Bienvenue au cours

Systèmes embarqués et robotique

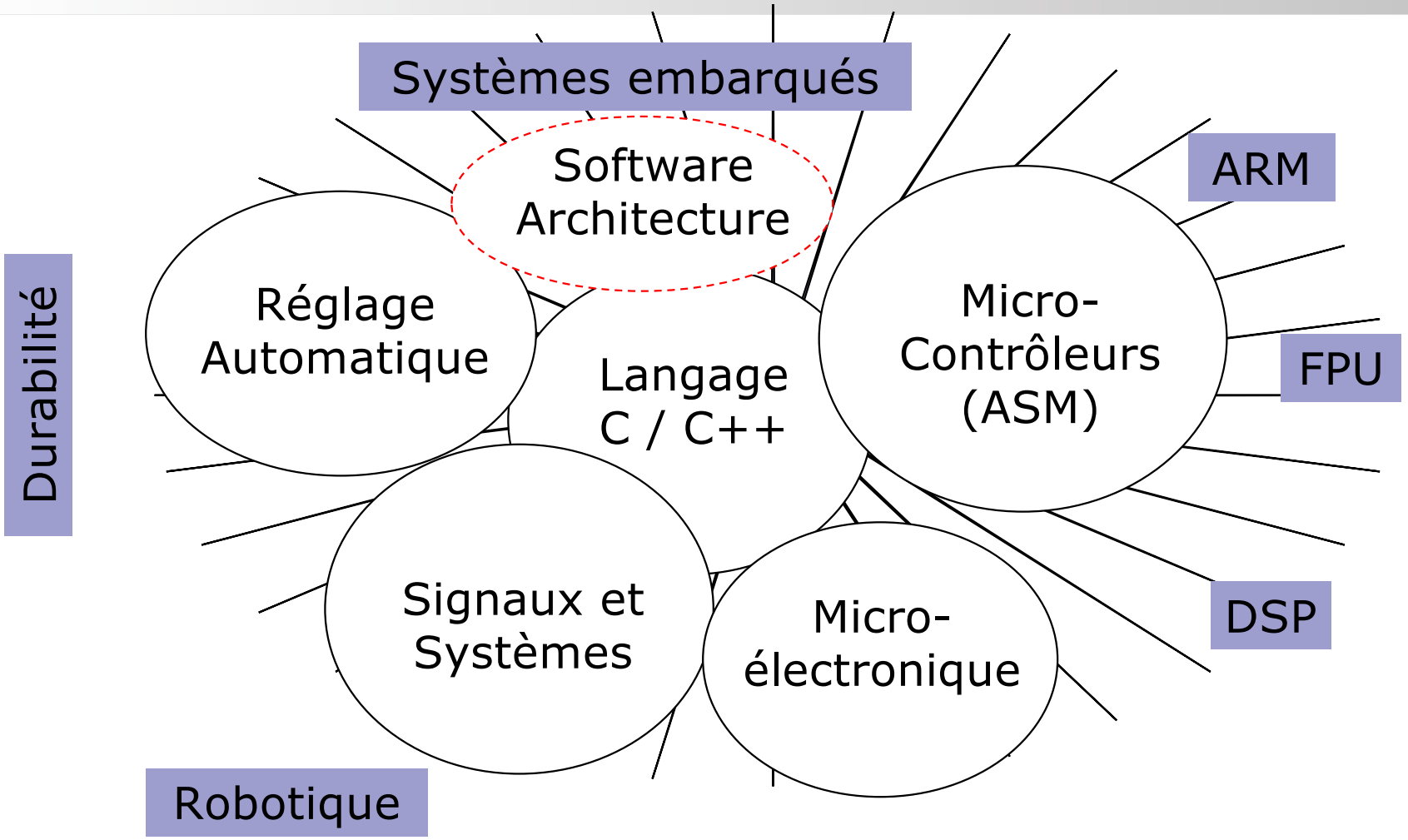
Prof. Francesco Mondada
Dr. Frank Bonnet
IEM - STI - EPFL

But du cours:

Vous voulez développer un robot innovant?

- Mécanique
- Électronique
- Logiciel temps-réel
- Contrôle
- Traitement de capteurs
- Contrôle de actuateurs
- Navigation …

www.leobotics.fr


Systèmes embarqués
Software Architecture
Réglage Automatique
Langage C / C++
Micro-Contrôleurs (ASM)
ARM
FPU
DSP
Micro-électronique
Signaux et Systèmes
Robotique
Durabilité

But du cours:

Programmation de systèmes embarqués en C appliqué à la robotique

ce qui implique:

- compréhension contexte / intégration des domaines appris

avec une attention spécifique à:

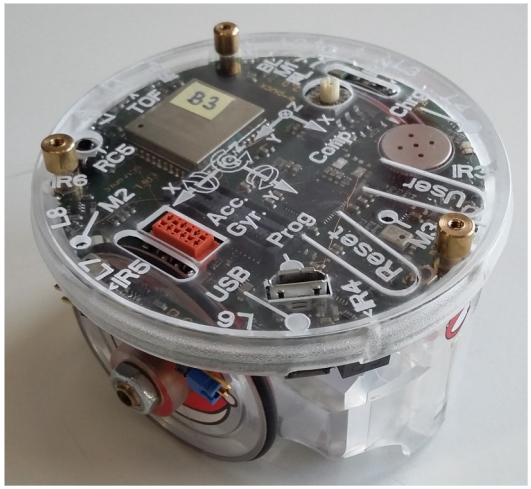
- fonctionnement d'un cross-compilateur
- code généré par un compilateur en fonction du code C
- optimisation de code / des ressources embarquées / OS
- utilisation de fonctionnalités avancées d'un microcontrôleur
- utilisation dans un contexte robotique: capteurs, actuateurs, temps-réel
- aspects de durabilité liés au microcontrôleur
- contexte de travail en groupe et utilisation de l'IA

Application:

Programmation d'un robot mobile

Langage C / C++

ARM



Micro-Contrôleurs (ASM)

DSP

Signaux et Systèmes

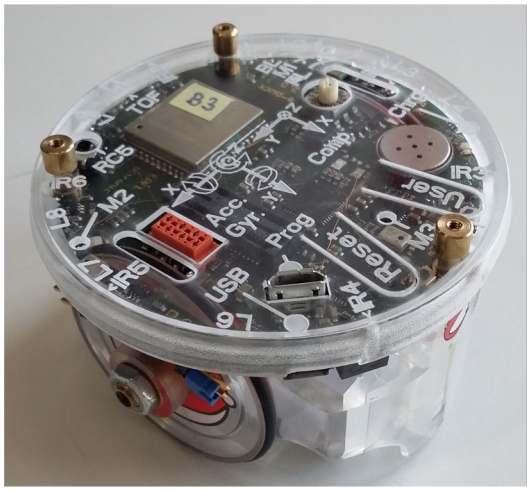
FPU

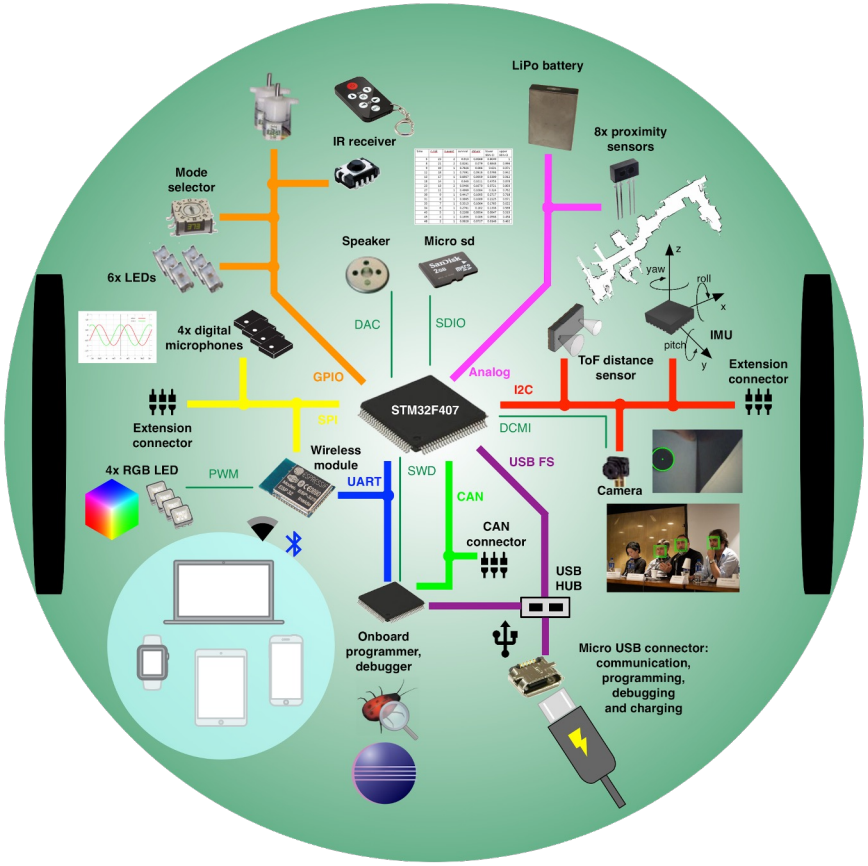
Electronique

Git

Réglage Automatique

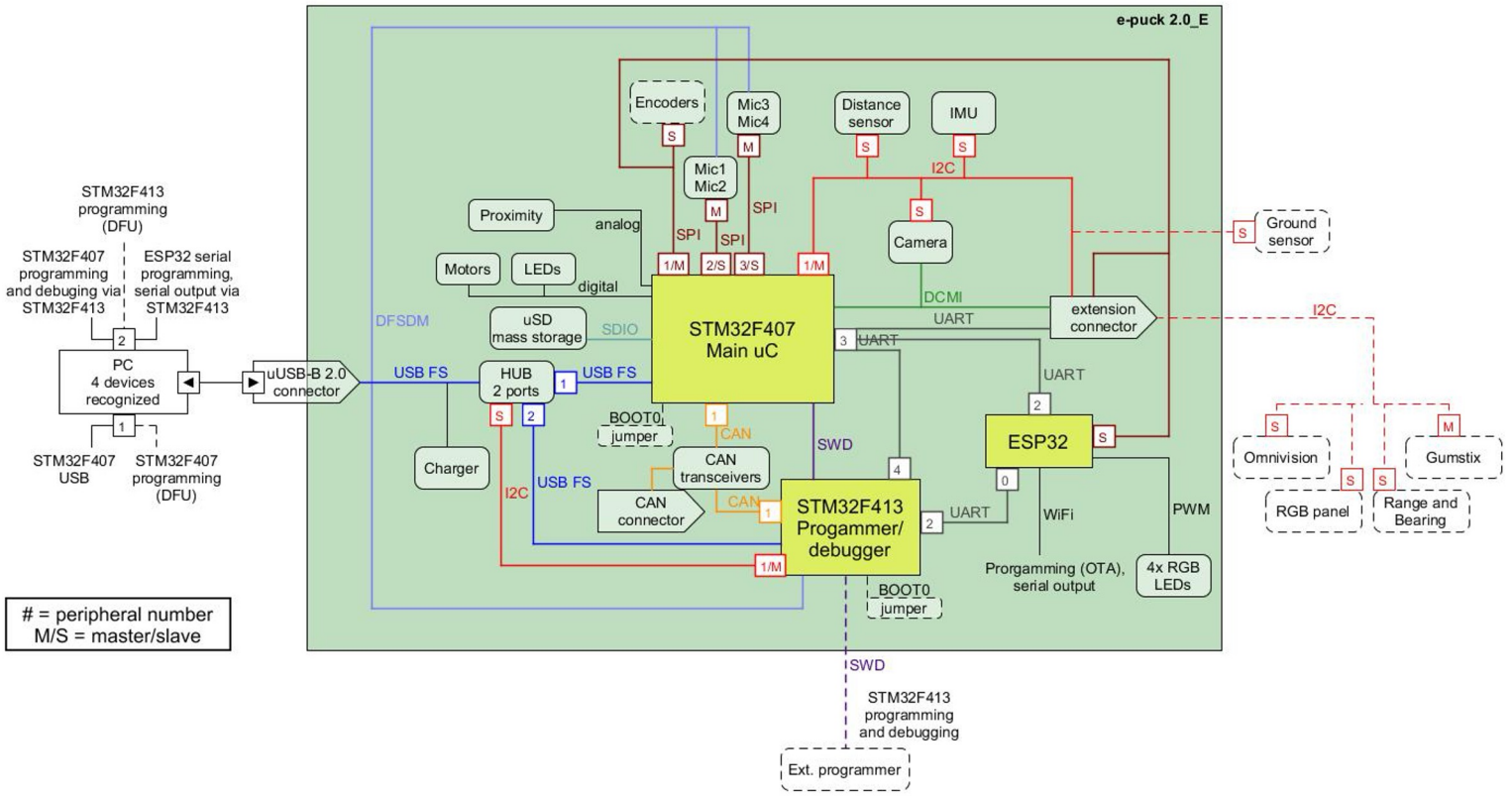
e-puck2: fonctionnalités





http://www.gctronic.com

e-puck2: fonctionnalités



Plan du cours en deux parties :

1. Intégration des connaissances asm, C
   But: intégrer les mécanismes C / assembleur sur ARM
   Période: première moitié du semestre
   Contrôle: test (40% de la note) closed book

2. Réalisation
   But: intégration de la matière dans un projet
   Période: milieu - deuxième moitié du semestre
   Contrôle: miniprojet (60% de la note)

Plan détaillé:

Date					
18.Feb.25		Cours 1		**08.Apr.25**	8:15 test valable pour 40%
20.Feb.25		TPIntro		10.Apr.25	miniprojet
25.Feb.25		Cours 2		**15.Apr.25**	miniprojet
27.Feb.25		TP1		17.Apr.25	miniprojet
04.Mar.25		Cours 3		**22.Apr.25**	Vacances de Pâques
06.Mar.25		TP2		24.Apr.25	Vacances de Pâques
11.Mar.25		Cours 4		**29.Apr.25**	miniprojet
13.Mar.25		TP3		01.May.25	miniprojet
18.Mar.25		Cours 5		**06.May.25**	miniprojet
20.Mar.25		TP4		08.May.25	miniprojet
25.Mar.25		Cours 6		**13.May.25**	présentations miniprojet
27.Mar.25		TP5		15.May.25	présentations miniprojet
01.Apr.25		8:15 test à blanc		**20.May.25**	présentations miniprojet
03.Apr.25		miniprojet		22.May.25	présentations miniprojet
				27.May.25	présentations miniprojet
				29.May.25	ascension

Date					
18.Feb.25		Cours 1		**08.Apr.25**	8:15 test valable pour 40%
20.Feb.25		TPIntro		10.Apr.25	miniprojet
25.Feb.25		Cours 2		**15.Apr.25**	miniprojet
27.Feb.25		TP1		17.Apr.25	miniprojet
04.Mar.25		Cours 3		**22.Apr.25**	Vacances de Pâques
06.Mar.25		TP2		24.Apr.25	Vacances de Pâques
11.Mar.25		Cours 4		**29.Apr.25**	miniprojet
13.Mar.25		TP3		01.May.25	miniprojet
18.Mar.25		Cours 5		**06.May.25**	miniprojet
20.Mar.25		TP4		08.May.25	miniprojet
25.Mar.25		Cours 6		**13.May.25**	miniprojet
27.Mar.25		TP5		15.May.25	miniprojet - deadline 23h
01.Apr.25		8:15 test à blanc		**20.May.25**	présentations miniprojet
03.Apr.25		miniprojet		22.May.25	présentations miniprojet
				27.May.25	présentations miniprojet
				29.May.25	ascension

Fonctionnement du cours:

Ce cours n'a pas de polycopié. Il se base sur:

- Les slides du cours sur moodle.
- Des documents techniques (data sheets, reference manuals…) d'usage courant.
- **Quizz hebdomadaire sur moodle.**
- Les TP indispensables pour la compréhension.
- Github avec les exercices des TP et les corrigés.
- Un forum / un wiki.
- Un GPT spécifique (feedback bienvenu).
- Les supports des autres cours.
- Un feedback anonyme.

Matériel du cours:

Tout le matériel de reference se trouve sur moodle. Les TP sur Github

Documents techniques conseillés:

- Cortex-M4 Technical Reference Manual
  Référence pour la programmation du processeur
- Documentation STM32F (datasheet)
  Description du microcontrôleur, ses périphériques etc.
- STM32F Programmer's Reference Manual
  Référence pour l'architecture/périphériques du microcontrôleur

Utilisation de l'IA générative (chatbots)

- N'utilisez pas un chatbot générique pour chercher de l'information. Il y a des chatbots spécifiques plus intéressants: graphsearch.epfl.ch
- Les chatbots savent bien générer du code, mais attention à la qualité, surtout si dans un domaine spécifique.
- L'utilisation des chatbots demande des connaissances de base, c'est le but de ce cours.
- Avec les chatbot on peut accélérer l'exécution d'une tâche, mais on va apprendre moins. Dans ce cours le but est d'apprendre.
- Exploitez les chatbot pour élargir vos réflexions (brainstorming) ou pour vous aider à vous exercer dans un domaine.
- Attention à l'impact écologique de cet usage!

Travaux pratiques (TP):

Les TP permettent d'acquérir les connaissances données lors du cours par leur mise en pratique dans un problème concret, comme il pourrait apparaître dans un projet réel.

La donnée du TP est sur GitHub. En salles **ME D2 2519** et **ME D2 2524**.

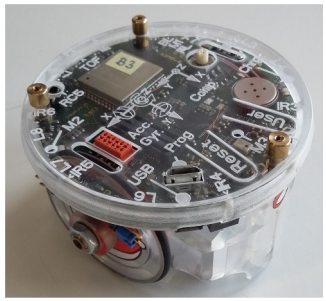
Les TPs sont réalisés par équipes de deux étudiants. Créer les groupes aujourd'hui sur moodle: merci de les communiquer avant demain (mercredi 21) midi.

Organisation: 5h les jeudi (10h15 -> 15h00)

- 10h15: Introduction au TP (lu à l'avance) ensuite réponse aux questions précédentes. ATTENTION: venez à l'heure, intro au début.
- 12h : (optionnel) approfondissement du TP
- 12h30 pause repas…

Travaux pratiques (TP):

Dans les salles DLL (Discovery Learning Laboratory)



Robot e-puck2

Travaux pratiques (TP): demande d'assistant

Request for an assistant

Small form to request the help of an assistant during the practical sessions

francesco.mondada@epfl.ch Changer de compte

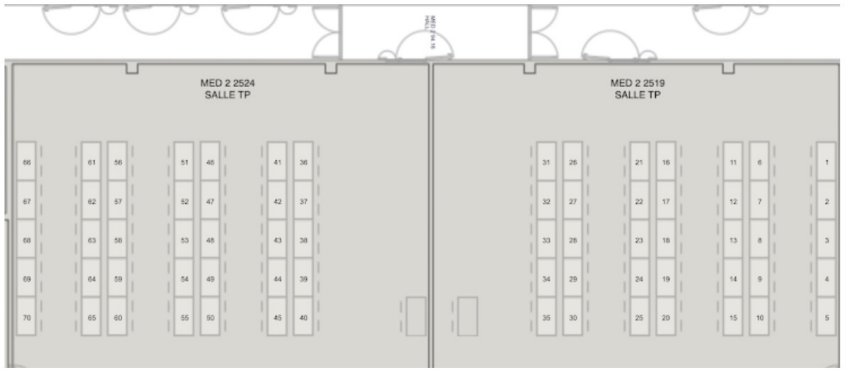
Votre adresse e-mail est enregistrée lorsque vous envoyez ce formulaire

*Obligatoire

Small description of your problem/question (helps to choose the assistant :-)) *

Votre réponse

Your place number or Zoom link *



Votre réponse

Envoyer

Effacer le formulaire

Objectif du miniprojet:

Mettre en pratique ce qui a été vu au cours dans un projet robotique complet et motivant.

Donnée miniprojet:

Le but du miniprojet est de partir sur la base des éléments que vous aurez vu lors des TPs 1-5 pour créer plusieurs tâches plus complexes à résoudre par le robot e-puck2.

Donnée:

Vous êtes libres de déterminer vous-même les tâches que le robot doit effectuer, et donc la forme de la démonstration de votre programme. Les contraintes sont les suivantes :

1) Le projet doit être fait sur la base de la librairie e-puck2_main-processor vue lors des TPs 4-5.

2) Vous devez obligatoirement utiliser les éléments suivants du robot e-puck2 dans votre projet :

a. **Les deux moteurs pas-à-pas**. Par exemple Régulation PID, odométrie précise, forme géométrique, etc.
b. **Un des capteurs de distance** (Capteurs de proximités infrarouges ou capteur Time-of-Flight). Par exemple détection d'obstacle petite ou grande distance.
c. **Un capteur parmi ceux que vous avez investigués pendant les TPs 3-5, donc un capteur parmi : la caméra, les micros, l'IMU**. Par exemple avec la caméra : détection d'objet, suivi de lignes. Par exemple avec les micros : détection de sons sur la base de l'amplitude et/ou fréquence. Par exemple avec l'accéléromètre, détection d'un plan incliné, d'un choc. Par exemple avec le gyroscope, détection du mouvement d'un plan incliné.

Donnée:

3) Chaque capteur/actuateur doit être utilisé/géré dans une Thread, à l'image de leur utilisation dans les TPs 4-5. La création de threads doit suivre les besoins, mais toujours de manière à respecter les taux de rafraichissements des capteurs et en faisant un usage intelligent des outils disponibles dans ChibiOS (messagebus, sémaphores, etc.).

4) Le code doit être rendu sous la forme d'une librairie (avec divers .c/.h) qui s'intègre avec la librairie e-puck2_main-processor que vous avez utilisé lors des Travaux pratiques.

Voici donc un exemple de projet : Le robot e-puck2 reproduit les mouvements d'une balle sur un plan incliné en utilisant l'accéléromètre comme détecteur de la direction du vecteur gravité. Le robot "rebondit" lorsqu'il détecte un obstacle avec ses capteurs de proximité infrarouges.

Miniprojet :

Extension d'un des travaux pratiques et réalisation d'une librairie avec diverses fonctionnalités exploitant les périphériques du robot e-puck2

GCtronic Award

to

Student #1
Student #2

Prix décerné au meilleur miniprojet

Microinformatique

Introduction au STM32F4

Prof. Francesco Mondada / Dr. Frank Bonnet
IEM - STI - EPFL

CISC (Complex Instruction Set Computer)

- Grand nombre d'instructions
- Instructions complexes, plus de modes d'adressage
- Exécution en un ou plusieurs cycles
- Hardware complexe: plus de transistors
- Code plus court, instructions qui traitent plus, moins besoin de mémoire
- Axé sur hardware, peu de RAM

RISC (Reduced Instruction Set Computer)

- Nombre réduit d'instructions
- Instructions simples, modes d'adressage simples
- Exécution principalement en un cycle (mieux prévisible > pipeline)
- Hardware plus simple: moins de transistors
- Code plus long, plus axé sur la mémoire, besoin de plus de mémoire
- Axé sur software

Introduction: the RISC vs CISC

"One indication of complexity is the size of the documentation.

The ISA manual for x86-32 is 2,198 pages or 2,186,259 words. The RISC-V equivalents are 236 pages or 76,702 words.

If someone were to read manuals as an (incredibly boring) fulltime job—eight hours a day for five days a week—it would take a month to read the x86-32 manual but less than a day to read the RISC-V manual."

ISA = Instruction Set Architecture

From D. Patterson, "Reduced Instruction Set Computers Then and Now," in Computer, vol. 50, no. 12, pp. 10-12, December 2017.

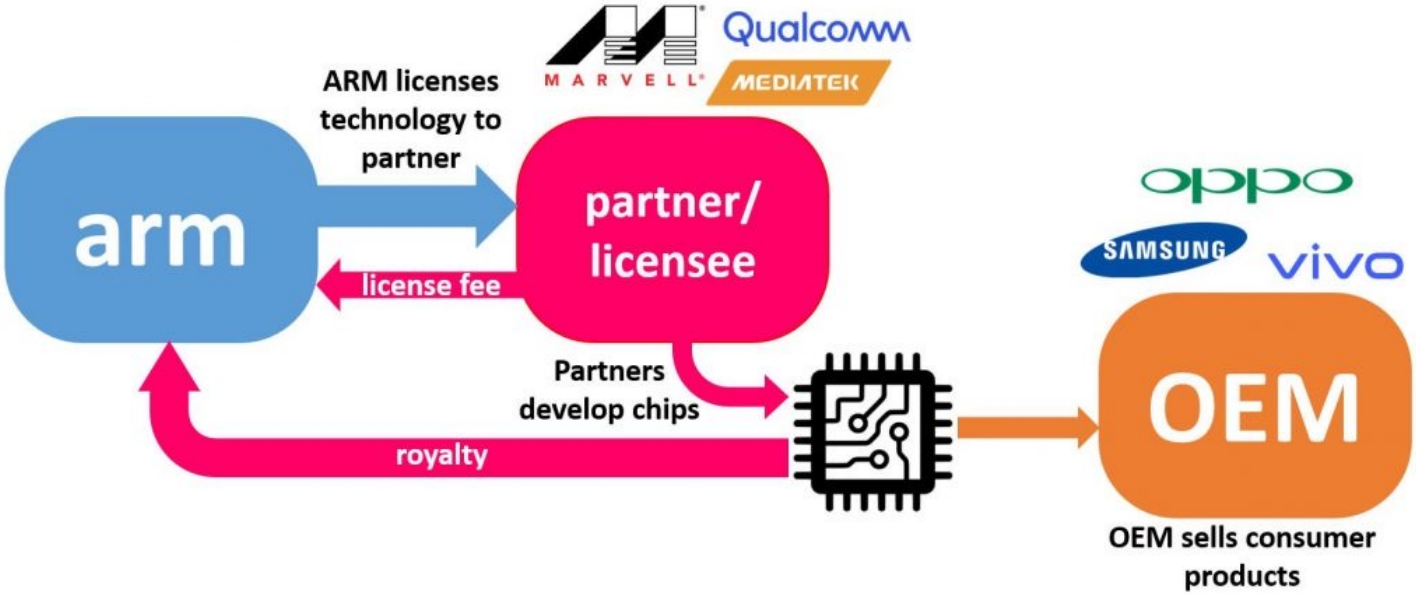
Introduction: le futur est RISC… (une entreporise: ARM)

(Advanced (Acorn) RISC Machine, RISC = reduced instruction set computer)

"The 80×86 ISA (instruction set architecture) dominated sales in the PC era, but RISC architectures are kings of the post-PC era. Annual 80×86 shipments peaked in 2011 at 365 million and have been declining about 8 percent annually since; Intel made fewer 80×86 chips in 2016 than in 2007. While the 80×86 dominates the cloud portion of the post-PC era, it's estimated that Amazon, Google, and Microsoft clouds collectively contain only 10 million servers. Although these chips are expensive, their volume is negligible; in 2017, 10 million RISC chips ship every four hours. RISCs make up 99 percent of microprocessor volume today."

