o`
   - chaque objectif fournit la méthode pour passer des dépendances à cet objectif, préfixé de TAB
     `gcc -o mon_programme objet1.o objet2.o`

```
all: mon_programme

mon_programme: objet1.o objet2.o
	gcc -o $@ objet1.o objet2.o # $@ (cible) = mon_programme

objet1.o: objet1.c
	gcc -c objet1.c

objet2.o: objet2.c
	gcc -c $<                                  # $< = premiere dependance
clean:
	rm *.o mon_programme
```

Archive pour TP

1. Cette archive est compatible avec de nombreux microcontrôleurs ARM ⇒ lien symbolique (`ln -s`) de `Makefile.stm32f1` vers `Makefile`
2. `Makefile` contient les méthodes `flash` et `run` (qemu)
3. s'appuie sur `libopencm3`[4] pour accéder aux périphériques : `/home/jmfriedt/libopencm3`
4. pour reproduire l'installation chez soi : `http://jmfriedt.free.fr/ftdi_stm32_prog.tar.gz` pour manipuler les broches Reset et Boot lors du transfert du programme
5. noter l'existence de CMSIS *Cortex Microcontroller Software Interface Standard*[5]
6. quelques fonctions de base : `mon_putchar (char)`, `mon_puts(char*)`, `init_gpio()`, `led_set(int)`, `led_clr(int)`, `usart_setup()`, `clock_setup()`
7. Principale difficulté d'appréhender le STM32 : **gestion des horloges** qui cadencent les périphérique (inactives par défaut)

4. `https://github.com/libopencm3/libopencm3`
5. https ://developer.arm.com/tools-and-software/embedded/cmsis

Objectif

1. Récupérer l'archive `github.com/jmfriedt/stm32`
2. Compiler et flasher ce premier exemple[6]
   `stm32flash.sh -w $(PROJ_NAME).bin /dev/ttyUSB0`
3. Compter de 0 à 255 et afficher la variable en hexadécimal
4. Compter de 0 à 999 et afficher la variable en décimal
5. Considérant une fréquence d'horloge cadençant le DDS de 70 MHz, calculer et afficher en hexadécimal les deux mots nécessaires à configurer le DDS pour générer un signal à 4 MHz, *sans utiliser de calcul sur des flottants*.

Ressources :

1. Paquets `gcc-arm-none-eabi` et `binutils-arm-none-eabi` de Debian/GNU Linux
2. VirtualBox GNU/Linux : `jmfriedt.org/debian_redpitaya.ova` (2,5 GB, `root` passwd=root, pas de `user` passwd ; SHIFT-ALT pour azerty↔qwerty)

6. installation décrite à `http://jmfriedt.free.fr/ftdi_stm32_prog.tar.gz`