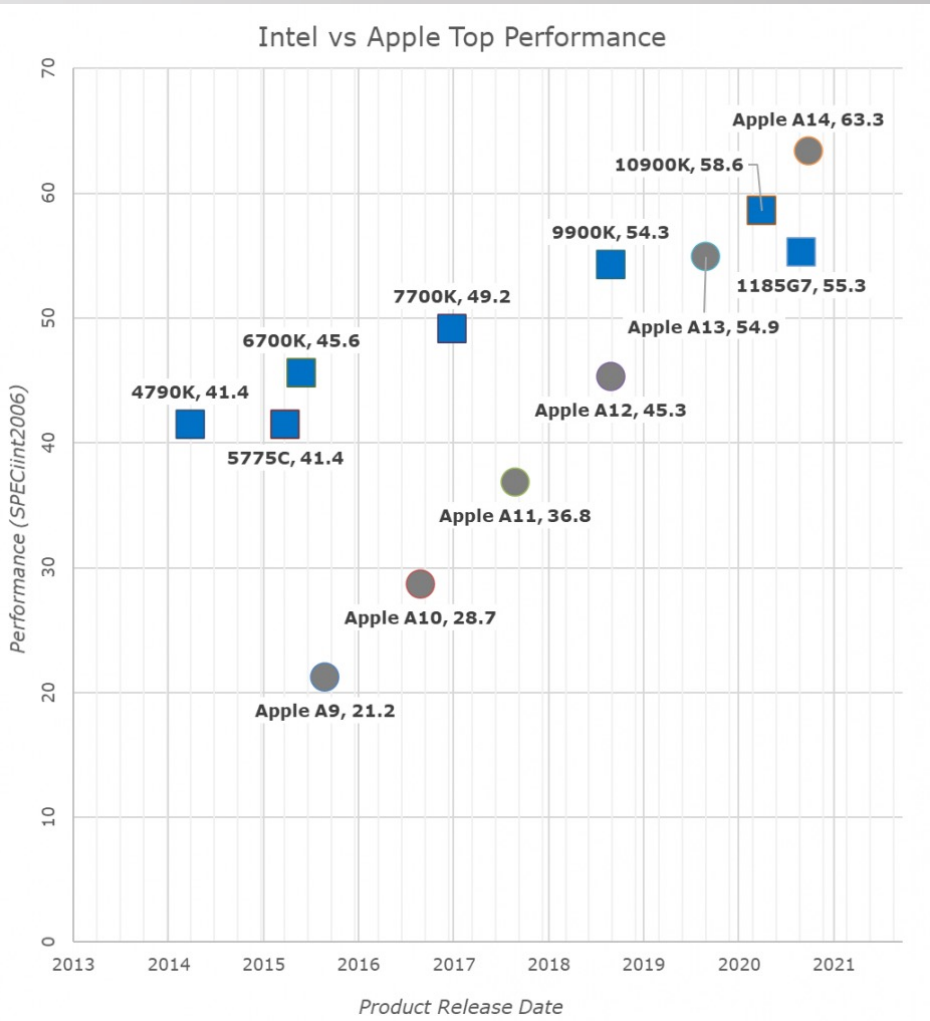
From D. Patterson, "Reduced Instruction Set Computers Then and Now," in Computer, vol. 50, no. 12, pp. 10-12, December 2017.

ARM: un modèle particulier de business



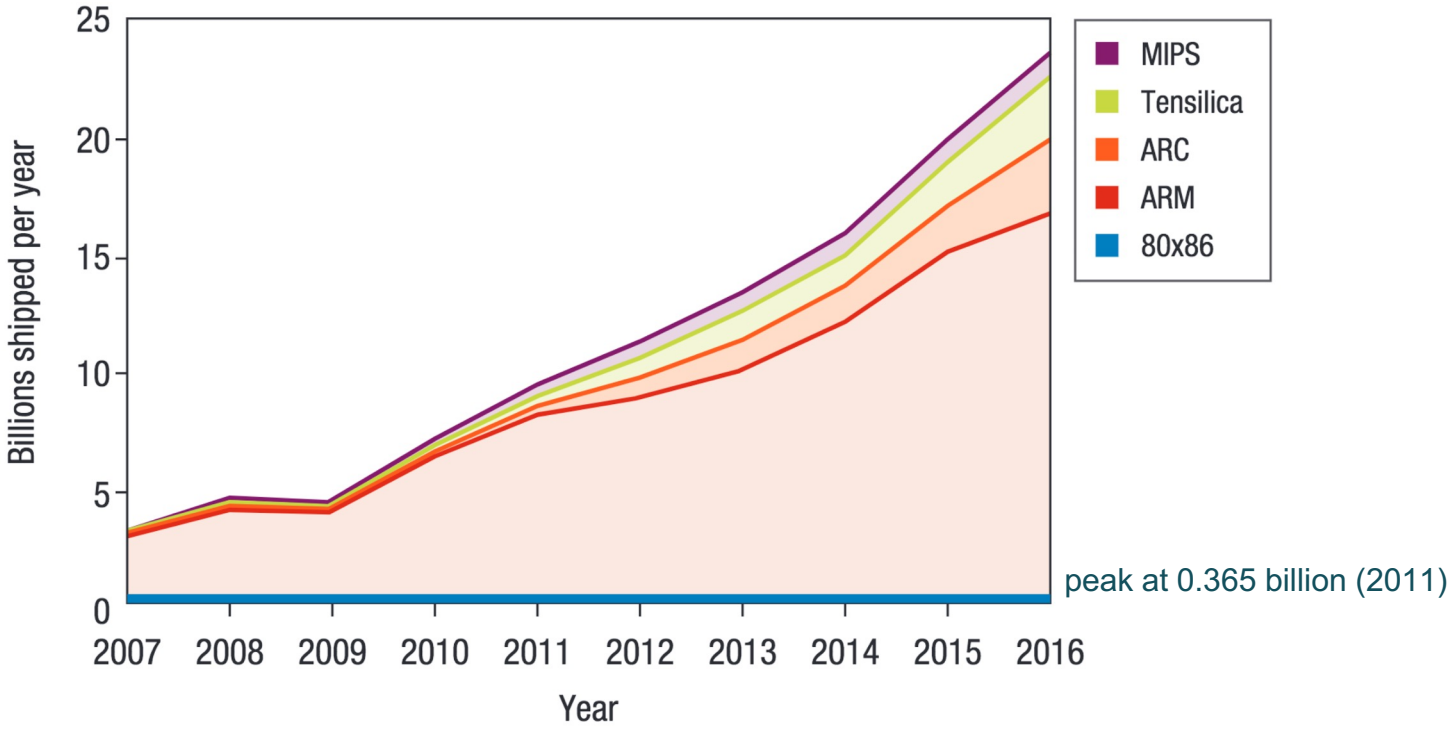
https://linexplore.com/arm-the-revolution/

- *Intel (CISC) VS*
- *Apple (RISC) performances*



https://linexplore.com/arm-the-revolution/

Introduction: ARM *(Advanced RISC Machine, RISC = reduced instruction set computer)*



From D. Patterson, "Reduced Instruction Set Computers Then and Now," in Computer, vol. 50, no. 12, pp. 10-12, December 2017.

Introduction: ARM *(Advanced RISC Machine, RISC = reduced instruction set computer)*

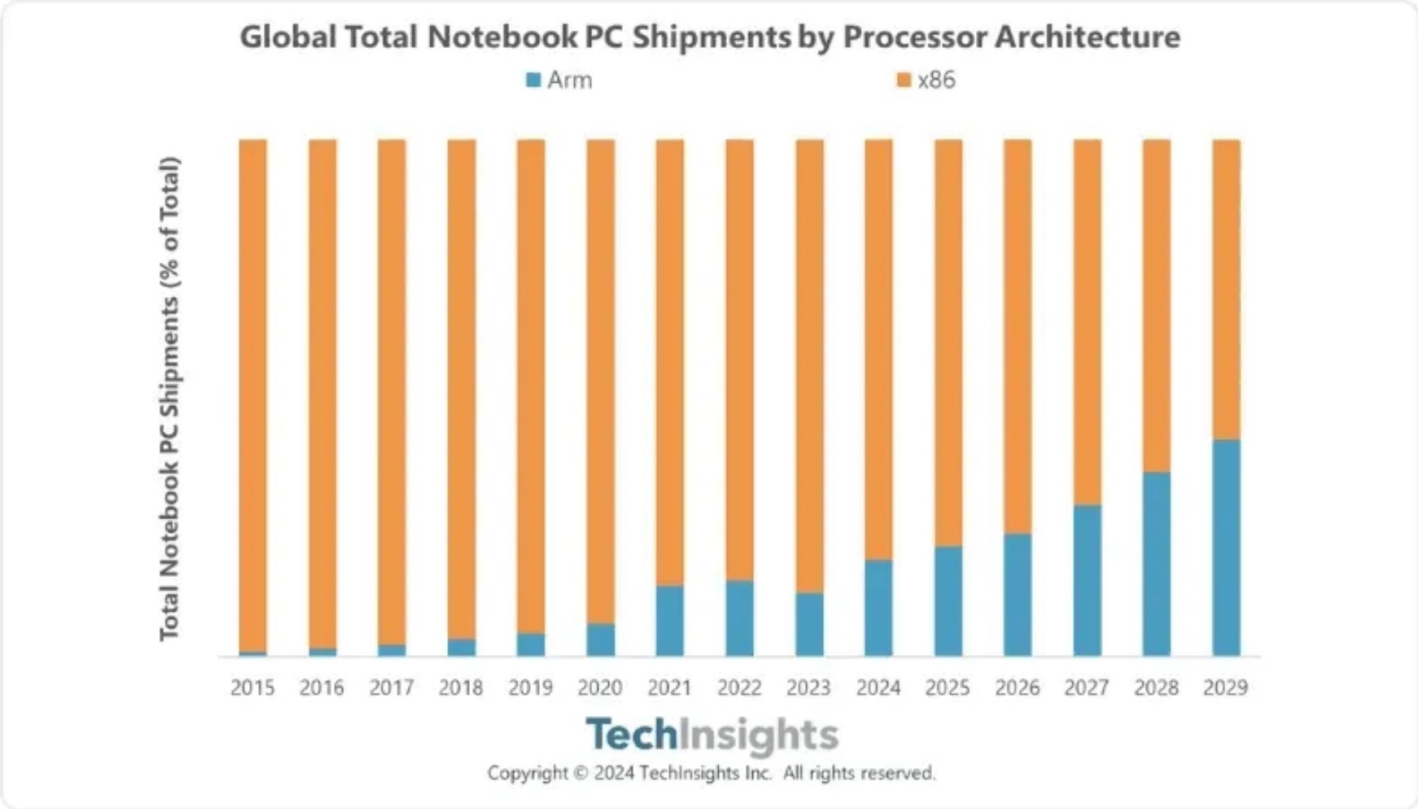


Figure 1 Global Notebook PC Arm vs x86 Forecast 2015-2029, October 2024. Source TechInsights.png

Introduction

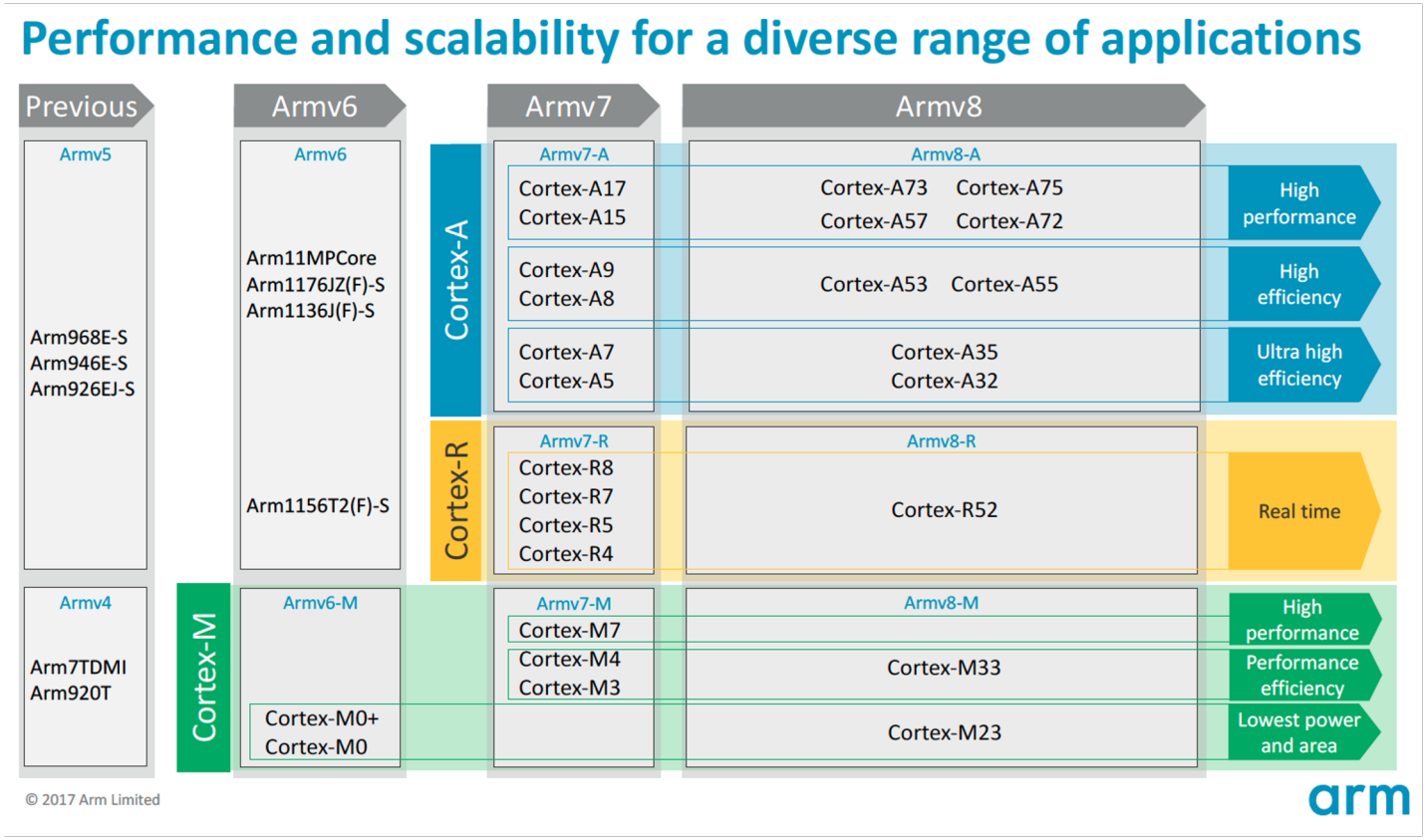
(from sirinsoftware.com *web site)*

A: Smartphones and other portable devices

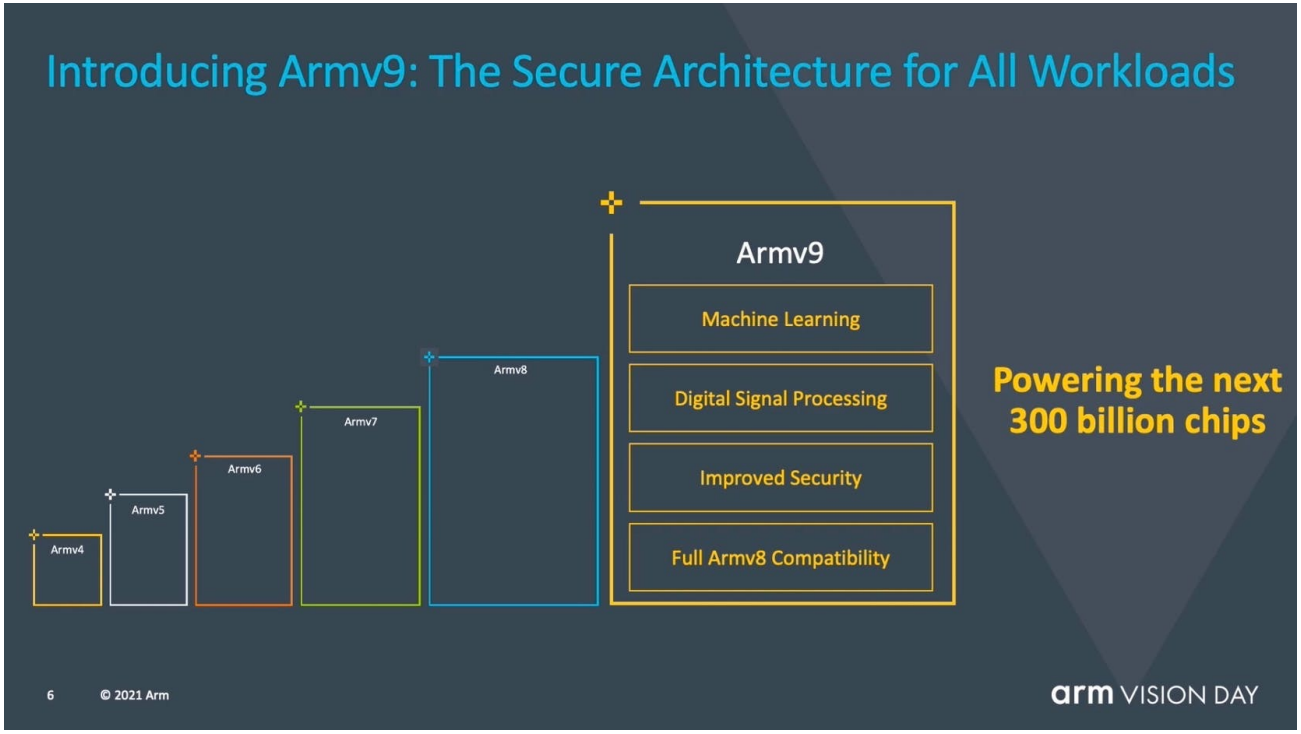
R: real-time solutions: controllers, networking equipment, media players, and other similar devices.

M: microcontrollers

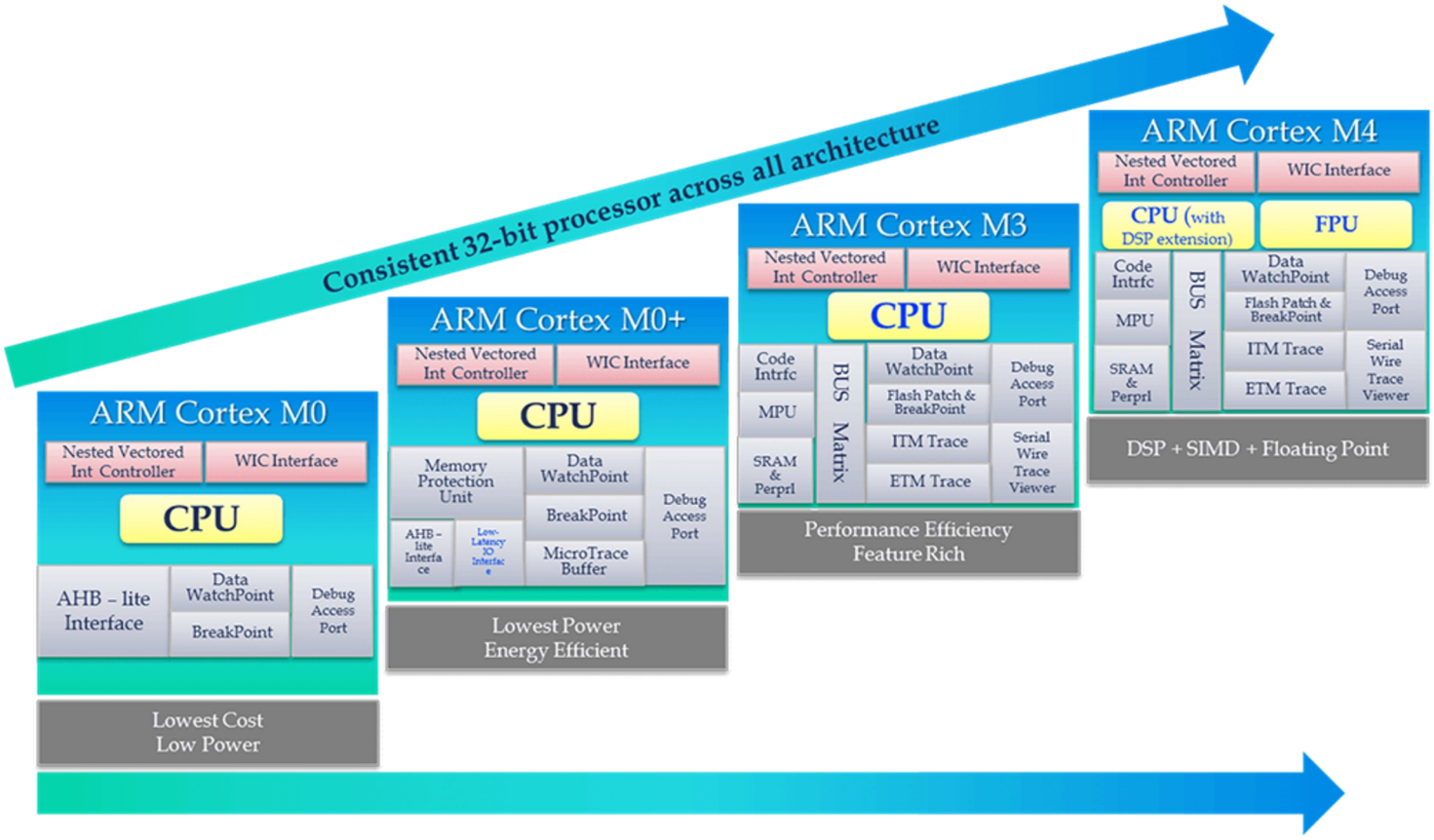
Introduction *(from* arm.com *web site)*



Introduction *(from* arm.com *web site)*



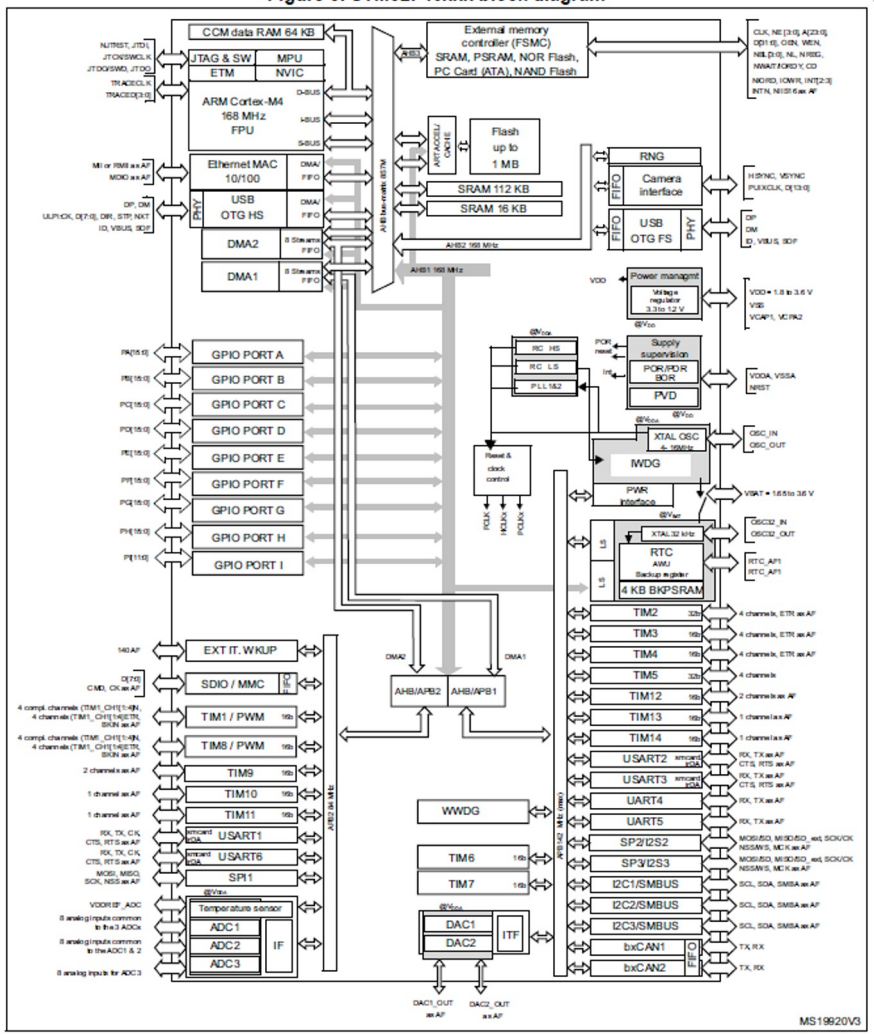
Introduction: cortex M family *(from* arm.com *web site)*



Structure **STM32F104** Microcontôleur de STMicroelectronics qui utilise ARM Cortex M4

"trois ténors européens (des semi-conducteurs) Infineon Technologies (12,7 milliards de dollars en 2022), STMicroelectronics (12,6 milliards de dollars) et NXP (10,8 milliards de dollars)" usinenouvelle.com

Figure 5. STM32F40xxx block diagram



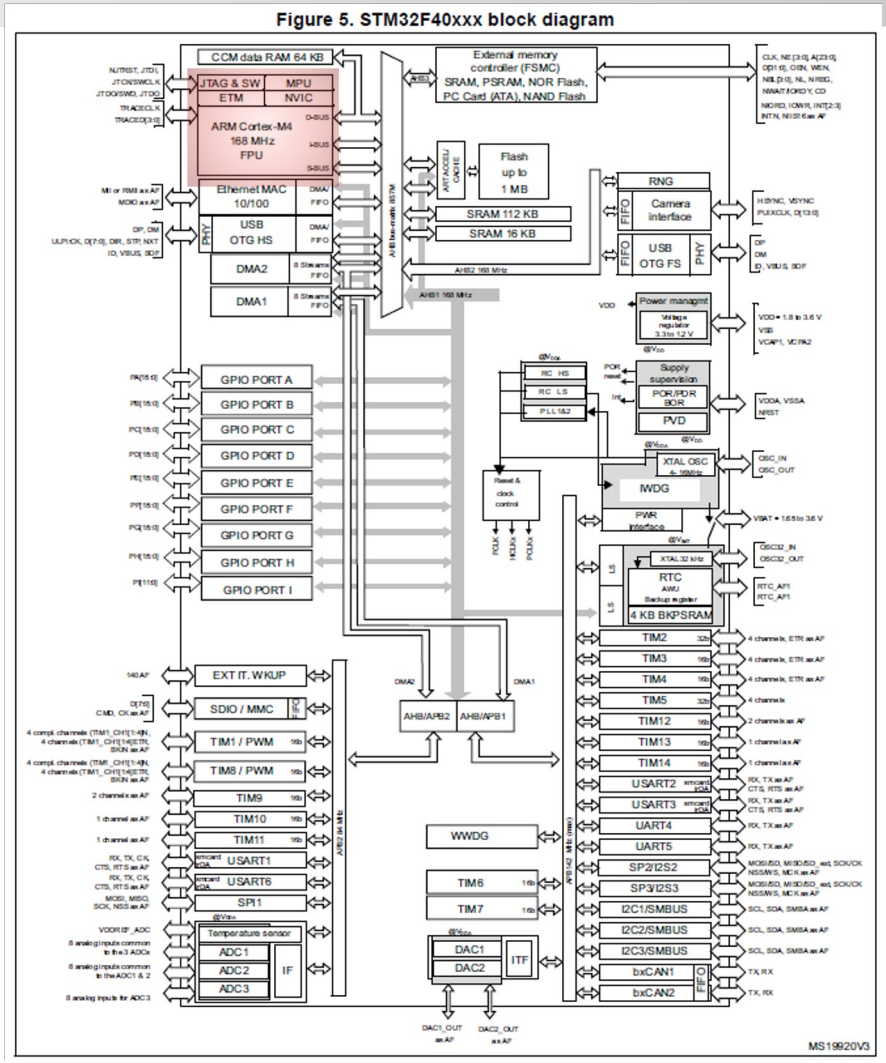
STM32F104 datasheet

Structure

Coeur du microcontroleur: le CPU

STM32F104 datasheet

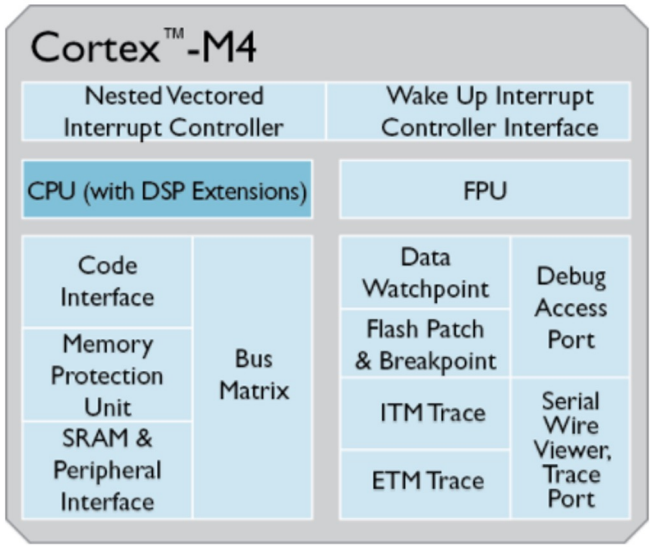
Figure 5. STM32F40xxx block diagram



Cortex M-4

Coeur du microcontrôleur

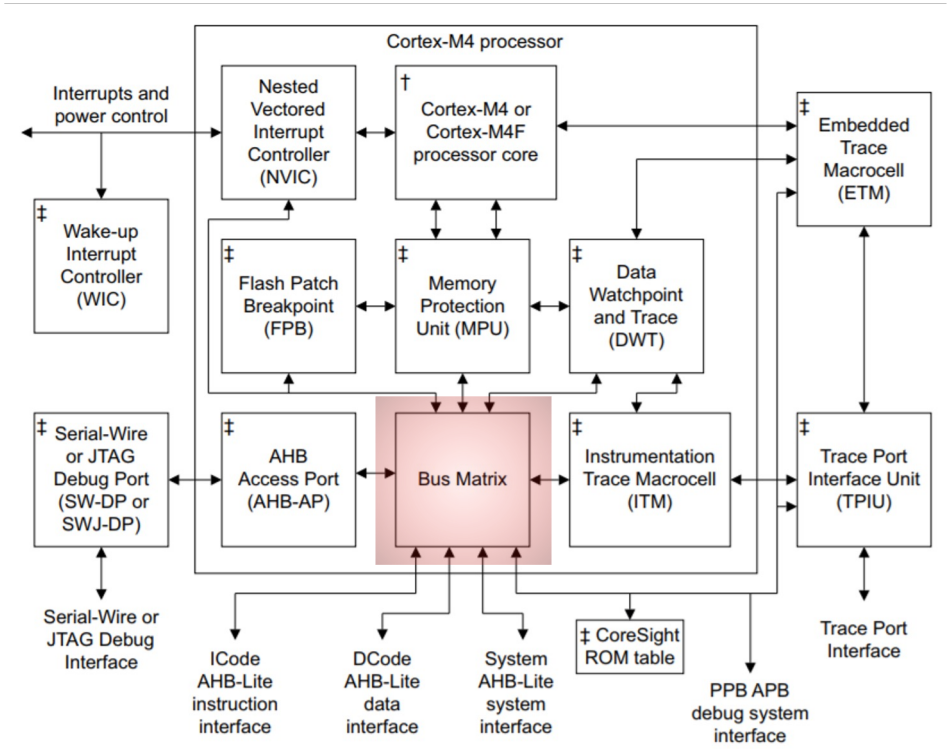
avec “DSP extensions”!



https://community.arm.com/processors/b/blog/posts/armv6-m-vs-armv7-m---unpacking-the-microcontrollers

Cortex M-4

Coeur du microcontrôleur



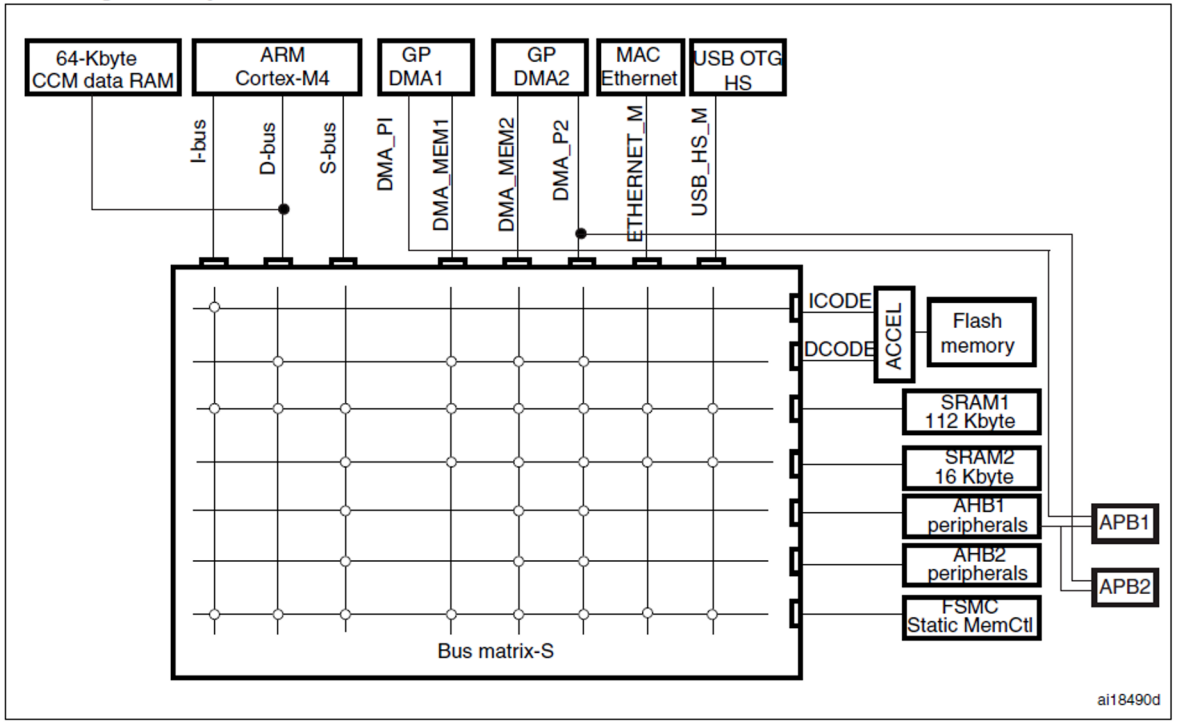
Cortex-M4 Technical Reference Manual

STM32F

Bus Matrix

Interface
Between
Cortex M4, memory,
peripherals etc.

Figure 1. System architecture for STM32F405xx/07xx and STM32F415xx/17xx devices



Cortex-M4
Technical Reference
Manual

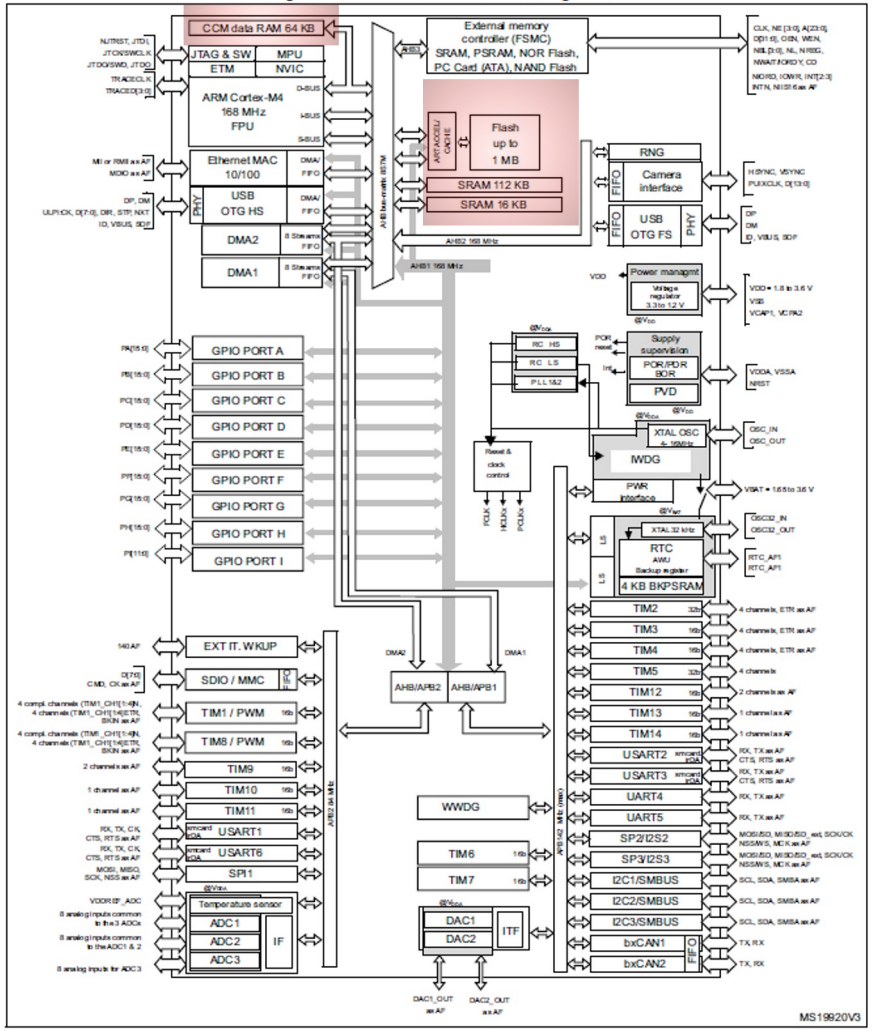
I-bus: instruction bus
D-bus: Data bus
S-bus: System bus
USB OTG: On-The-Go
FSMC: Flexible Static Memory Controller.

Structure

1 MB Flash +
192 KBytes de
RAM/SRAM

STM32F104
datasheet

Figure 5. STM32F40xxx block diagram

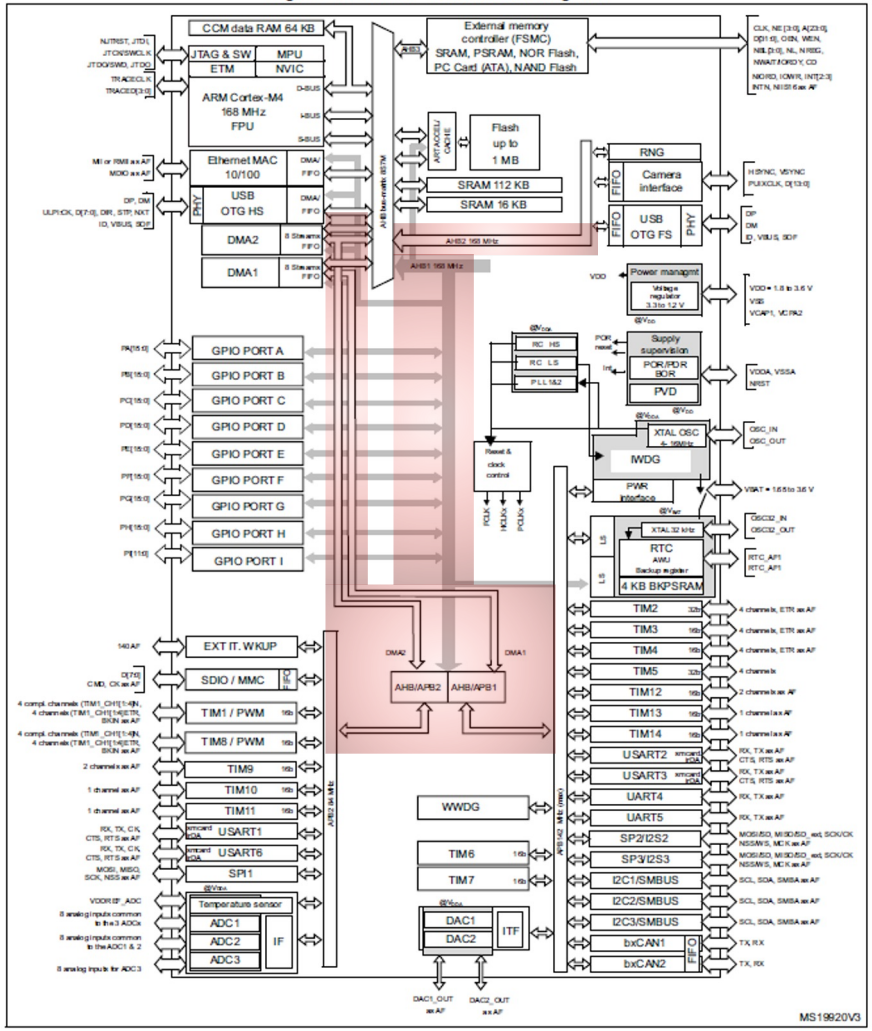


Structure

Deux bus de communications avec les périphériques
＋ DMA

STM32F104 datasheet

Figure 5. STM32F40xxx block diagram

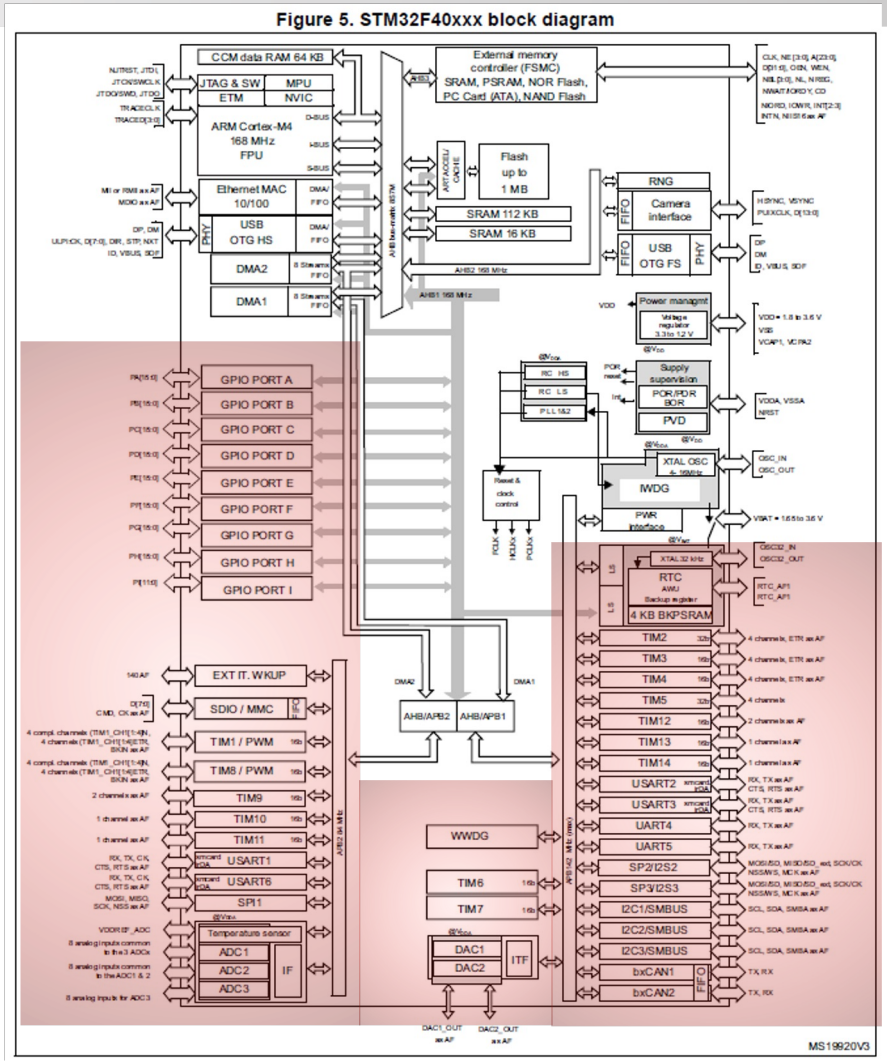


Structure

Une multitude de périphériques

STM32F104 datasheet

Figure 5. STM32F40xxx block diagram



Structure

Pinning: partage des fonctions

Table 7. STM32F40xxx pin and ball definitions (continued)

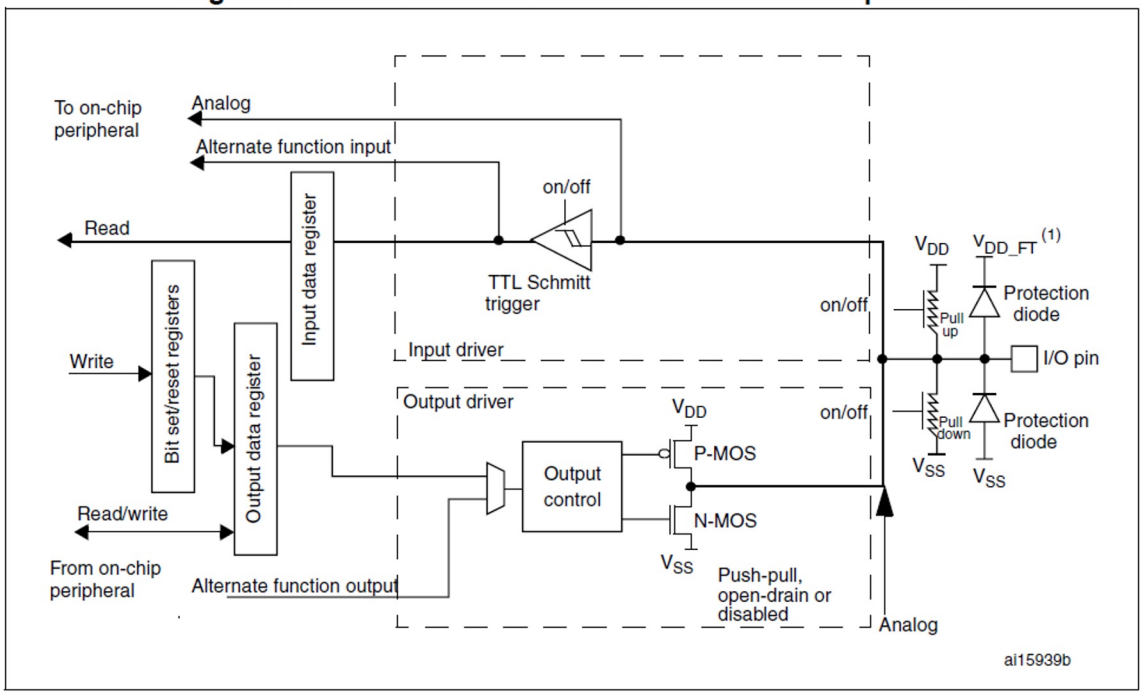
Pin number						Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP64	WLCSP90	LQFP100	LQFP144	UFBGA176	LQFP176						
33	J3	51	73	P12	92	PB12	I/O	FT	-	SPI2_NSS / I2S2_WS / I2C2_SMBA/ USART3_CK/ TIM1_BKIN / CAN2_RX / OTG_HS_ULPI_D5/ ETH_RMII_TXD0 / ETH_MII_TXD0/ OTG_HS_ID/ EVENTOUT	-
34	J1	52	74	P13	93	PB13	I/O	FT	-	SPI2_SCK / I2S2_CK / USART3_CTS/ TIM1_CH1N /CAN2_TX / OTG_HS_ULPI_D6 / ETH_RMII_TXD1 / ETH_MII_TXD1/ EVENTOUT	OTG_HS_VBUS
35	J2	53	75	R14	94	PB14	I/O	FT	-	SPI2_MISO/ TIM1_CH2N / TIM12_CH1 / OTG_HS_DM/ USART3_RTS / TIM8_CH2N/I2S2ext_SD/ EVENTOUT	-
36	H1	54	76	R15	95	PB15	I/O	FT	-	SPI2_MOSI / I2S2_SD/ TIM1_CH3N / TIM8_CH3N / TIM12_CH2 / OTG_HS_DP/ EVENTOUT	RTC_REFIN

STM32F104
Reference manual

Structure

Pinning: partage des fonctions

Figure 25. Basic structure of a five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

STM32F104
Reference manual

Structure

Pinning: Configuration

4 registres de configuration

Table 35. Port bit configuration table

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

Structure

Pinning:
Configuration

1) Mode (MODER)

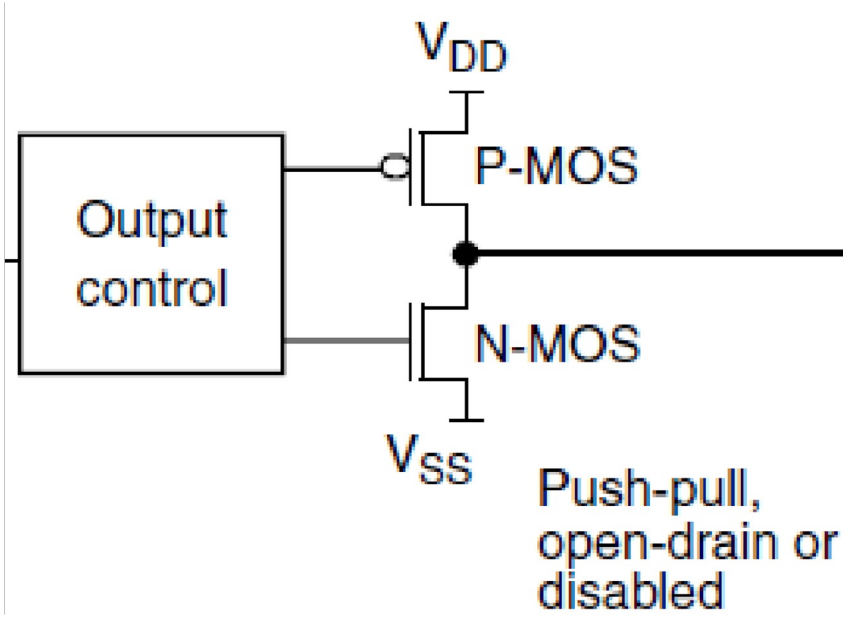
La pin sera configurée en :

- Entrée (Input)
- Sortie (Output)
- Fonctions alternées (Alternate function) (I2C, UART, PWM, etc.)
- Analogique (Analog)

Structure

Pinning:
Configuration

1) Mode (MODER)
2) **Type d'"output" (OTYPER)**

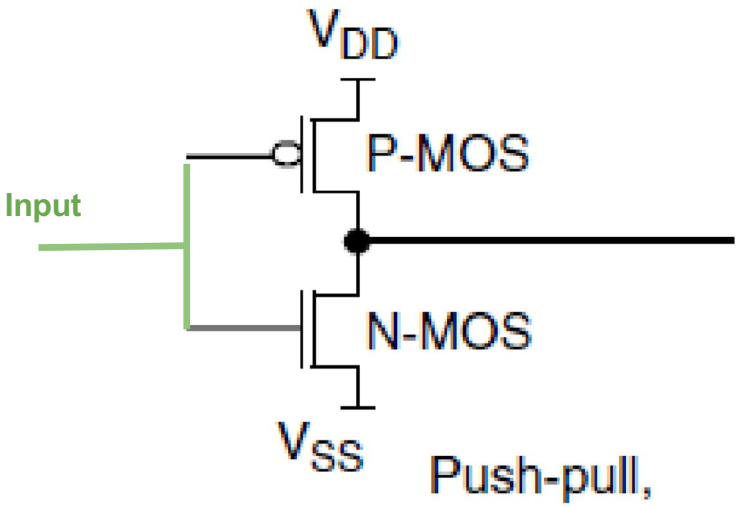


Structure

Pinning:

Configuration push-pull

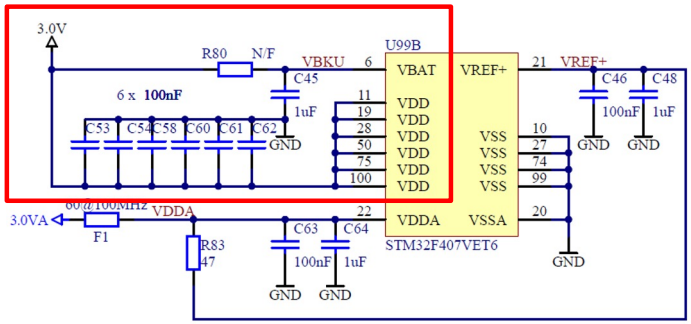
1) Mode (MODER)
2) **Type d'output (OTYPER)**

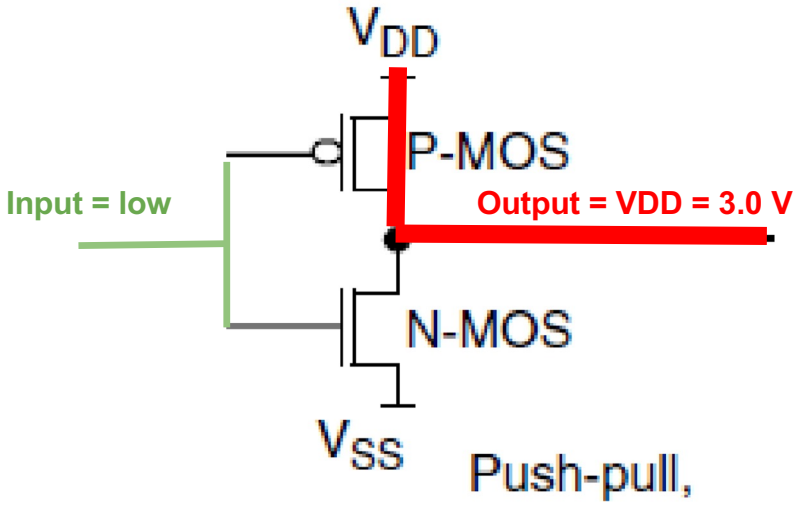


Structure

Pinning:
Configuration push-pull

1) Mode (MODER)
2) Type d'output (OTYPER)





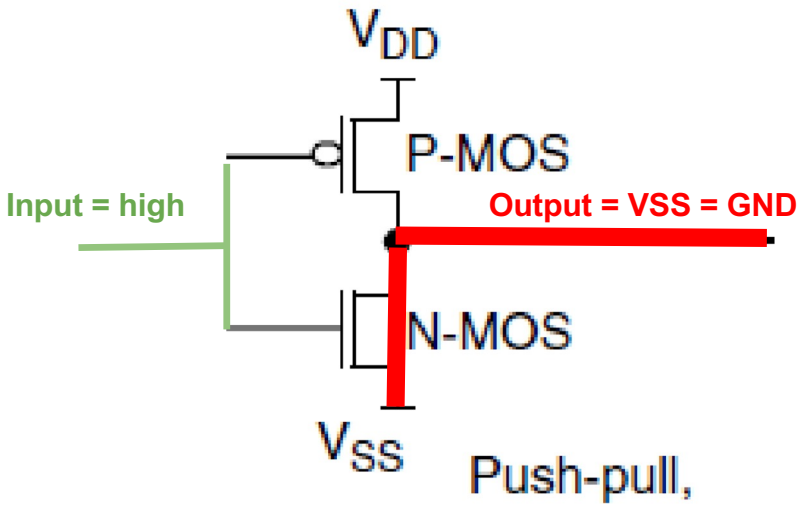
Alimentation du processeur (VDD, VSS)
Cf Dossier électronique de l'e-puck2

47

Structure

Pinning:
Configuration push-pull

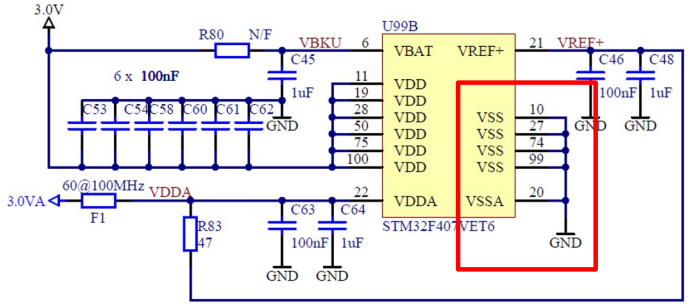
1) Mode (MODER)
2) Type d'output (OTYPER)



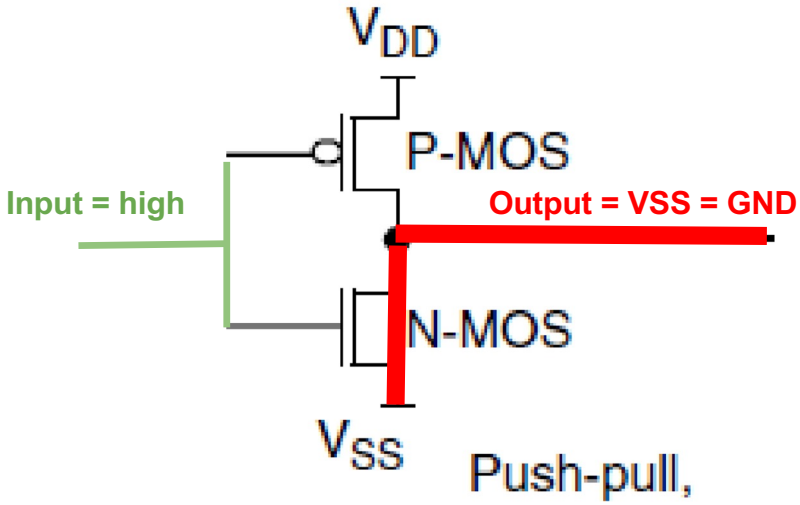
Structure

Pinning:
Configuration push-pull

1) Mode (MODER)
2) Type d'output (OTYPER)



Alimentation du processeur (VDD, VSS)
Cf Dossier électronique de l'e-puck2



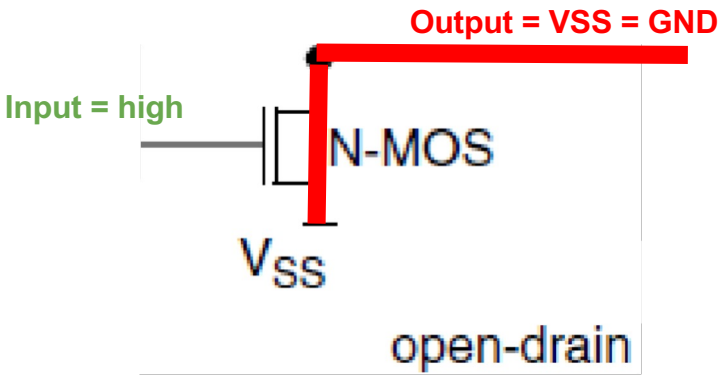
49

Structure

Pinning:

Configuration open-drain

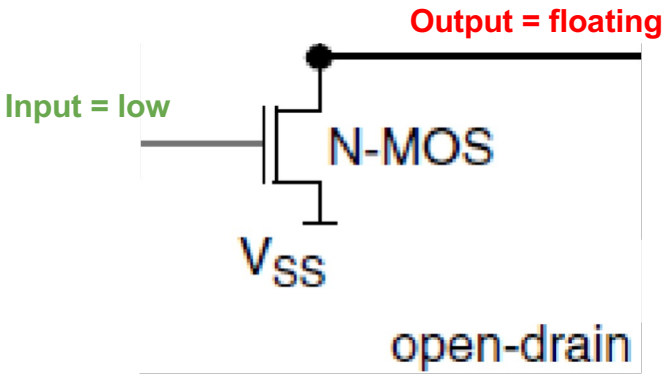
1) Mode (MODER)
2) **Type d'output (OTYPER)**



Structure

Pinning:
Configuration open-drain

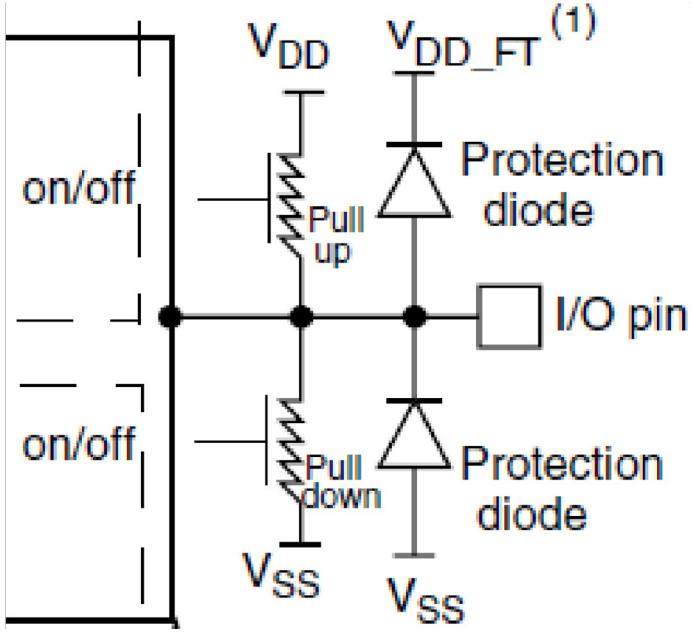
1) Mode (MODER)
2) **Type d'output (OTYPER)**



Structure

Pinning:
Configuration

1) Mode (MODER)
2) Type d'output (OTYPER)
3) **Type de "pull" (PUPDR)**

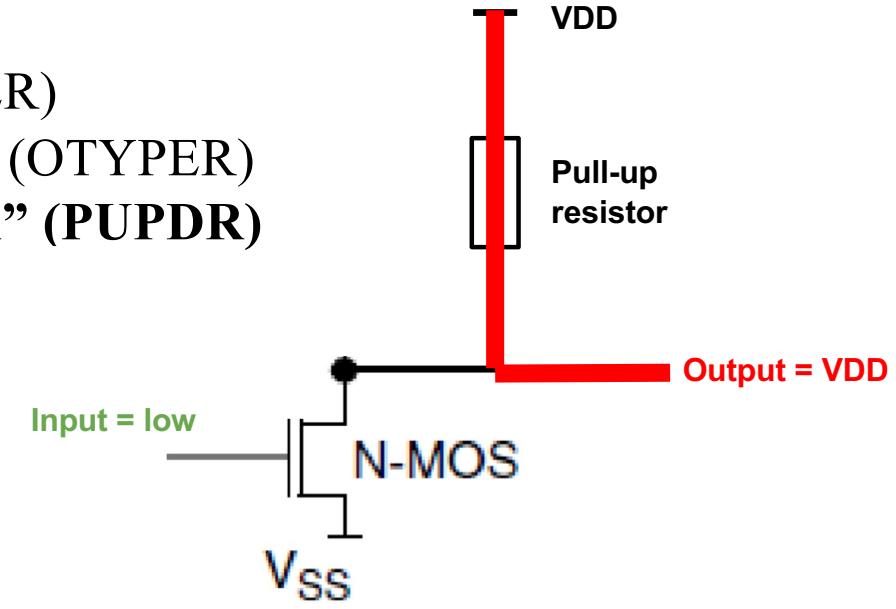


PUPDR = PullUpPullDownRegister

Structure

Pinning:
Configuration

1) Mode (MODER)
2) Type d'output (OTYPER)
3) **Type de "pull" (PUPDR)**

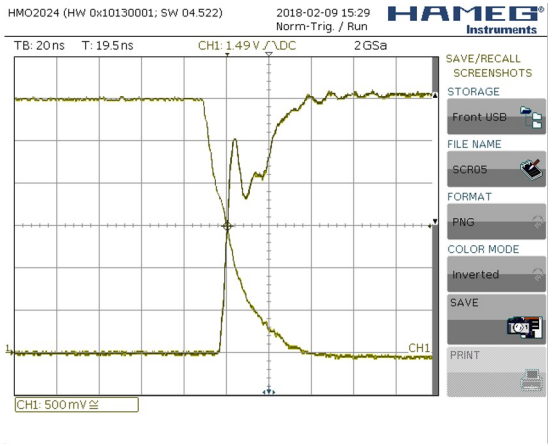


open-drain + **pull-up**

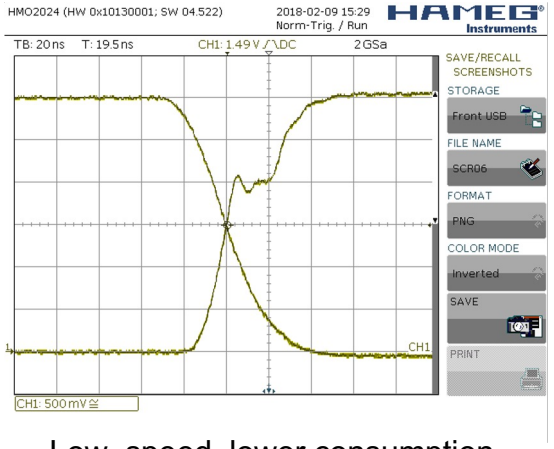
Structure

Pinning:
Configuration

1) Mode (MODER)
2) Type d'output (OTYPER)
3) Type de "pull" (PUPDR)
4) **Vitesse de sortie (OSPEEDR)**



High-speed, higher consumption

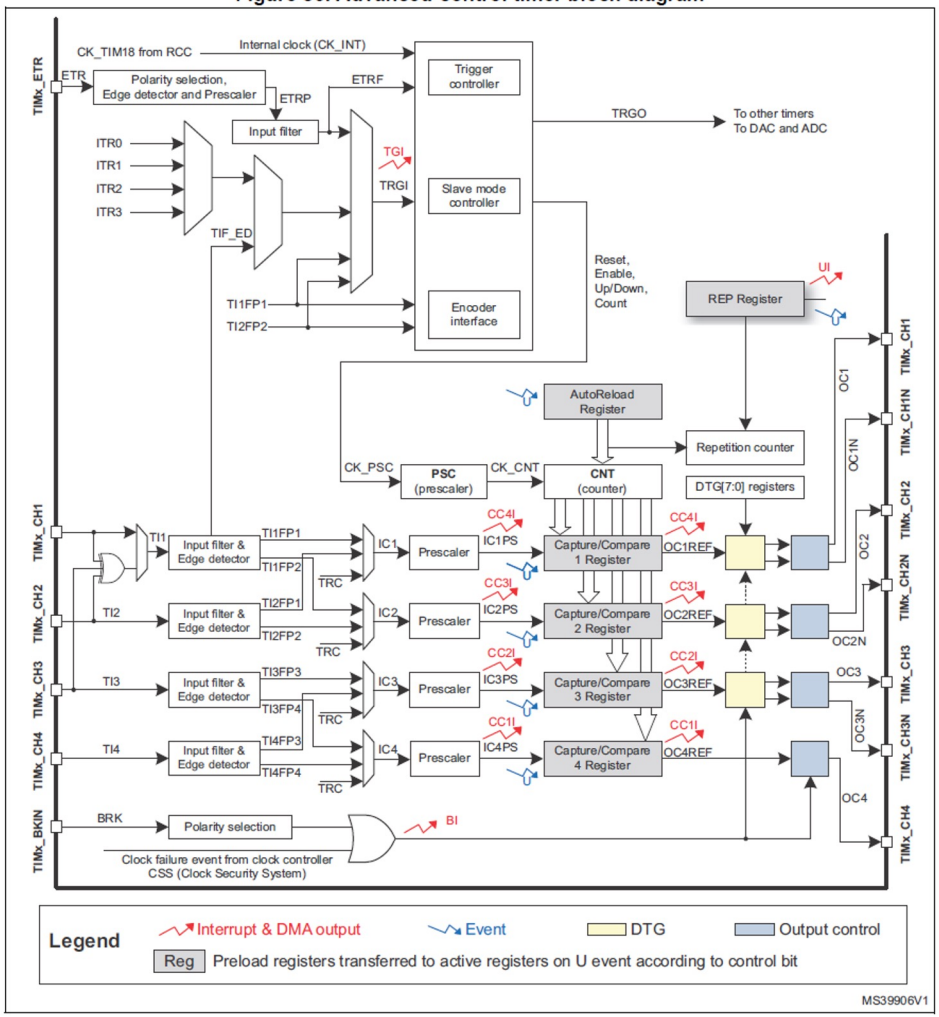


Low_speed, lower consumption

Structure

Timer
(+input capture,
output compare)

Figure 86. Advanced-control timer block diagram



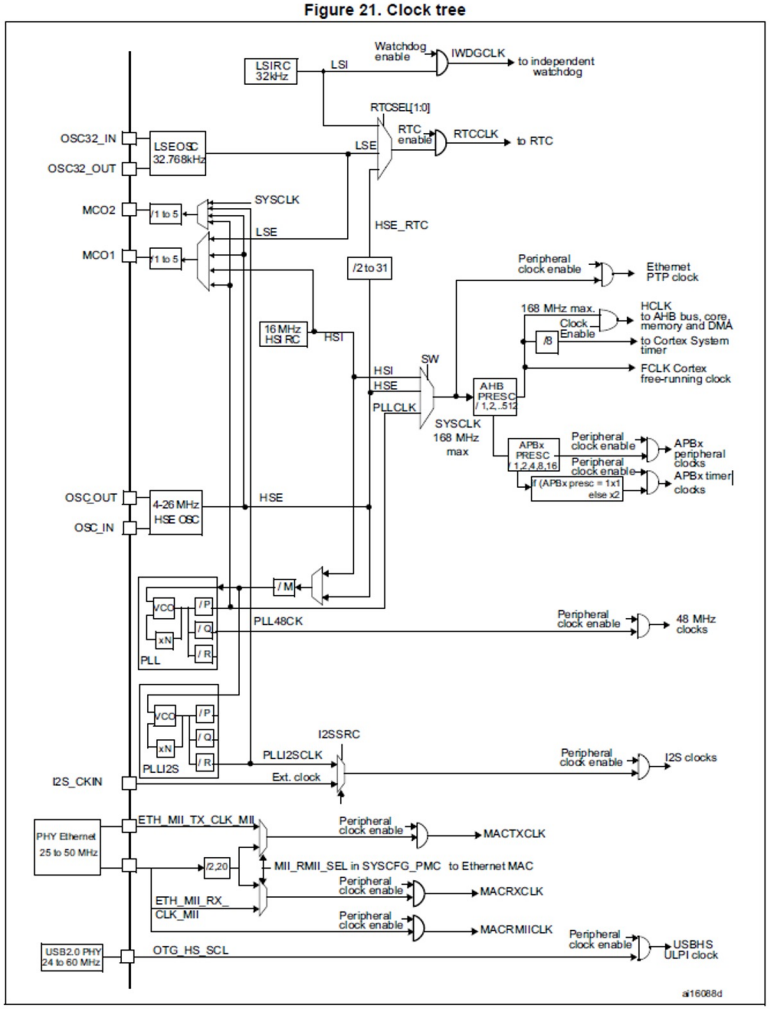
STM
Reference Manual

Structure

Clock

STM Reference Manual

Figure 21. Clock tree

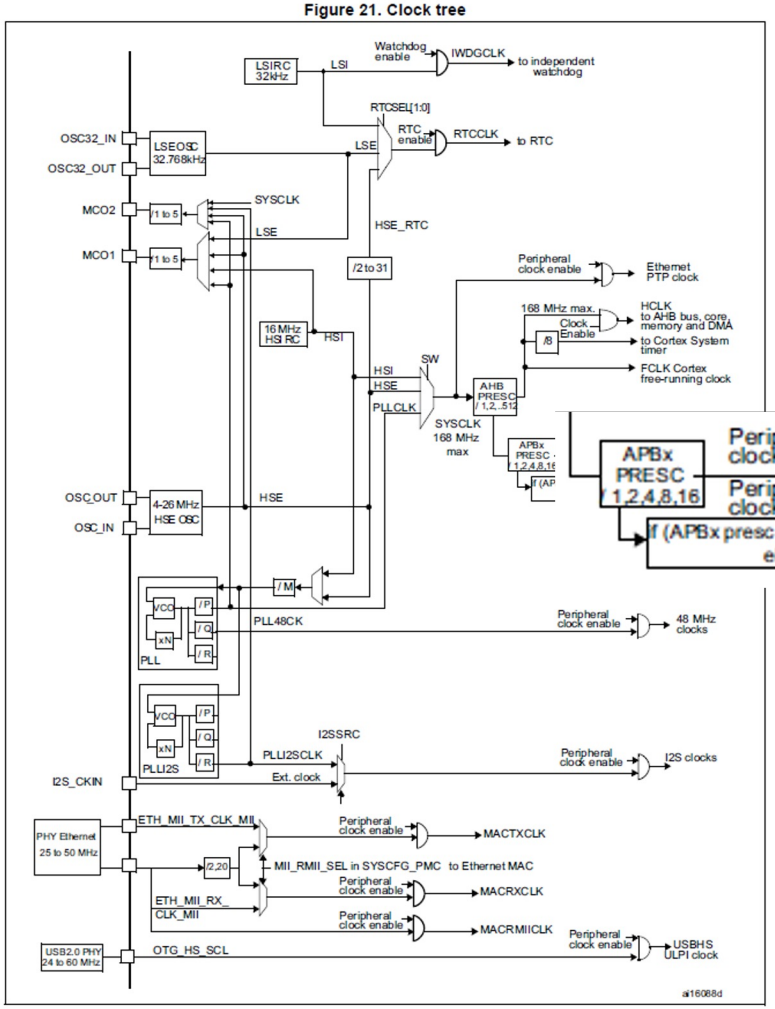


1. For full details about the internal and external clock source characteristics, refer to the Electrical characteristics section in the device datasheet.

Structure

Clock

Figure 21. Clock tree

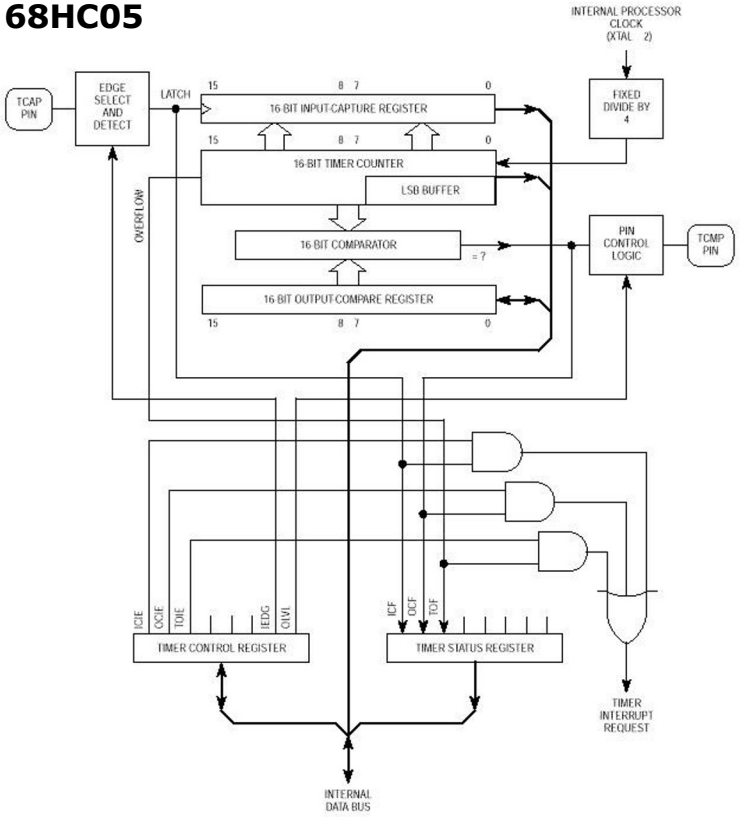


1. For full details about the internal and external clock source characteristics, refer to the Electrical characteristics section in the device datasheet.

STM Reference Manual

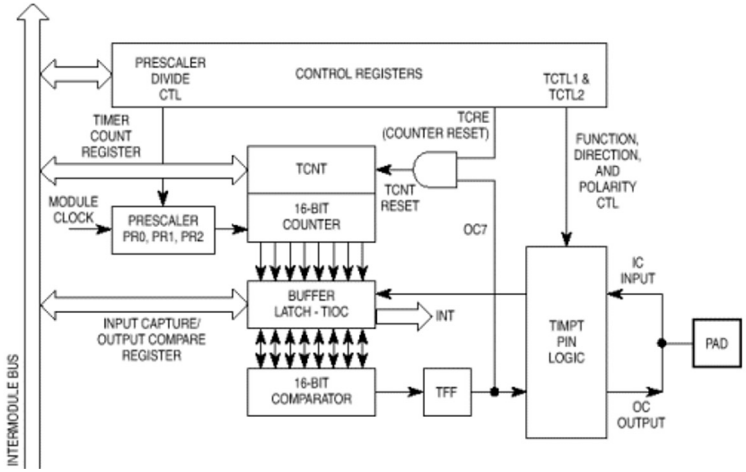
Exemples d'autres structures

68HC05



http://rab.ict.pwr.wroc.pl/~mw/mcu/overview/images/hc05/timer05.jpg

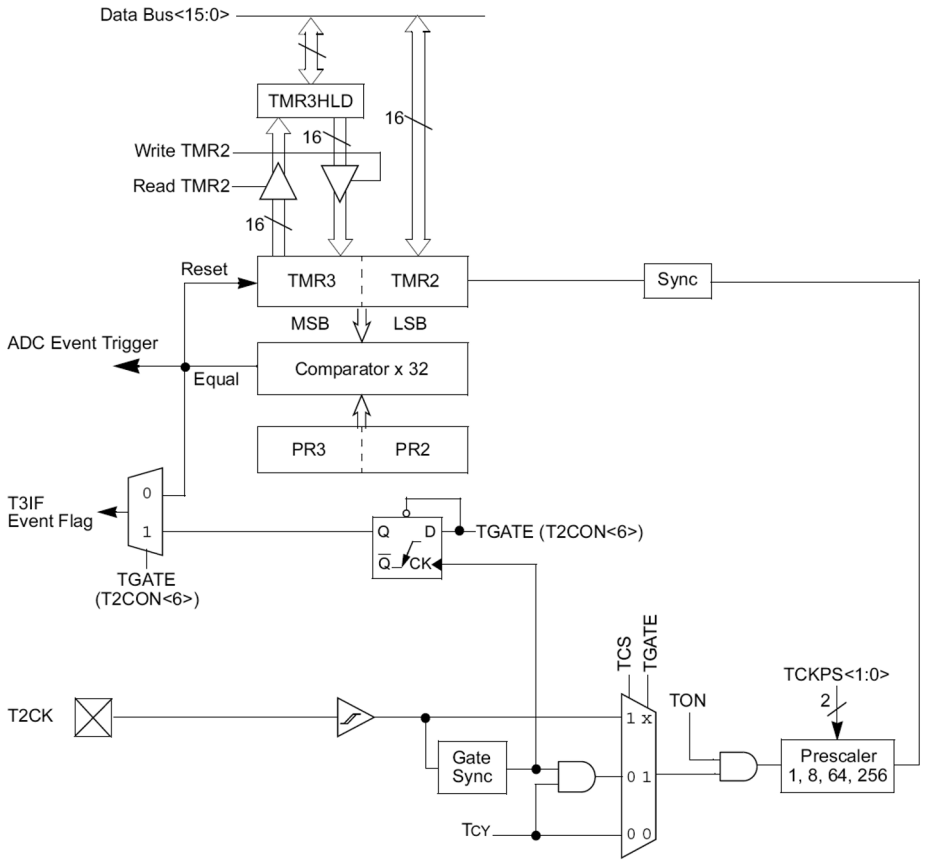
68HC12



http://www.seattlerobotics.org/encoder/nov97/tim_block.gif

Exemple d'autre structures

Timer (2 x 32bit)



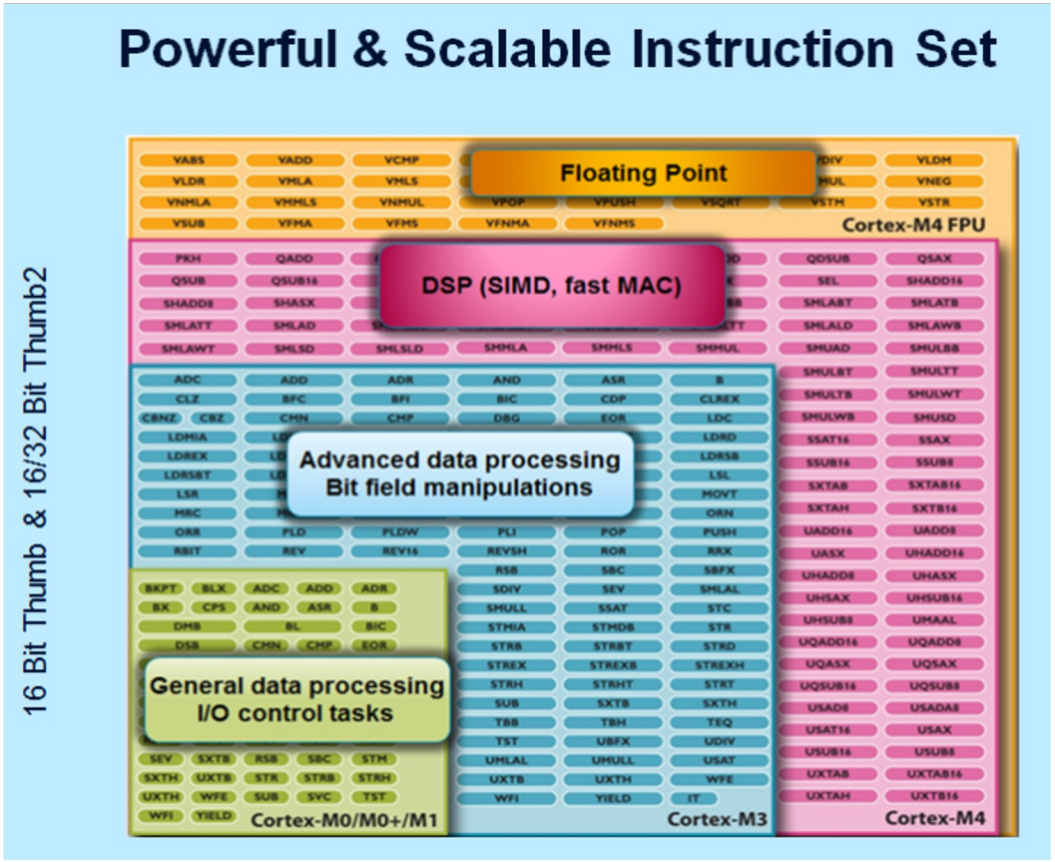
dsPIC datasheet

Programmation

Les processeurs ARM ont un set d'instructions assez important pour un RISC, dont la grande majorité est exécutée en un cycle

Les instructions sont optimisées pour effectuer du calcul rapide

Instructions



https://community.arm.com/processors/b/blog/posts/armv6-m-vs-armv7-m---unpacking-the-microcontrollers

Programmation

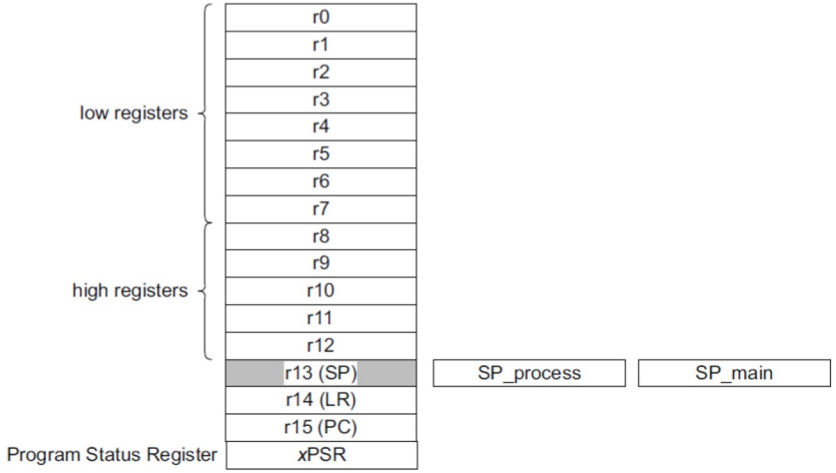
Modèle de programmation

Processor core register summary

The processor has the following 32-bit registers:

- 13 general-purpose registers, r0-r12
- *Stack Pointer* (SP) alias of banked registers, SP_process and SP_main
- *Link Register* (LR), r14
- *Program Counter* (PC), r15
- Special-purpose *Program Status Registers*, (*x*PSR).

Figure 3-3 shows the processor register set.



Cortex-M4 Technical Reference Manual

Programmation

Modèle
de
programmation

Cortex-M4 Technical Reference Manual

The processor core registers are:

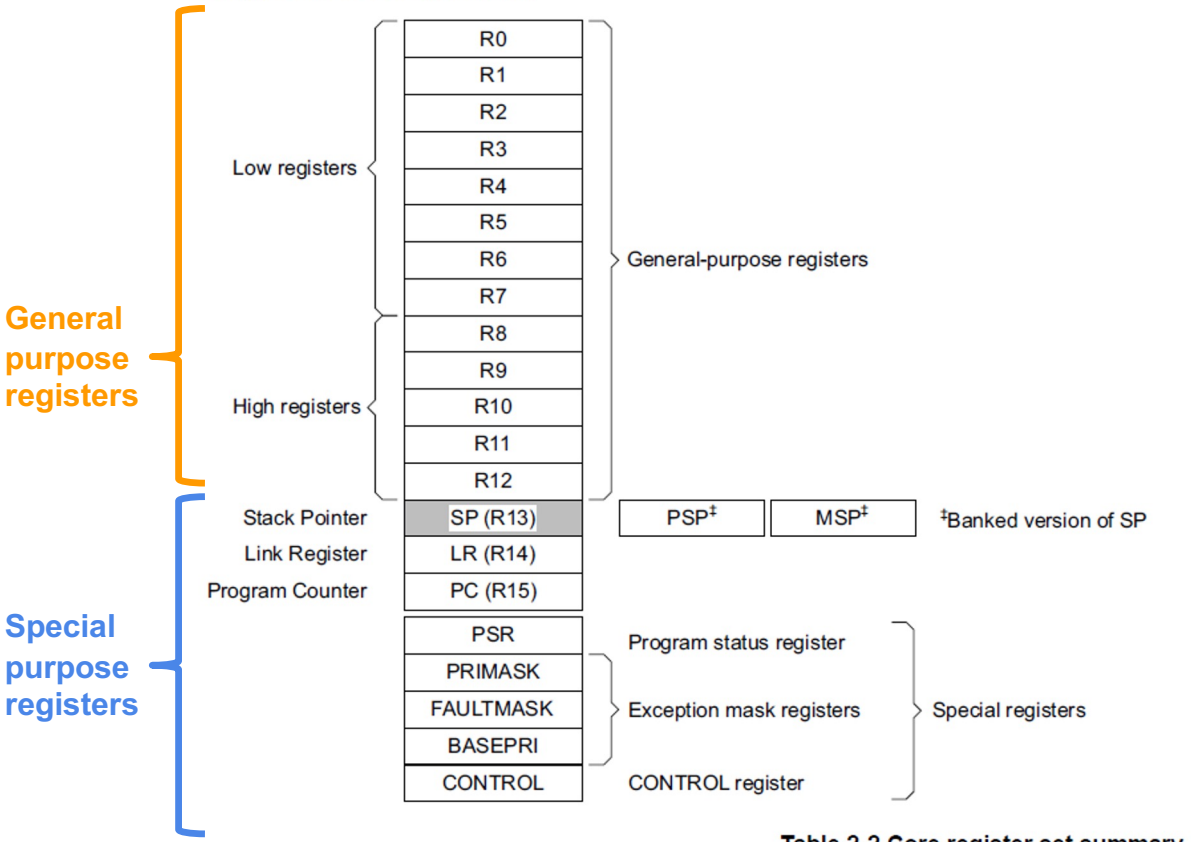


Table 2-2 Core register set summary

Programmation

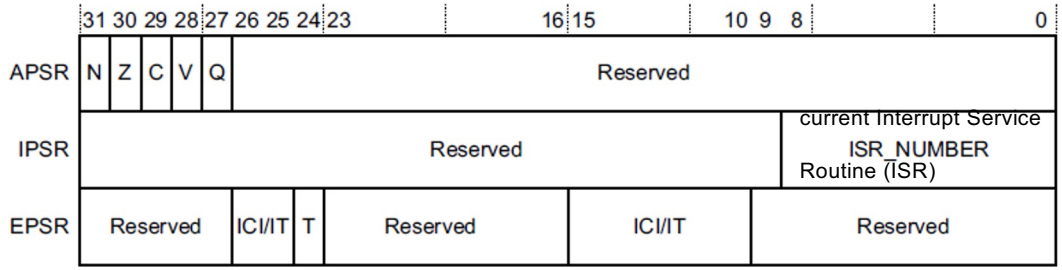
Modèle de programmation

Program Status Register

The *Program Status Register* (PSR) combines:

- *Application Program Status Register* (APSR)
- *Interrupt Program Status Register* (IPSR)
- *Execution Program Status Register* (EPSR).

These registers are mutually exclusive bitfields in the 32-bit PSR. The bit assignments are:



Cortex-M4 Technical Reference Manual

Programmation

Memory map

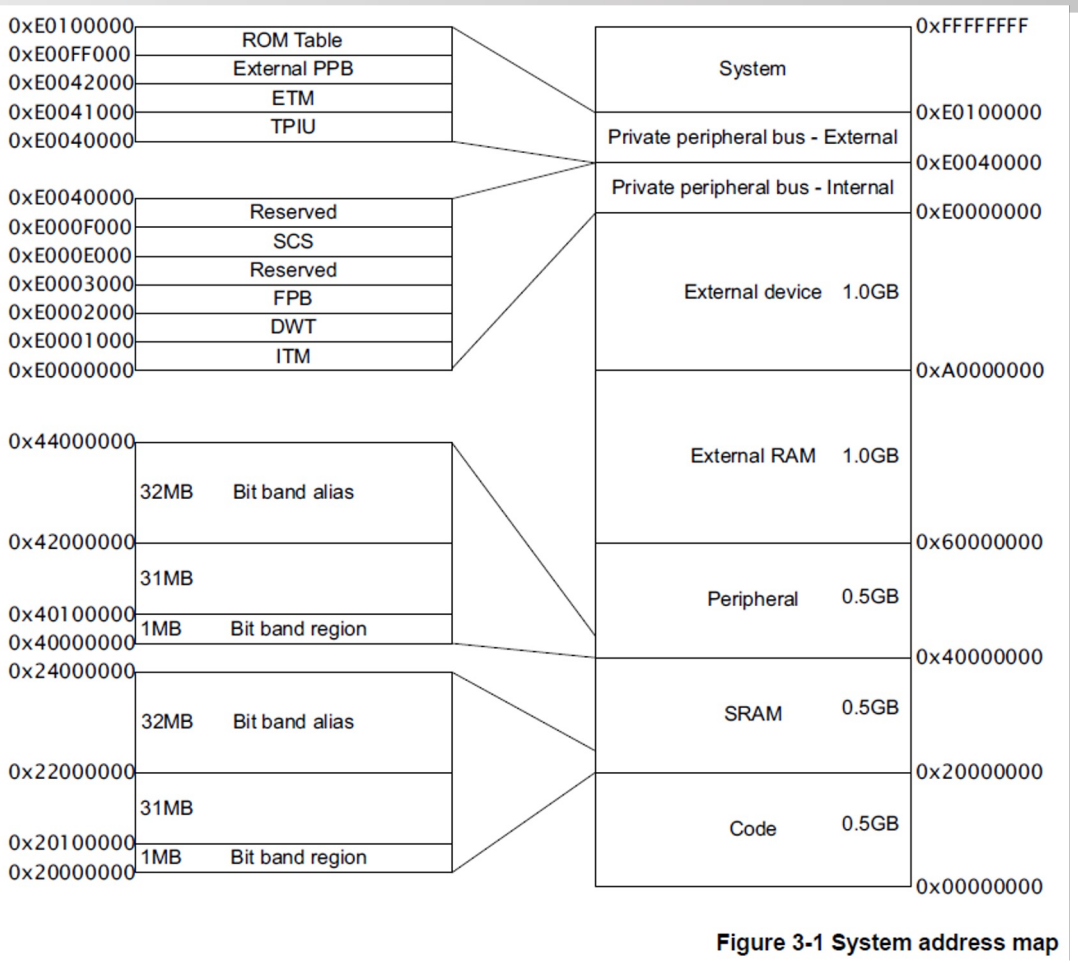


Figure 3-1 System address map

Cortex M4
Technical Reference Manual (p. 39)

Programmation

Instructions:
Move, Add

Cortex-M4 Technical Reference Manual

Table 3-1 Cortex-M4 instruction set summary

Operation	Description	Assembler	Cycles
Move	Register	`MOV Rd, <op2>`	1
	16-bit immediate	`MOVW Rd, #<imm>`	1
	Immediate into top	`MOVT Rd, #<imm>`	1
	To PC	`MOV PC, Rm`	1 + P
Add	Add	`ADD Rd, Rn, <op2>`	1
	Add to PC	`ADD PC, PC, Rm`	1 + P
	Add with carry	`ADC Rd, Rn, <op2>`	1
	Form address	`ADR Rd, <label>`	1

Un "cycle" d'horloge du processeur = un flanc montant et descendant de la clock

3.3.3 Flexible second operand

Many general data processing instructions have a flexible second operand. This is shown as *Operand2* in the descriptions of the syntax of each instruction.

Operand2 can be a:

- *Constant*
- *Register with optional shift* on page 3-13

Instructions:
Math…

Operation	Description	Assembler	Cycles
Subtract	Subtract	`SUB Rd, Rn, <op2>`	1
	Subtract with borrow	`SBC Rd, Rn, <op2>`	1
	Reverse	`RSB Rd, Rn, <op2>`	1
Multiply	Multiply	`MUL Rd, Rn, Rm`	1
	Multiply accumulate	`MLA Rd, Rn, Rm`	2
	Multiply subtract	`MLS Rd, Rn, Rm`	2
	Long signed	`SMULL RdLo, RdHi, Rn, Rm`	1
	Long unsigned	`UMULL RdLo, RdHi, Rn, Rm`	1
	Long signed accumulate	`SMLAL RdLo, RdHi, Rn, Rm`	1
	Long unsigned accumulate	`UMLAL RdLo, RdHi, Rn, Rm`	1
Divide	Signed	`SDIV Rd, Rn, Rm`	2 to 12[a]
	Unsigned	`UDIV Rd, Rn, Rm`	2 to 12[a]
Saturate	Signed	`SSAT Rd, #<imm>, <op2>`	1
	Unsigned	`USAT Rd, #<imm>, <op2>`	1

Cortex-M4 Technical Reference Manual

Instructions: Math…(2)

Compare	Compare	`CMP Rn, <op2>`	1
	Negative	`CMN Rn, <op2>`	1
Logical	AND	`AND Rd, Rn, <op2>`	1
	Exclusive OR	`EOR Rd, Rn, <op2>`	1
	OR	`ORR Rd, Rn, <op2>`	1
	OR NOT	`ORN Rd, Rn, <op2>`	1
	Bit clear	`BIC Rd, Rn, <op2>`	1
	Move NOT	`MVN Rd, <op2>`	1
	AND test	`TST Rn, <op2>`	1
	Exclusive OR test	`TEQ Rn, <op1>`	
Shift	Logical shift left	`LSL Rd, Rn, #<imm>`	1
	Logical shift left	`LSL Rd, Rn, Rs`	1
	Logical shift right	`LSR Rd, Rn, #<imm>`	1
	Logical shift right	`LSR Rd, Rn, Rs`	1
	Arithmetic shift right	`ASR Rd, Rn, #<imm>`	1
	Arithmetic shift right	`ASR Rd, Rn, Rs`	1
Rotate	Rotate right	`ROR Rd, Rn, #<imm>`	1
	Rotate right	`ROR Rd, Rn, Rs`	1
	With extension	`RRX Rd, Rn`	1

Cortex-M4 Technical Reference Manual

Instructions:

Operation	Description	Assembler	Cycles
Count	Leading zeroes	`CLZ Rd, Rn`	1
Load	Word	`LDR Rd, [Rn, <op2>]`	2^b
	To PC	`LDR PC, [Rn, <op2>]`	$2^b + P$
	Halfword	`LDRH Rd, [Rn, <op2>]`	2^b
	Byte	`LDRB Rd, [Rn, <op2>]`	2^b
	Signed halfword	`LDRSH Rd, [Rn, <op2>]`	2^b
	Signed byte	`LDRSB Rd, [Rn, <op2>]`	2^b
	User word	`LDRT Rd, [Rn, #<imm>]`	2^b
	User halfword	`LDRHT Rd, [Rn, #<imm>]`	2^b
	User byte	`LDRBT Rd, [Rn, #<imm>]`	2^b
	User signed halfword	`LDRSHT Rd, [Rn, #<imm>]`	2^b
	User signed byte	`LDRSBT Rd, [Rn, #<imm>]`	2^b
	PC relative	`LDR Rd,[PC, #<imm>]`	2^b
	Doubleword	`LDRD Rd, Rd, [Rn, #<imm>]`	1 + N
	Multiple	`LDM Rn, {<reglist>}`	1 + N
	Multiple including PC	`LDM Rn, {<reglist>, PC}`	1 + N + P

Cortex-M4 Technical Reference Manual

Instructions: Store, push, pull

Store	Word	`STR Rd, [Rn, <op2>]`	2[b]
	Halfword	`STRH Rd, [Rn, <op2>]`	2[b]
	Byte	`STRB Rd, [Rn, <op2>]`	2[b]
	Signed halfword	`STRSH Rd, [Rn, <op2>]`	2[b]
	Signed byte	`STRSB Rd, [Rn, <op2>]`	2[b]
	User word	`STRT Rd, [Rn, #<imm>]`	2[b]
	User halfword	`STRHT Rd, [Rn, #<imm>]`	2[b]
	User byte	`STRBT Rd, [Rn, #<imm>]`	2[b]
	User signed halfword	`STRSHT Rd, [Rn, #<imm>]`	2[b]
	User signed byte	`STRSBT Rd, [Rn, #<imm>]`	2[b]
	Doubleword	`STRD Rd, Rd, [Rn, #<imm>]`	1 + N
	Multiple	`STM Rn, {<reglist>}`	1 + N
Push	Push	`PUSH {<reglist>}`	1 + N
	Push with link register	`PUSH {<reglist>, LR}`	1 + N
Pop	Pop	`POP {<reglist>}`	1 + N
	Pop and return	`POP {<reglist>, PC}`	1 + N + P

Cortex-M4 Technical Reference Manual

Instructions: Semaphore, branch

Operation	Description	Assembler	Cycles
Semaphore	Load exclusive	`LDREX Rd, [Rn, #<imm>]`	2
	Load exclusive half	`LDREXH Rd, [Rn]`	2
	Load exclusive byte	`LDREXB Rd, [Rn]`	2
	Store exclusive	`STREX Rd, Rt, [Rn, #<imm>]`	2
	Store exclusive half	`STREXH Rd, Rt, [Rn]`	2
	Store exclusive byte	`STREXB Rd, Rt, [Rn]`	2
	Clear exclusive monitor	`CLREX`	1
Branch	Conditional	`B<cc> <label>`	1 or 1 + P^c
	Unconditional	`B <label>`	1 + P
	With link	`BL <label>`	1 + P
	With exchange	`BX Rm`	1 + P
	With link and exchange	`BLX Rm`	1 + P
	Branch if zero	`CBZ Rn, <label>`	1 or 1 + P^c
	Branch if non-zero	`CBNZ Rn, <label>`	1 or 1 + P^c
	Byte table branch	`TBB [Rn, Rm]`	2 + P
	Halfword table branch	`TBH [Rn, Rm, LSL#1]`	2 + P

Intéressant pour faire du multi-tâche

Cortex-M4 Technical Reference Manual

Instructions:

State change	Supervisor call	SVC #<imm>	-
	If-then-else	IT... <cond>	1[d]
	Disable interrupts	CPSID <flags>	1 or 2
	Enable interrupts	CPSIE <flags>	1 or 2
	Read special register	MRS Rd, <specreg>	1 or 2
	Write special register	MSR <specreg>, Rn	1 or 2
	Breakpoint	BKPT #<imm>	-
Extend	Signed halfword to word	SXTH Rd, <op2>	1
	Signed byte to word	SXTB Rd, <op2>	1
	Unsigned halfword	UXTH Rd, <op2>	1
	Unsigned byte	UXTB Rd, <op2>	1
Bit field	Extract unsigned	UBFX Rd, Rn, #<imm>, #<imm>	1
	Extract signed	SBFX Rd, Rn, #<imm>, #<imm>	1
	Clear	BFC Rd, Rn, #<imm>, #<imm>	1
	Insert	BFI Rd, Rn, #<imm>, #<imm>	1

Cortex-M4 Technical Reference Manual

Instructions:

Operation	Description	Assembler	Cycles
Reverse	Bytes in word	`REV Rd, Rm`	1
	Bytes in both halfwords	`REV16 Rd, Rm`	1
	Signed bottom halfword	`REVSH Rd, Rm`	1
	Bits in word	`RBIT Rd, Rm`	1
Hint	Send event	`SEV`	1
	Wait for event	`WFE`	1 + W
	Wait for interrupt	`WFI`	1 + W
	No operation	`NOP`	1
Barriers	Instruction synchronization	`ISB`	1 + B
	Data memory	`DMB`	1 + B
	Data synchronization	`DSB <flags>`	1 + B

Cortex-M4 Technical Reference Manual

Instructions: DSP

Table 3-2 Cortex-M4 DSP instruction set summary

Operation	Description	Assembler	Cycles
Multiply	32-bit multiply with 32-most-significant-bit accumulate	SMMLA	1
	32-bit multiply with 32-most-significant-bit subtract	SMMLS	1
	32-bit multiply returning 32-most-significant-bits	SMMUL	1
	32-bit multiply with rounded 32-most-significant-bit accumulate	SMMLAR	1
	32-bit multiply with rounded 32-most-significant-bit subtract	SMMLSR	1
	32-bit multiply returning rounded 32-most-significant-bits	SMMULR	1

Cortex-M4 Technical Reference Manual

Instructions DSP

3.6.7 SMMLA and SMMLS

Signed Most Significant Word Multiply Accumulate and Signed Most Significant Word Multiply Subtract.

Syntax

`op{R}{cond} Rd, Rn, Rm, Ra`

where:

op Is one of:

- `SMMLA` Signed Most Significant Word Multiply Accumulate.
- `SMMLS` Signed Most Significant Word Multiply Subtract.

`R` Is a rounding error flag. If *R* is specified, the result is rounded instead of being truncated. In this case the constant `0x80000000` is added to the product before the high word is extracted.

cond Is an optional condition code, see *Conditional execution* on page 3-18.

Rd Specifies the destination register.

Rn, Rm Are registers holding the first and second multiply operands.

Ra Specifies the register holding the accumulate value.

Cortex-M4 Generic User Guide

Instructions DSP

Syntax

`op{R}{cond} Rd, Rn, Rm, Ra`

Operation

The SMMLA instruction interprets the values from *Rn* and *Rm* as signed 32-bit words.

The SMMLA instruction:

- Multiplies the values in *Rn* and *Rm*.
- Optionally rounds the result by adding `0x80000000`.
- Extracts the most significant 32 bits of the result.
- Adds the value of *Ra* to the signed extracted value.
- Writes the result of the addition in *Rd*.

The SMMLS instruction interprets the values from *Rn* and *Rm* as signed 32-bit words.

The SMMLS instruction:

- Multiplies the values in *Rn* and *Rm*.
- Optionally rounds the result by adding `0x80000000`.
- Extracts the most significant 32 bits of the result.
- Subtracts the extracted value of the result from the value in *Ra*.
- Writes the result of the subtraction in *Rd*.

Cortex-M4 Generic User Guide

Instructions DSP

Examples

```
SMMLA      R0, R4, R5, R6   ; Multiplies R4 and R5, extracts top 32 bits, adds
                            ; R6, truncates and writes to R0
SMMLAR     R6, R2, R1, R4   ; Multiplies R2 and R1, extracts top 32 bits, adds
                            ; R4, rounds and writes to R6
SMMLSR     R3, R6, R2, R7   ; Multiplies R6 and R2, extracts top 32 bits,
                            ; subtracts R7, rounds and writes to R3
SMMLS      R4, R5, R3, R8   ; Multiplies R5 and R3, extracts top 32 bits,
                            ; subtracts R8, truncates and writes to R4.
```

Cortex-M4 Generic User Guide

Outil de gestion de versions

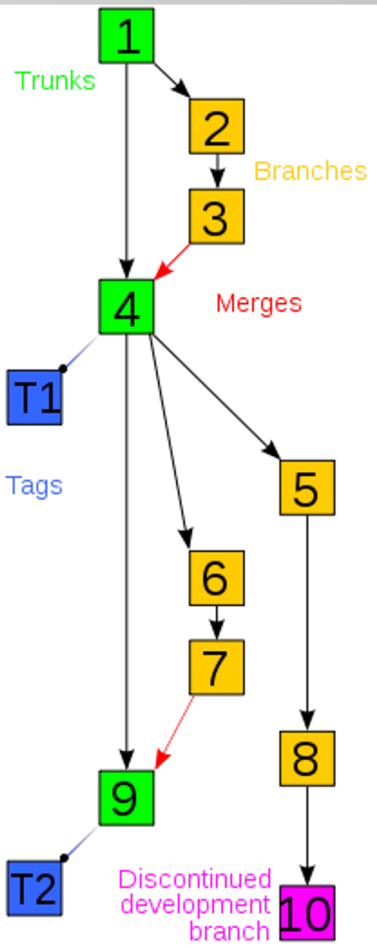
Prof. Francesco Mondada, Daniel Burnier, Antoine Martin
IEM - STI - EPFL

Outils logiciel: Gestion de versions

La **gestion de versions** consiste à maintenir l'ensemble des versions des codes source. Chaque étape d'avancement des logiciel est appelée **version** (ou *revision*). Les différentes versions sont nécessairement liées à travers des **modifications** : des fichier modifiés, ajoutés ou supprimés

Les fichiers *versionnés* sont mis à dispositions sur un **dépôt** géré par un logiciel de gestion de versions.

- Gestion de versions centralisée : il n'existe qu'un seul dépôt des versions (Subversion (SVN)).
- **Gestion de versions décentralisée** : il existe plusieurs dépôts pour un même logiciel (Git et Mercurial).



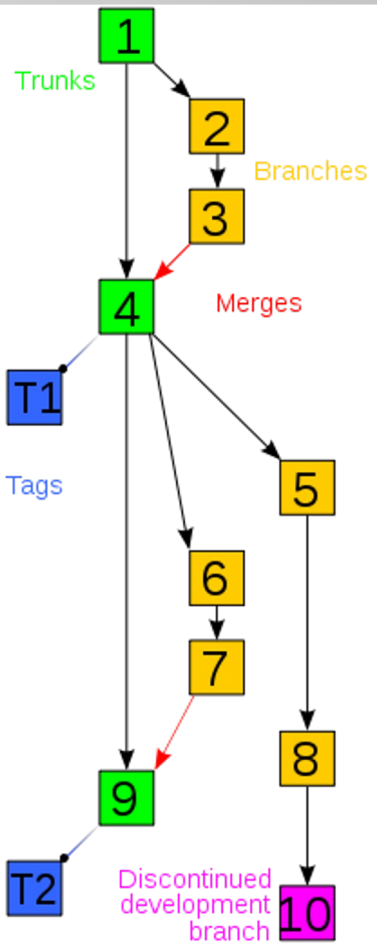
https://fr.wikipedia.org/wiki/Gestion_de_versions

Gestion de versions: Git et Github

Git, et le repository online GitHub sont très répandus.

Pour être rapidement efficaces il est conseillé d'utiliser un client avec une interface utilisateur (au lieu de la ligne de commande) comme par exemple GitKraken

De plus en plus d'outils de développement (comme celui qu'on va utiliser, Visual Studio Code) intègrent la gestion de version.



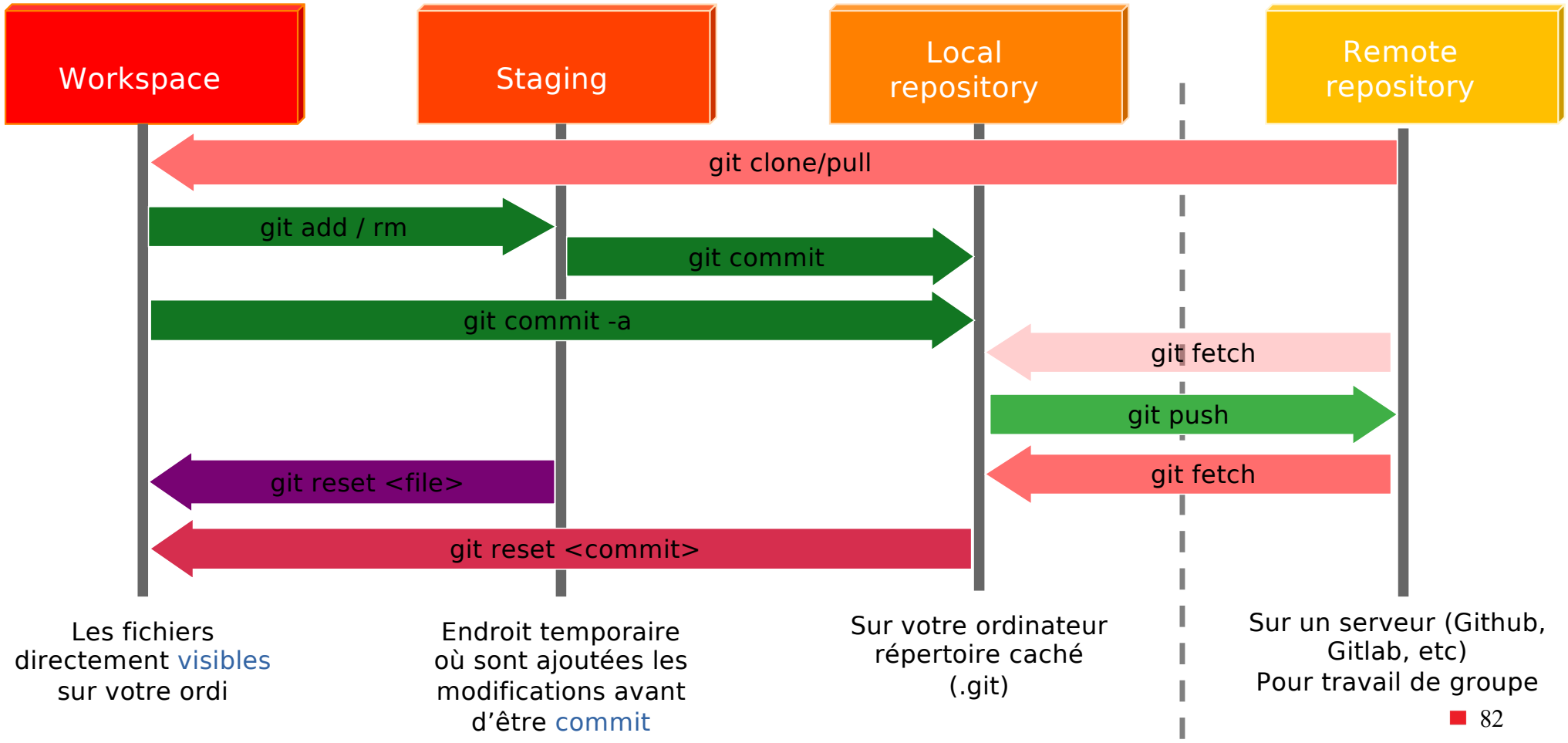
https://fr.wikipedia.org/wiki/Gestion_de_versions

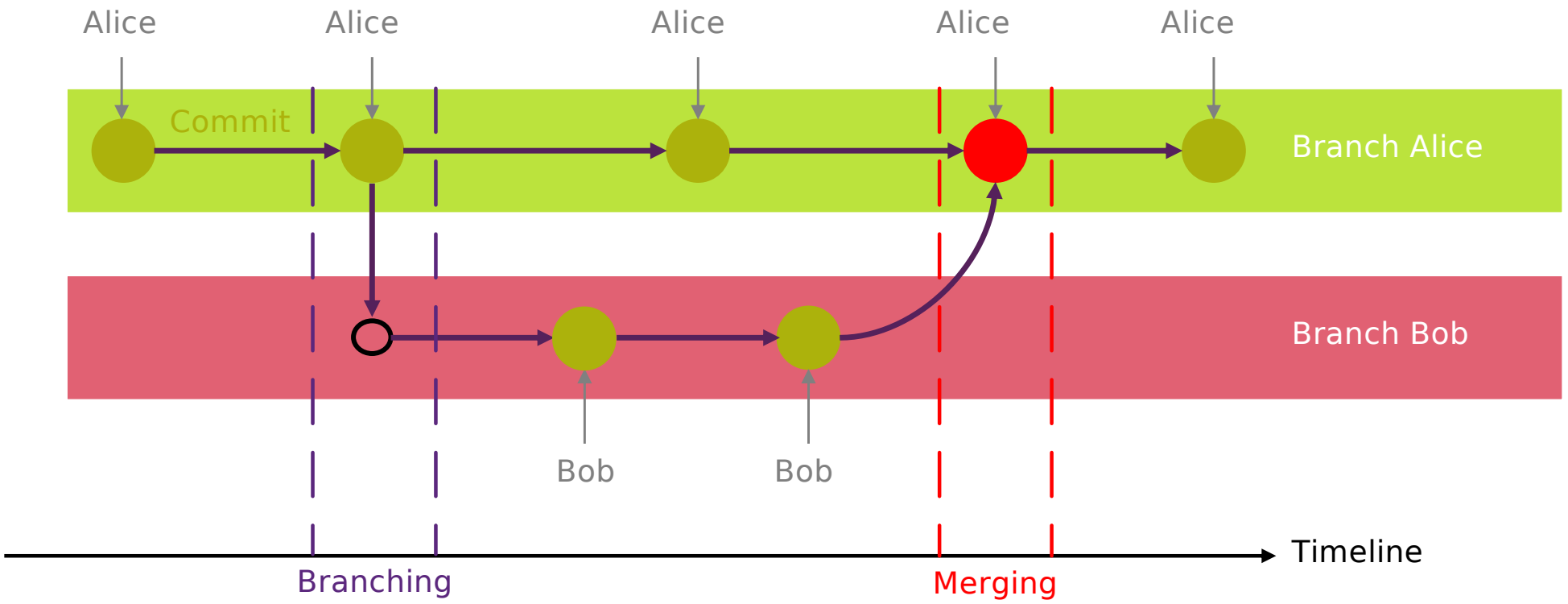
Git

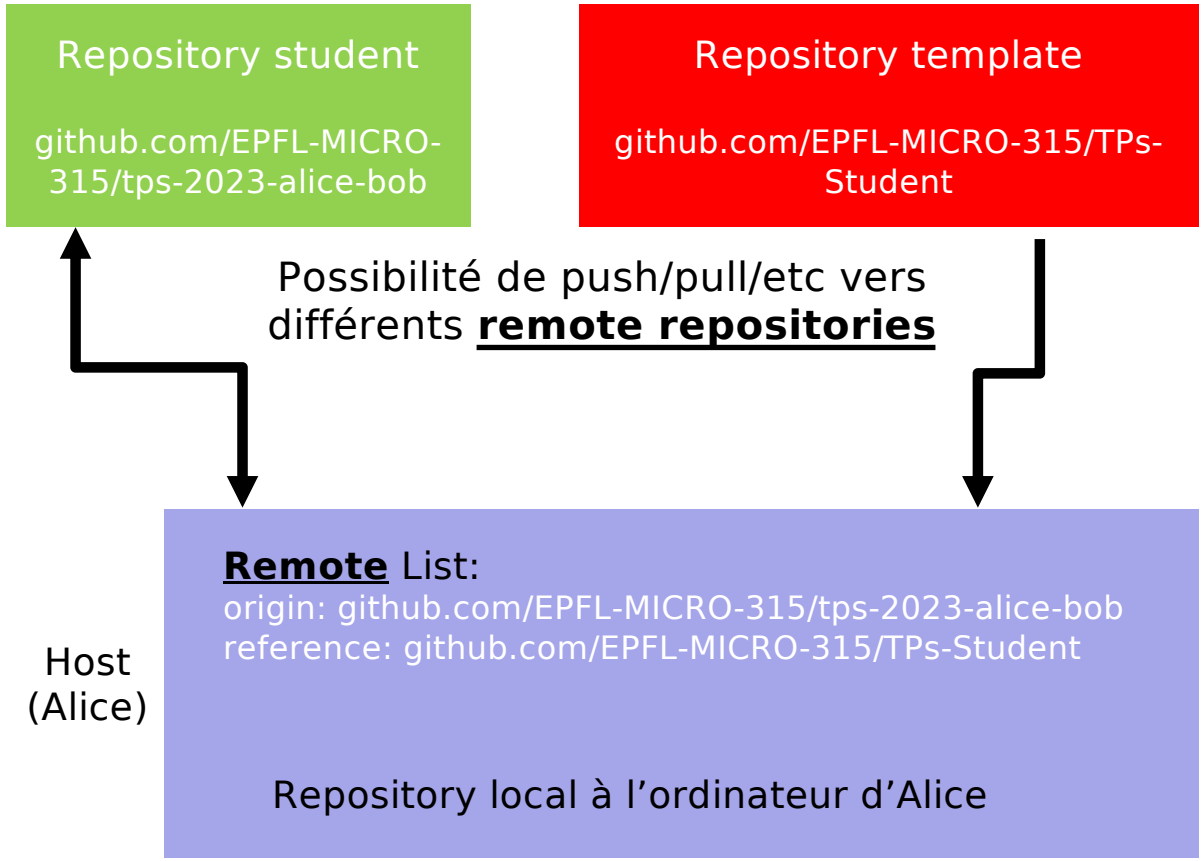
- gestion des versions
    - possibilité de sauvegarde des versions et de roll back à une version de code antérieure
    - commit: ”snapshot” d’une version précise de mon code, qui est sauvée
    - checkout: prendre une version dans le repository comme version de travail

- collaboration
    - facilite le travail en groupe
    - copie locale sur l’ordinateur, fonctionne en offline
    - échange par un repository online partagé (e.g Github, GitLab)
    - 1 branch par personne en général
    - merging: fusionner 2 branch

- un standard dans l’industrie


Workspace
Staging
Local repository
Remote repository
git clone/pull
git add / rm
git commit
git commit -a
git fetch
git push
git fetch
git reset <file>
git reset <commit>
Les fichiers directement visibles sur votre ordi
Endroit temporaire où sont ajoutées les modifications avant d'être commit
Sur votre ordinateur répertoire caché (.git)
Sur un serveur (Github, Gitlab, etc) Pour travail de groupe


Alice
Alice
Alice
Alice
Alice
Commit
Branch Alice
Branch Bob
Bob
Bob
Timeline
Branching
Merging



Cas des TPs

Repository student - modifiable
- code du groupe
- remote name: origin

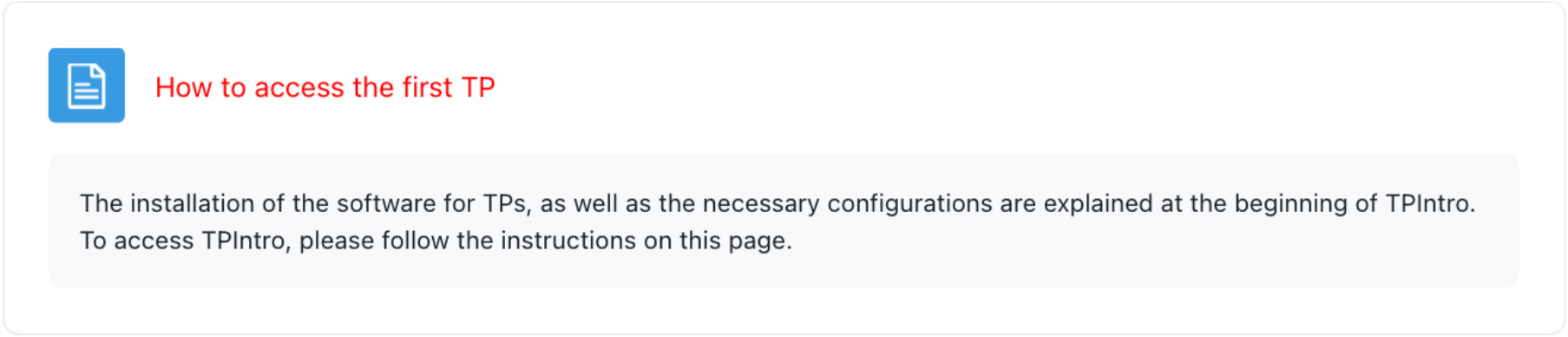
Repository template – non modifiable
- code de la donnée des exercices
- code des solutions
- remote name: reference

Avantages
Exercices, solutions et code du groupe accessibles depuis le même dossier sur un ordinateur

Github Classroom

- Pour accéder aux TPs, il est nécessaire d'utiliser Github :)
- il est n'est pas possible d'accéder aux TPs autrement

- Le repository de chaque groupe est stocké sous l'organisation EPFL-MICRO-315
- => un assistant peut directement accéder au code facilitant donc le debugging



CheatSheet

- git clone <url>: Clone (download) a repository that already exists on a server (Github), including all of the files, branches, and commits
- git status: Always good to use, this command shows you what branch you're on, what files are in the working or staging directory, and any other important information
- git branch: This shows the existing branches in your local repository
- git branch <branch-name>: Create a branch from your current location
- git branch –all: See all branches, both the local ones on your machine, and the remote tracked branches
- git checkout <branch-name>: Switches to the specified branch and updates the working directory
- origin: name given by default to the remote from which the repo was cloned

CheatSheet

- git add/rm <file>: Snapshots the file in preparation for versioning, adding/removing it to the staging area
- git commit -m "<descriptive message>": Records file snapshots permanently in version history
- git fetch: Updates your current local repository branch with all new commits from the corresponding remote branch on GitHub
- git pull: Updates your current local working branch with all new commits from the corresponding remote branch on GitHub
- git pull is a combination of git fetch and git merge
- git push: Uploads all local branch commits to the remote
- git log: Browse and inspect the evolution of project files
- git remote -v: Show the associated remote repositories and their stored name, like origin

CheatSheet

- git reset <file>: Unstage a file while retaining the changes in working directory
- git diff: diff of what is changed but not staged
- git diff --staged: diff of what is staged but not yet committed
- git merge <branch>: merge the specified branch's history into the current one
- git reset –-hard <commit>: clear staging area, rewrite working tree from specified commit
- git config –global user.name "<username>": set a name that is identifiable for credit when review version history
- git config –global user.email "<email>": set an email that will be associated with each history marker
- .gitignore: a file specifying any file/folder to be ignored when staging modifications

Introduction au dossier électronique

Jeudi 20.02.2025 10:15
*en **CM 1 1***

Prenez votre ordinateur, installez l'IDE

Formation des groupes: merci de les définir avant mercredi midi
le matériel (robot) et la configuration github en dépends

TP de ce Jeudi (essentiel, à ne pas rater!)

Installation et prise en main des outils utilisés en TP:

- Visual Studio Code
- Pyenv
- E-puck2 (distribution)
- Git

Utilisation de Visual Studio Code pour

- Programmer
- Debugger

sur cible e-puck2.

Ces outils seront utilisés durant tous les TPs, donc il est essentiel de les maitriser !!

TP de ce Jeudi (essentiel, à ne pas rater!)

Avant le TP:

- Créez votre groupe et inscrivez-le d'ici mercredi (lien sur Moodle)
- Installez l'IDE en suivant le tout début du TPIntro (Partie 1, le reste sera réalisée en TP)
- Si vous avez des problèmes d'installation, demandez de l'aide aux assistants durant le TP ou créez une issue GitHub

N'oubliez pas de prendre votre ordinateur aux TPs !

Introduction to the e-puck2 robot

Francesco Mondada
IEM - STI - EPFL

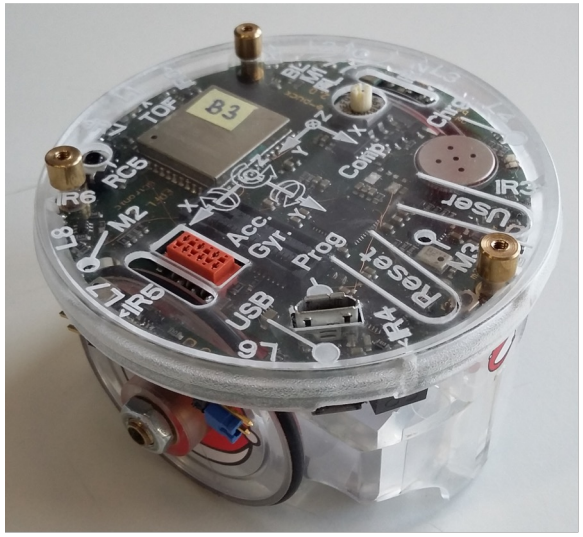
Introduction

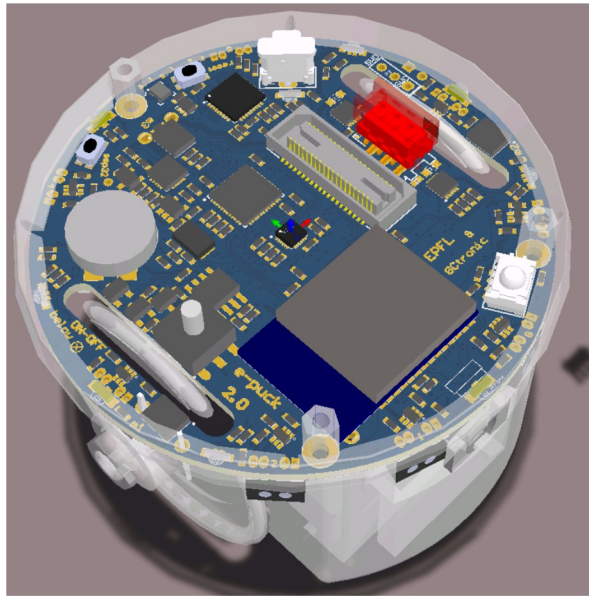
The goal of this intro is to provide you with some elements of the e-puck2 mobile robot and prepare you for the TP1

The e-puck2 mobile robot

Main features:

- Cylindrical, Ø 70mm
- STM32f407 processor
- Two stepper motors
- Multiple LEDs
- Many sensors:
  - ✓Camera
  - ✓Sound
  - ✓IR proximity sensors
  - ✓IMU (Acc., Gyro., ~~Mag.~~)
  - ✓Distance sensor (ToF)
- Li-ion accumulator
- Bluetooth/Wifi wireless
- USB communication
- Onboard Debug Server (GDB)
- Open hardware



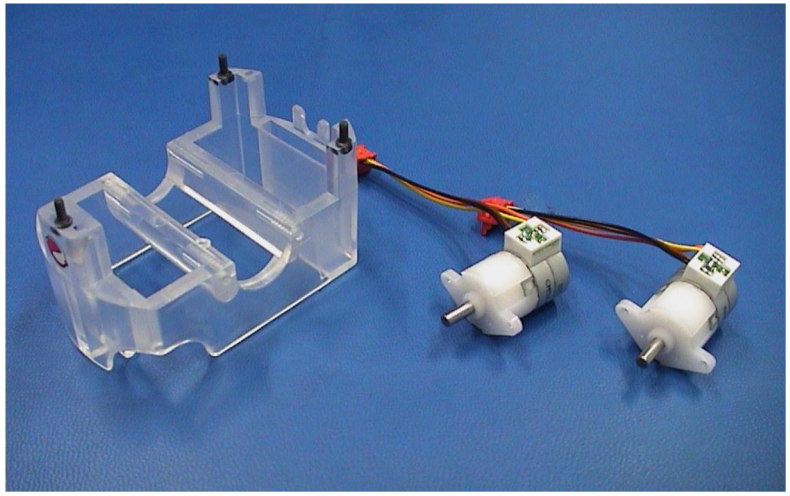


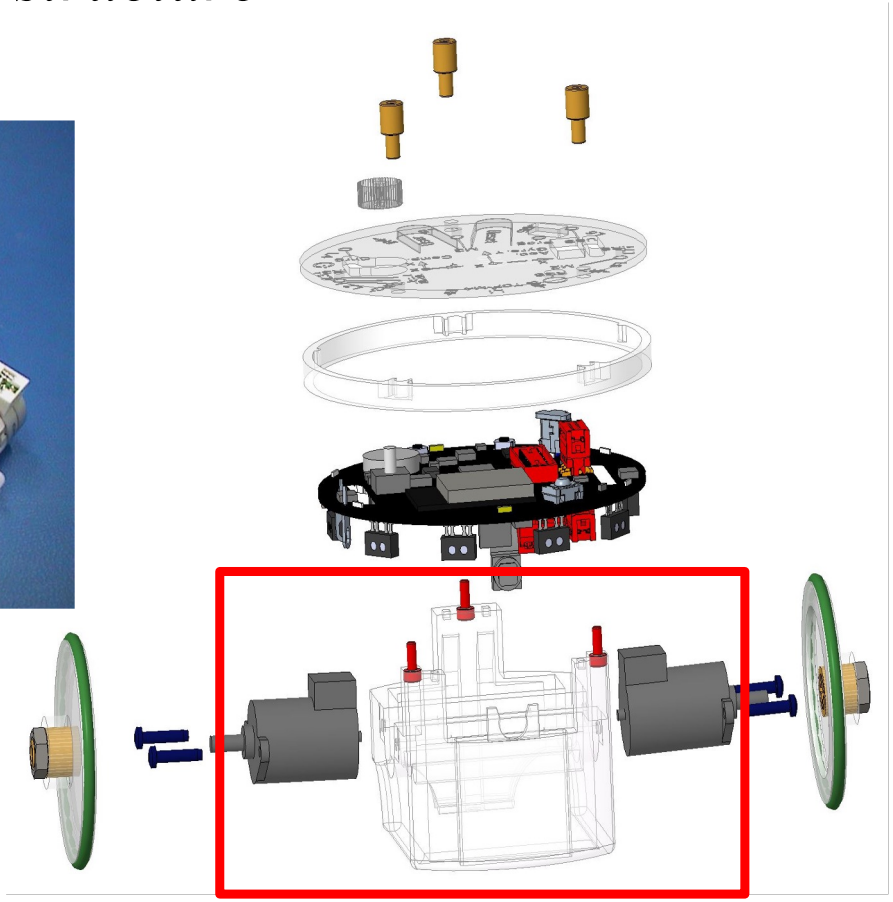
The e-puck2 open hardware license

The specifications of the e-puck2 mobile robot are "open source hardware". You can redistribute them and/or modify them under the terms of the e-puck2 Robot Open Source Hardware License as published by EPFL. You should have received a copy of the EPFL e-puck2 Robot Open Source Hardware License along with these specifications; if not, write to the Ecole Polytechnique Fédérale de Lausanne (EPFL), Industrial Relations Office, Station 10, 1015 Lausanne, Switzerland.

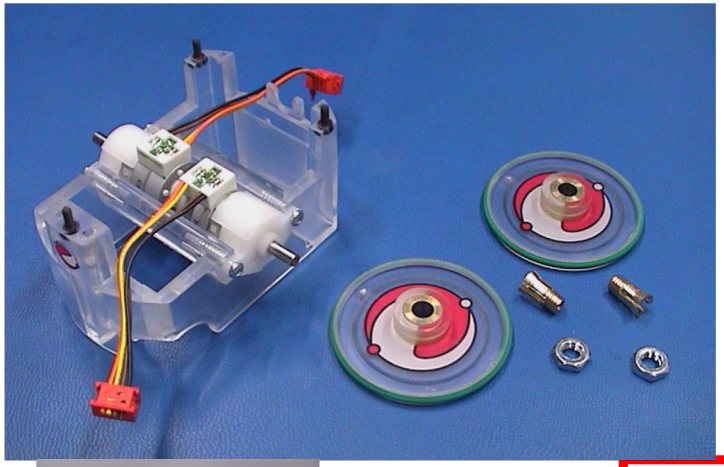
These specifications are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. For more details, see the EPFL e-puck2 Robot Open Source Hardware License.

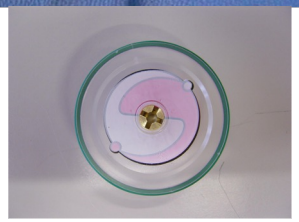
e-puck2 mobile robot mechanical structure

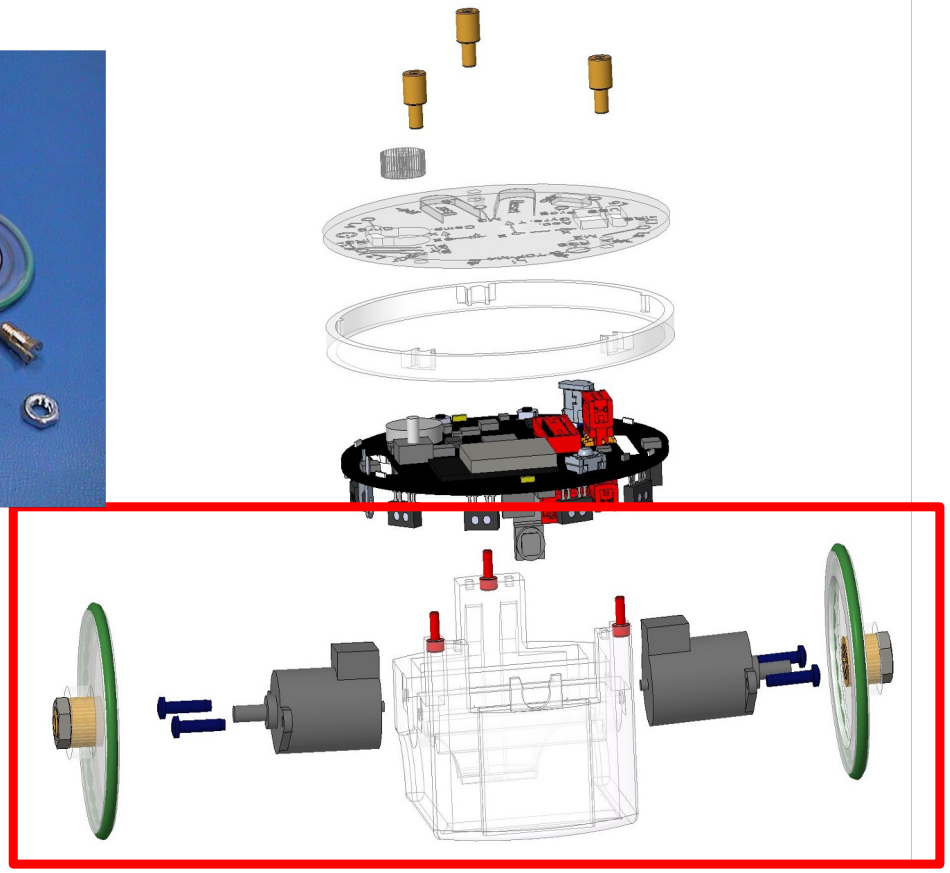




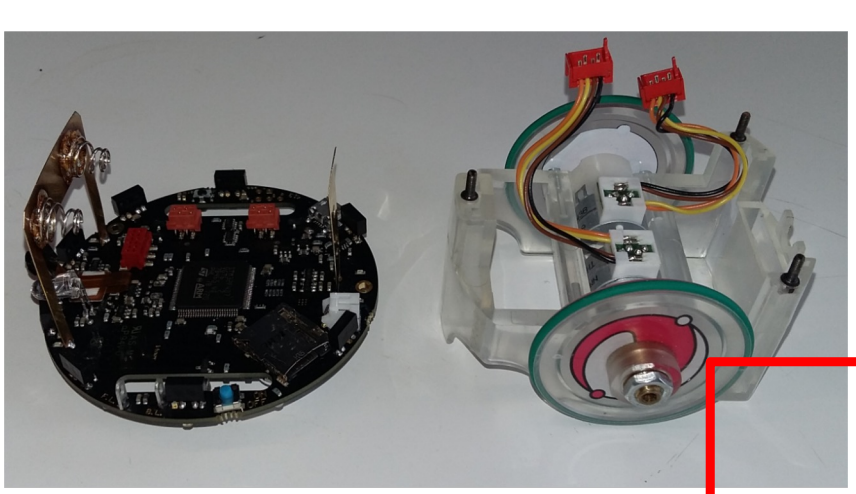
e-puck2 mobile robot mechanical structure

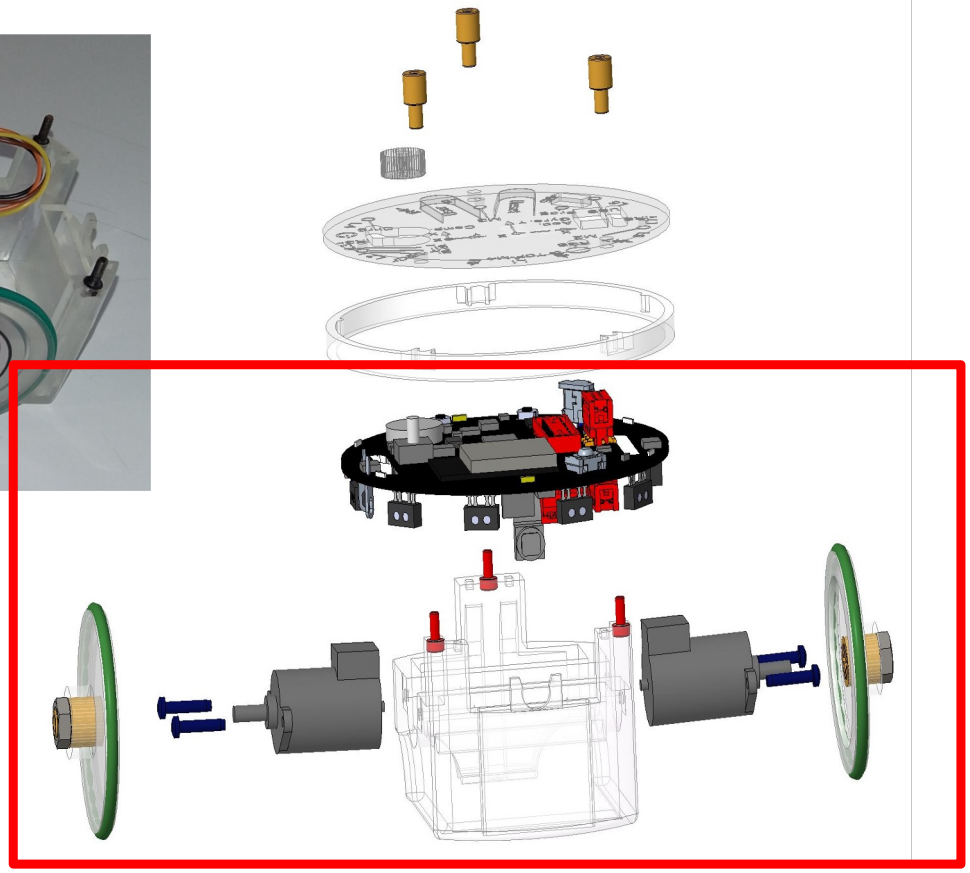




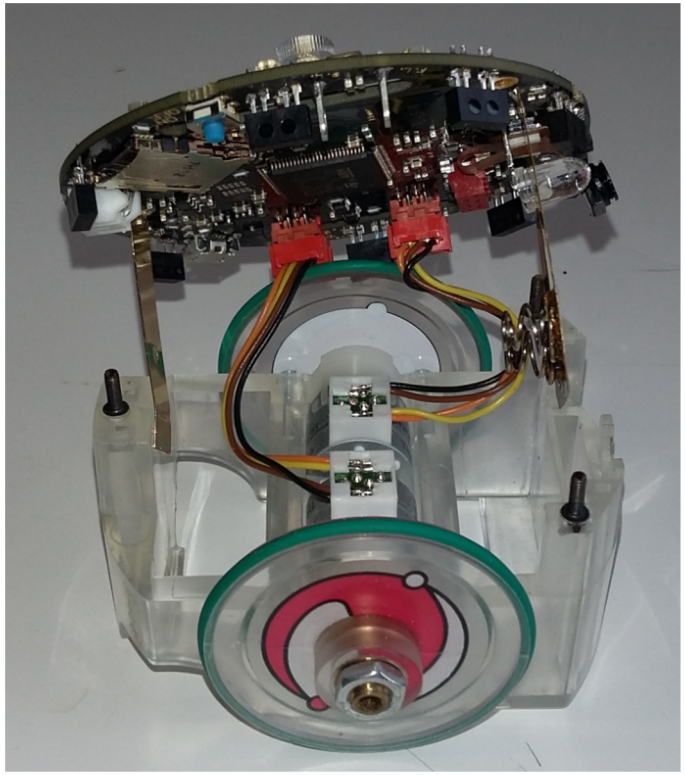


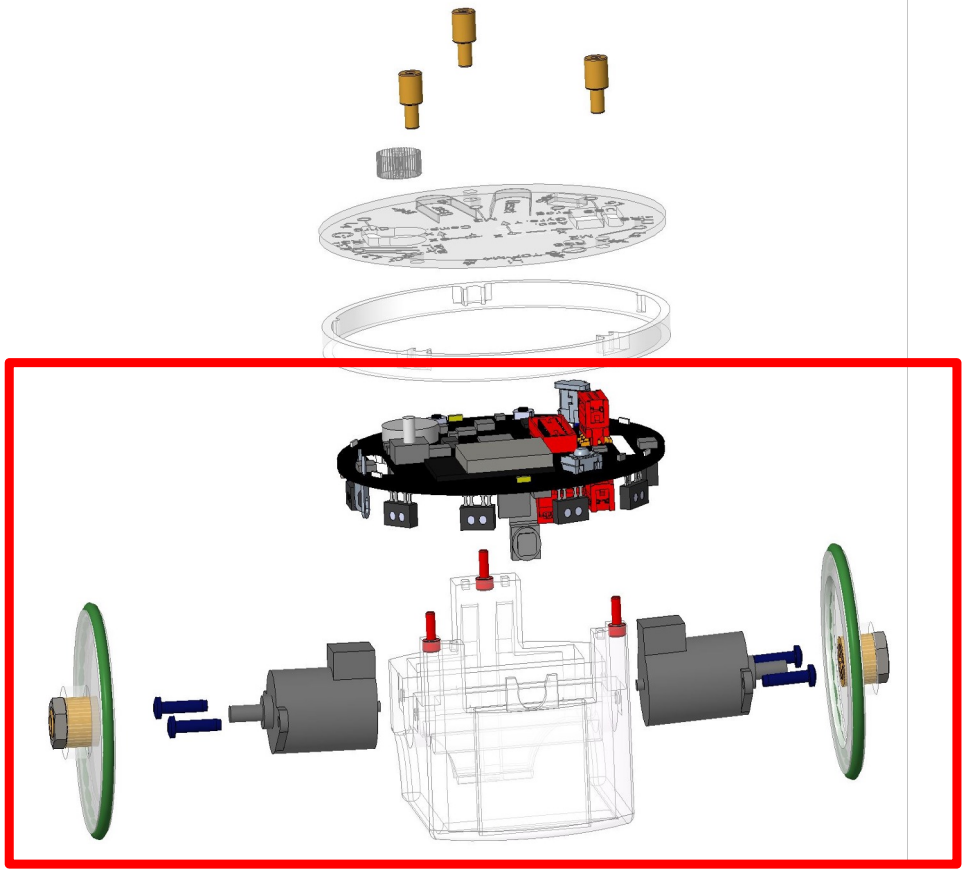
e-puck2 mobile robot mechanical structure



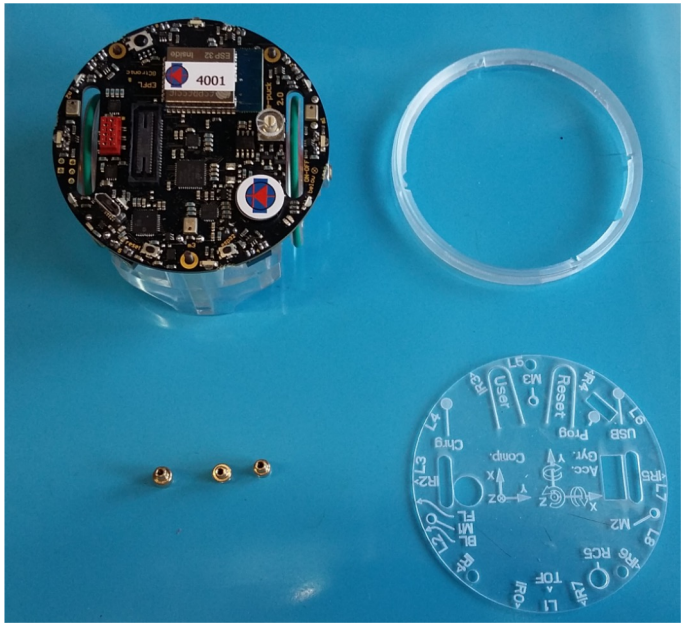


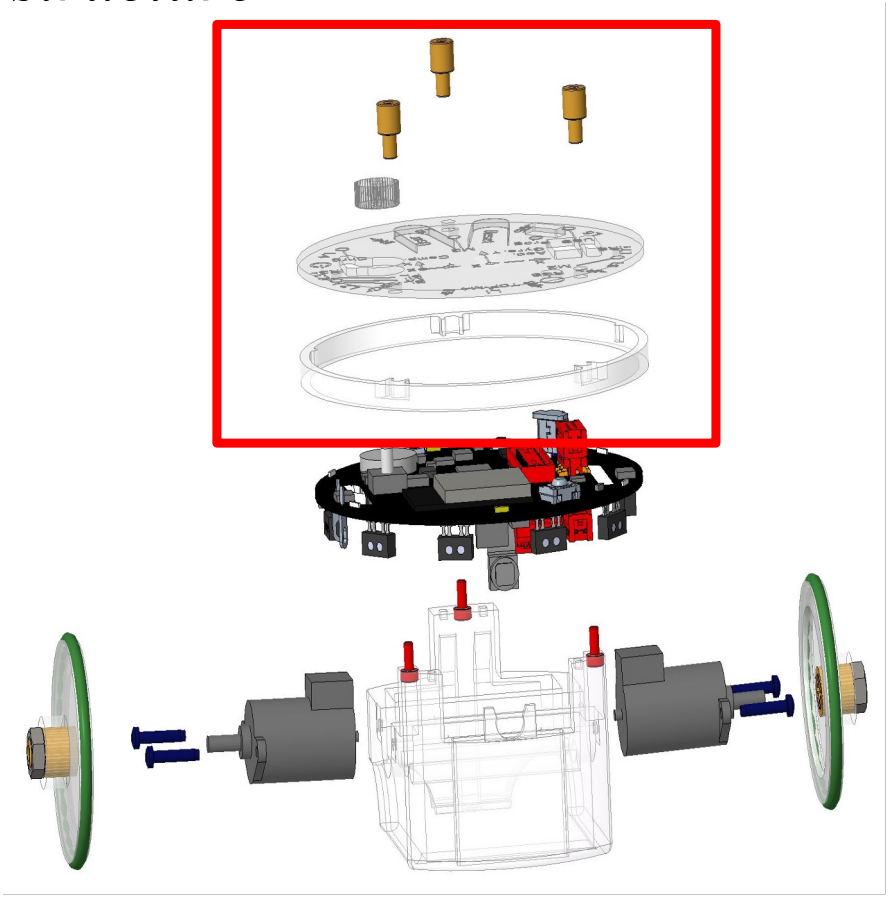
e-puck2 mobile robot mechanical structure



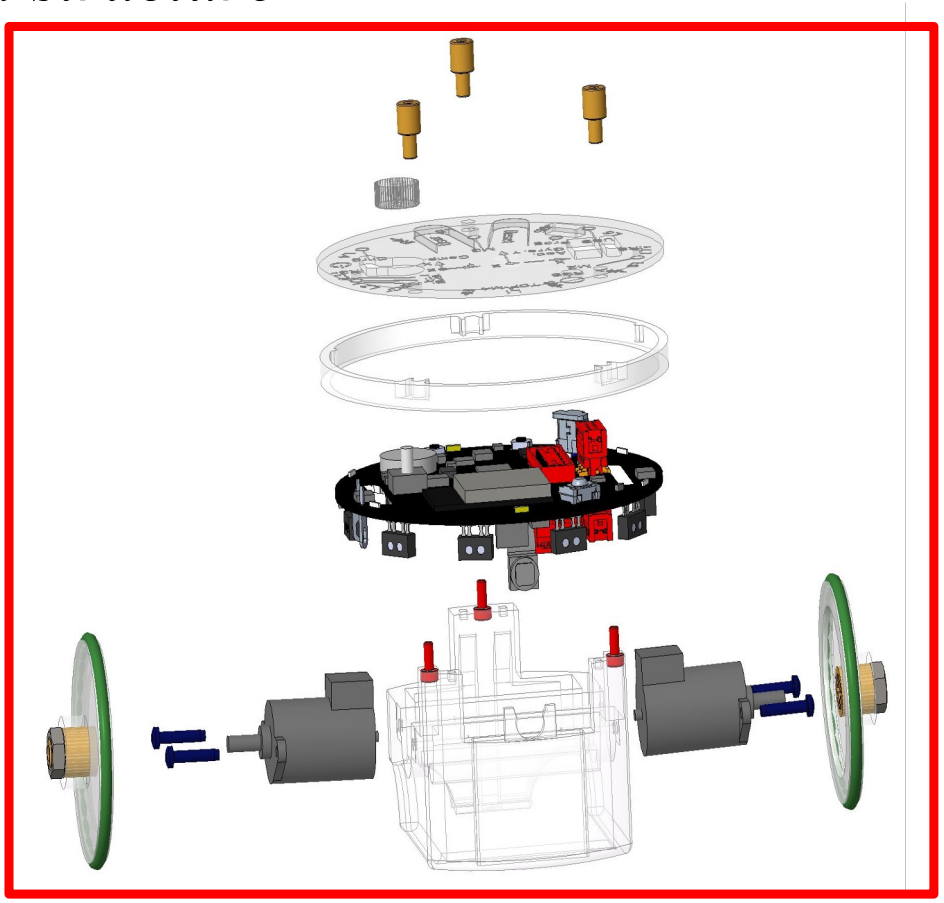


e-puck2 mobile robot mechanical structure

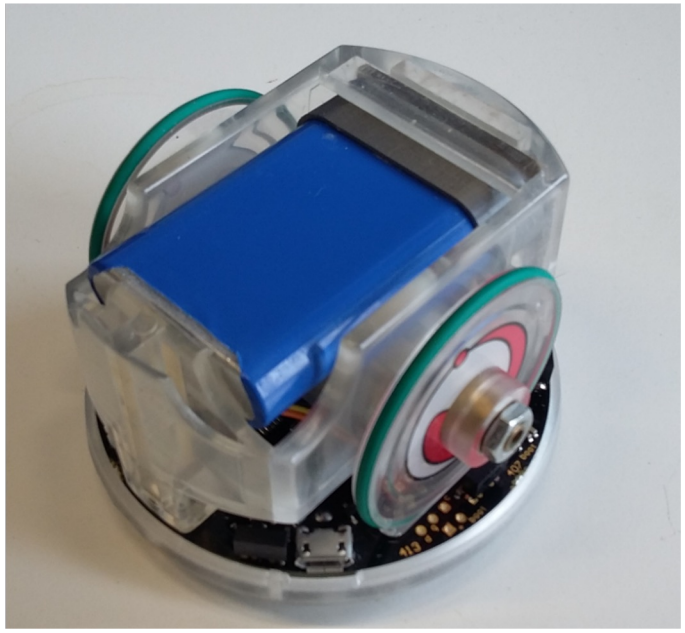


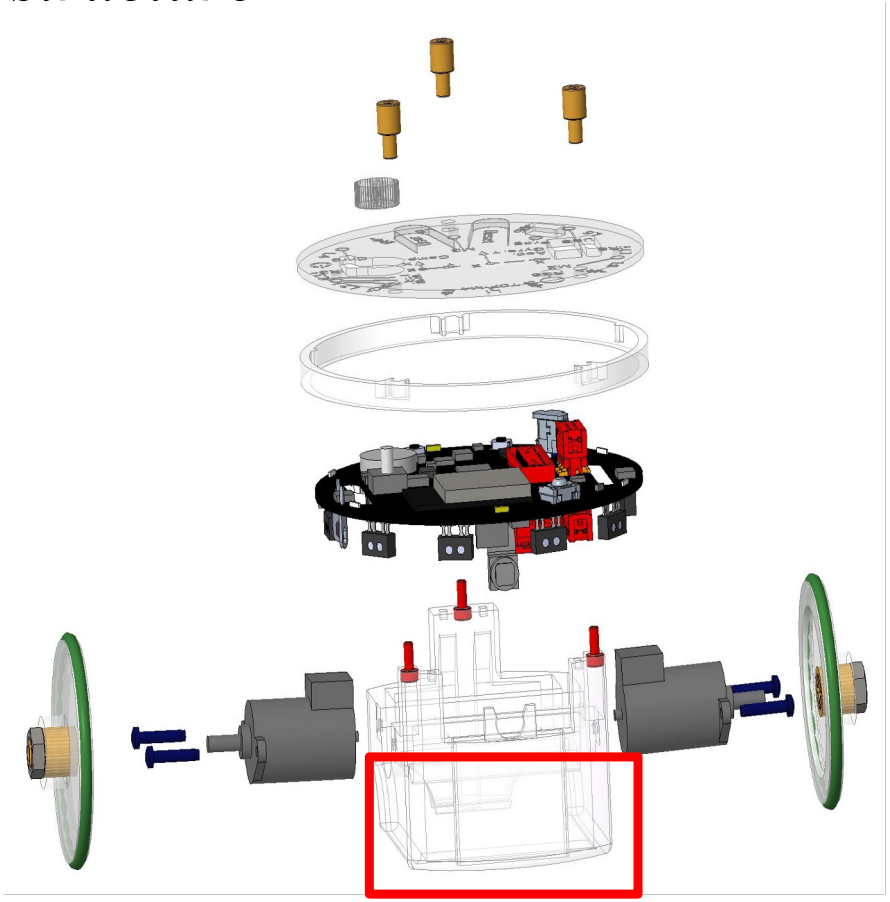


e-puck2 mobile robot mechanical structure



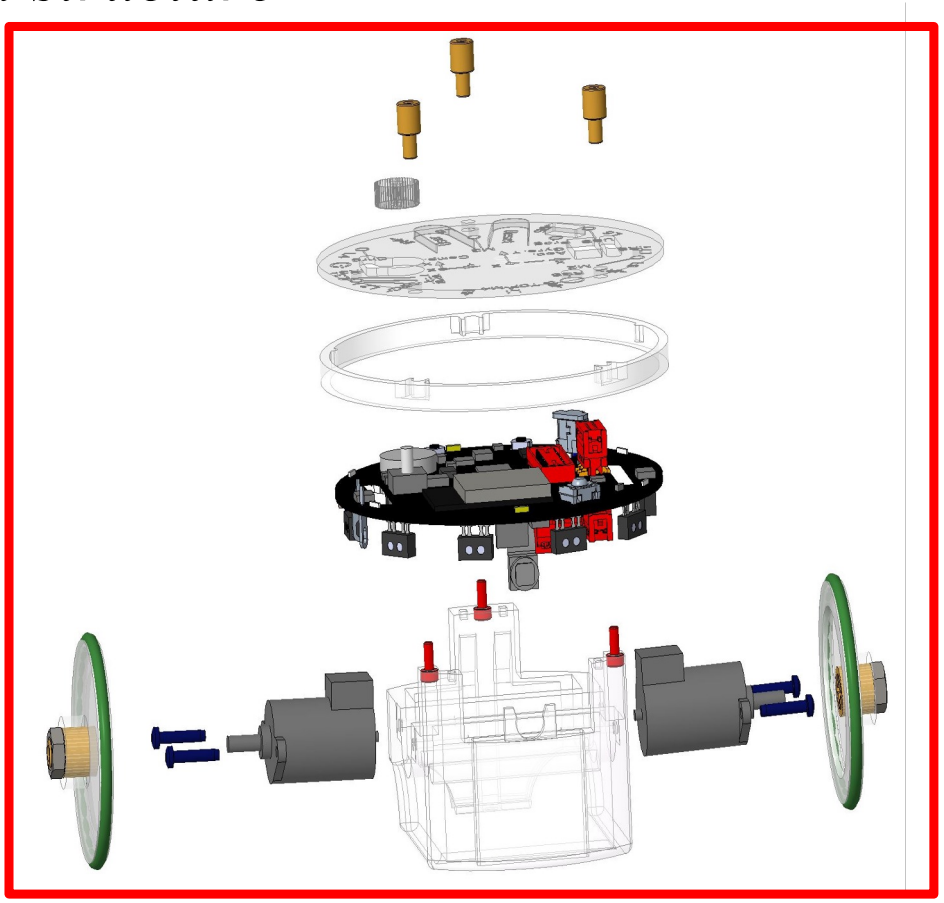
e-puck2 mobile robot mechanical structure



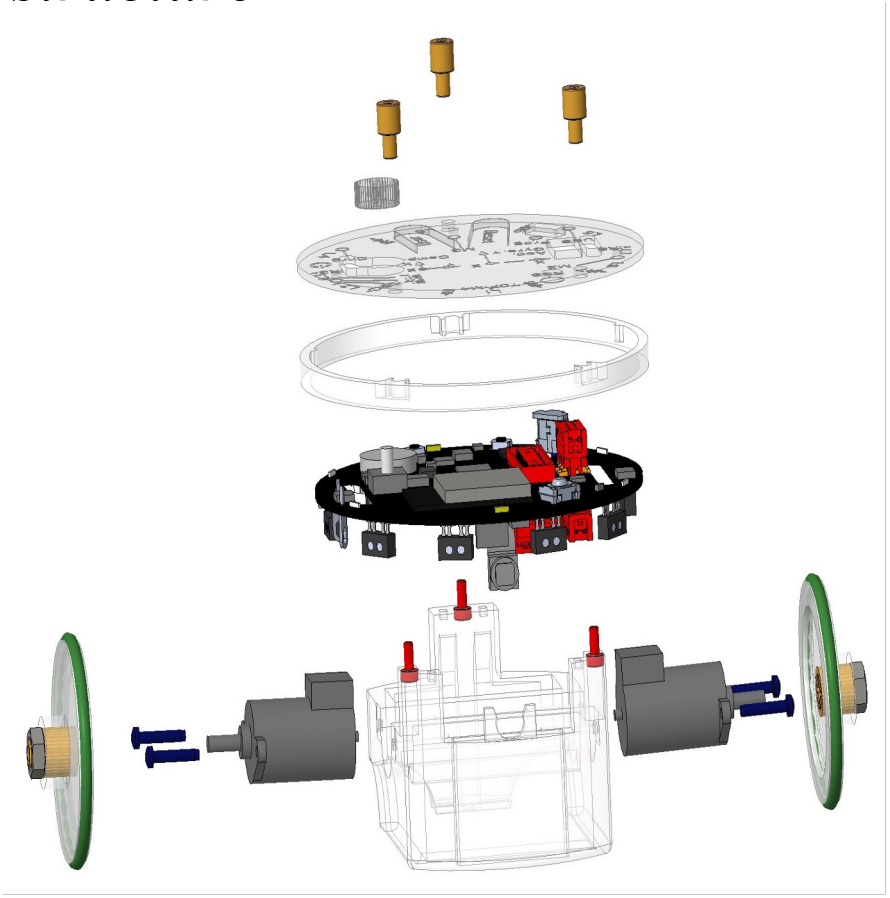


Do not extract the battery pack from the e-puck and do not charge it yourself !!!!

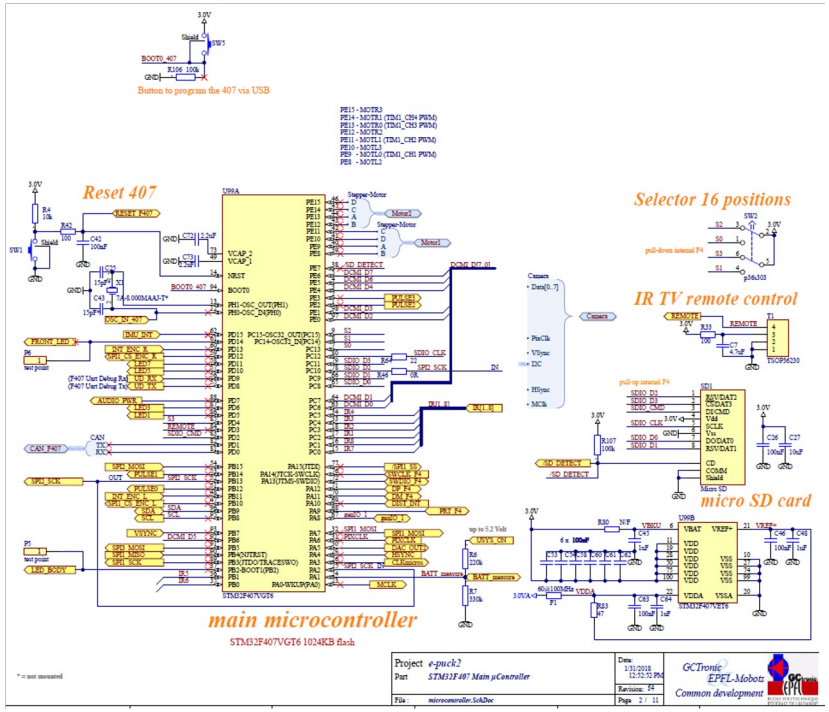
e-puck2 mobile robot mechanical structure

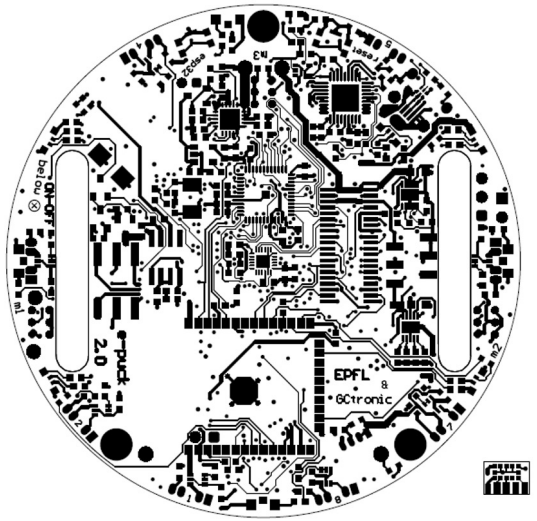


e-puck2 mobile robot mechanical structure



Dossier Electronique

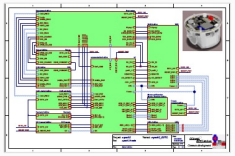




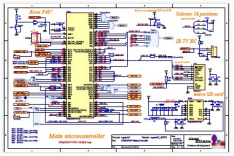
EPFL

Savoir extraire des informations

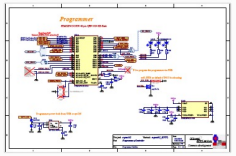
Systèmes embarqués et robotique



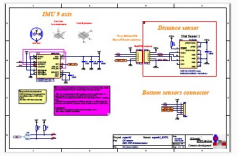
1



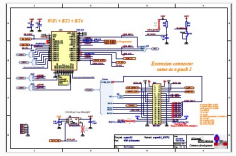
2



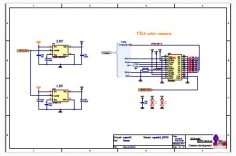
3



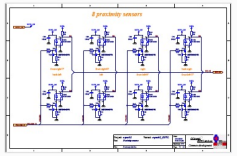
4



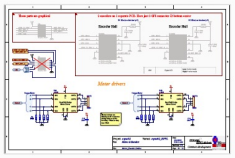
5



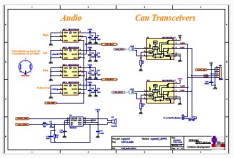
6



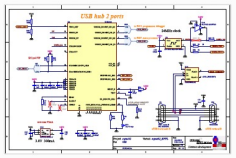
7



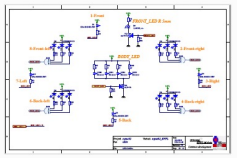
8



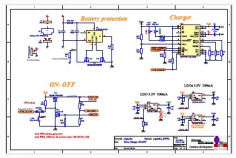
9



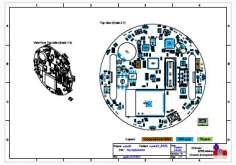
10



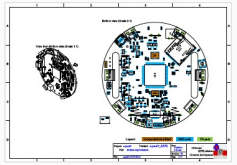
11



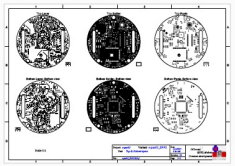
12



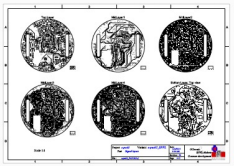
13



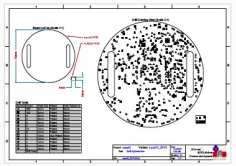
14



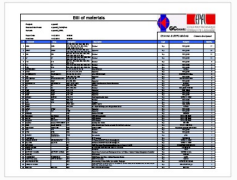
15



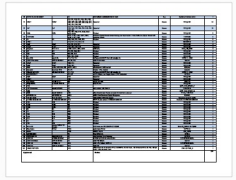
16



17



18



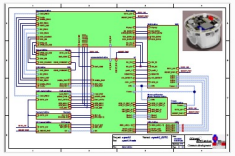
19

EPFL

Savoir extraire des informations

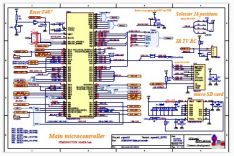
Systèmes embarqués et robotique

Vue générale

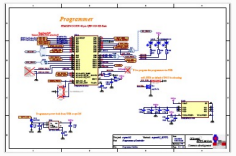


1

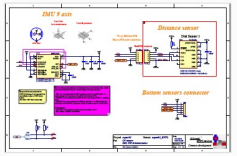
Schémas



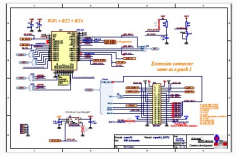
2



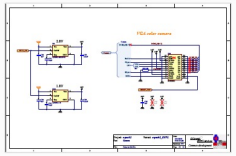
3



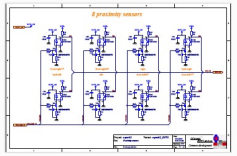
4



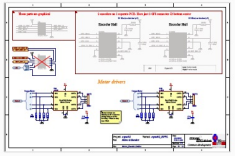
5



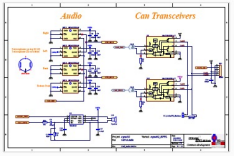
6



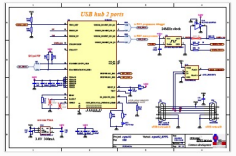
7



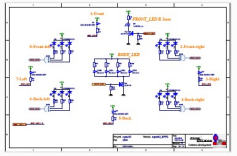
8



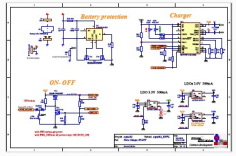
9



10

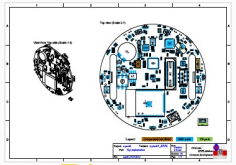


11

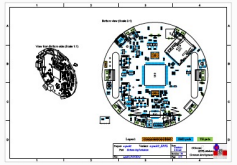


12

Placement composants

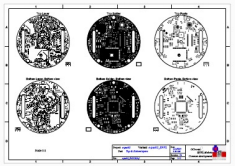


13

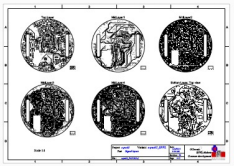


14

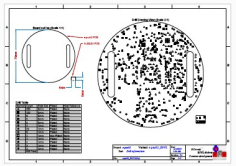
Fabrication (cuivre et trous de passage)



15

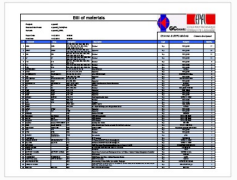


16

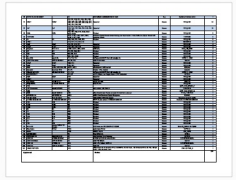


17

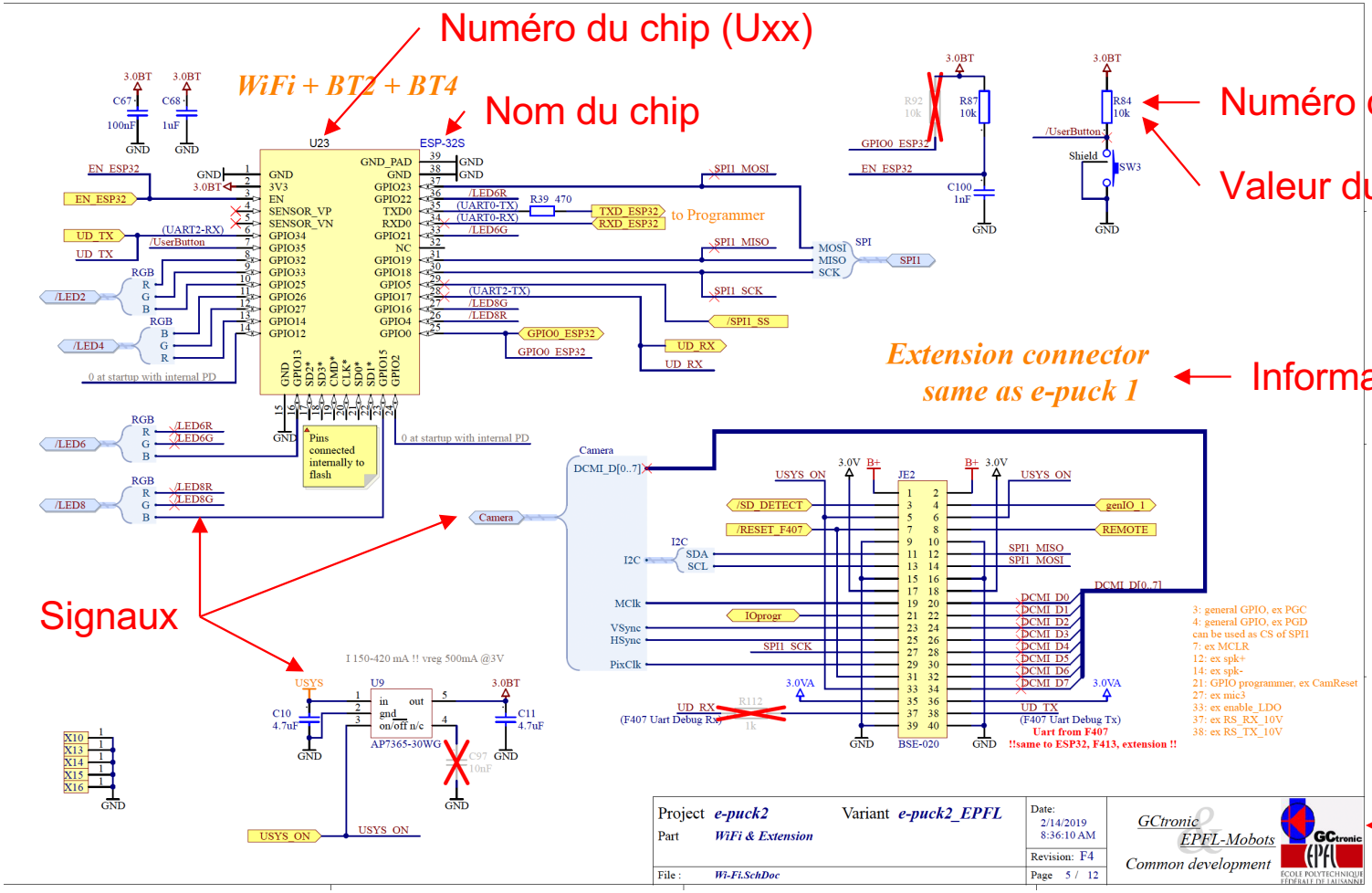
BOM (Bill Of Material)



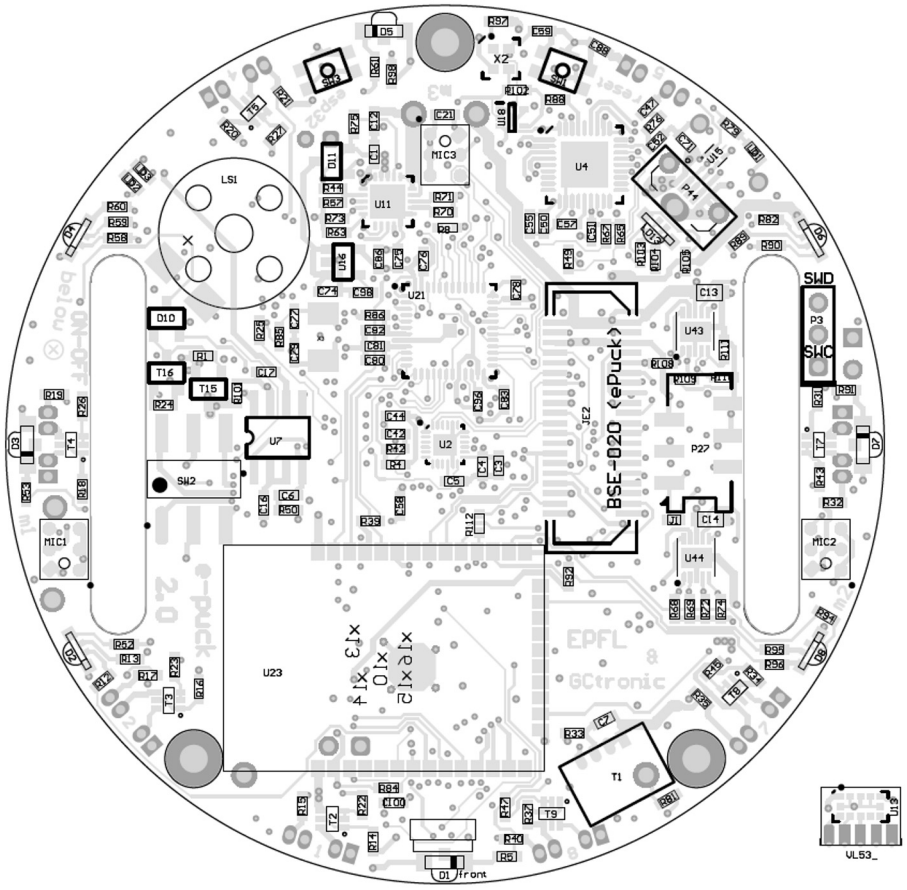
18



19

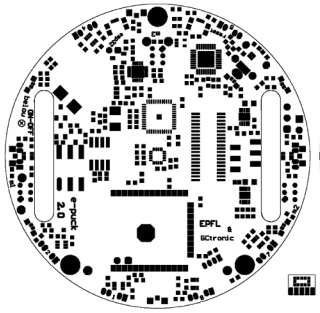

Numéro du chip (Uxx)
WiFi + BT2 + BT4
Nom du chip
U23
ESP-32S
Numéro du composant
Valeur du composant
Extension connector
same as e-puck 1
Informations locale (commentaire)
Signaux
Informations globales
Project e-puck2
Variant e-puck2_EPFL
Part WiFi & Extension
File : Wi-Fi.SchDoc
Date: 2/14/2019 8:36:10 AM
Revision: F4
Page 5 / 12
GCtronic & EPFL-Mobots
Common development
AP7365-30WG
BSE-020
to Programmer
Pins connected internally to flash
I 150-420 mA !! vreg 500mA @3V
Uart from F407
!!same to ESP32, F413, extension !!

Routage et placement des composants



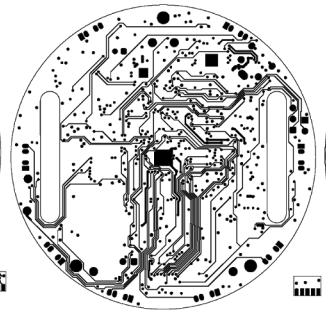
Couches de connexion électrique (pistes, pastilles, etc.)

Top Solder



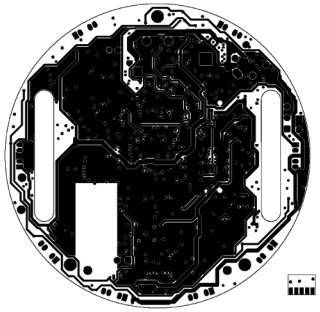
Top Layer

Layer 2

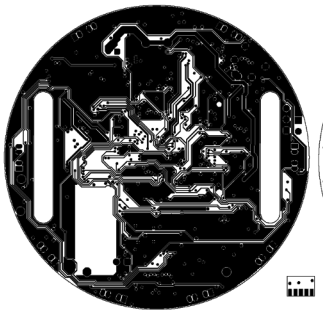


Layer 3

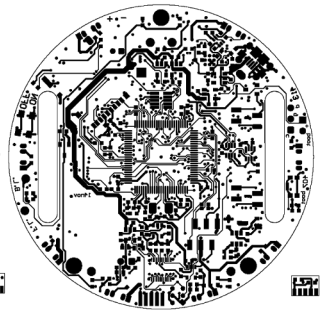
Layer 4



Layer 5



Bottom Layer



Bottom Solder