



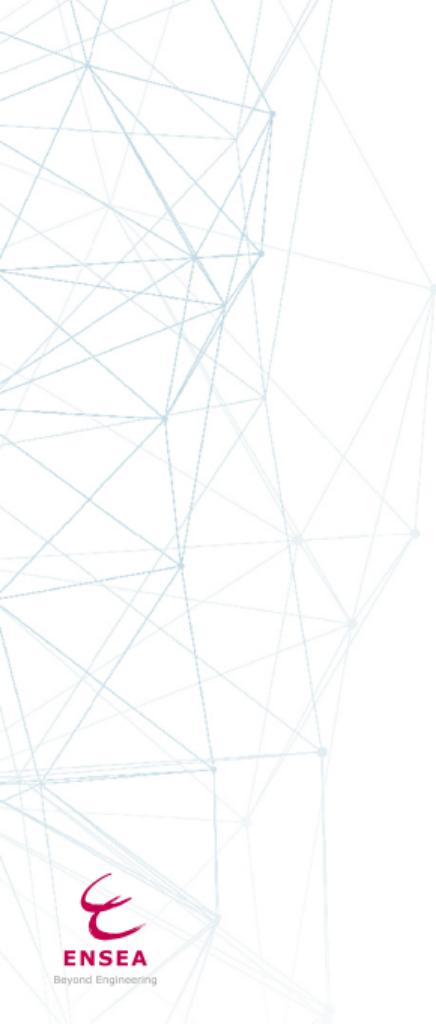
ACTIONNEUR ET AUTOMATIQUE APPLIQUÉE

[ESE_3745]

ENSEA

Nicolas Papazoglou nicolas.papazoglou@ensea.fr

November 14, 2021



Introduction



Organisation

Organisation

Objectifs :

Ce cours a pour objectif de présenter de manière fonctionnelle les différents types d'actionneurs électriques pouvant être utilisés dans des applications industrielles embarquées.

- Les différents types de machines électriques (continu, alternatif, pas à pas, etc.).
- Les principaux capteurs associés : mesures de courant, vitesse, position.
- Modélisation des actionneurs électriques : ensembles formés de la machine, de son dispositif d'alimentation, des capteurs associés et de la commande.
- Réalisation d'un système de commande numérique directe et asservissement d'actionneur électrique par micro-controleur STM32.
- Les travaux pratiques mettront en œuvre la commande numérique pour moteur à courant continu avec asservissement.

Organisation

Objectifs :

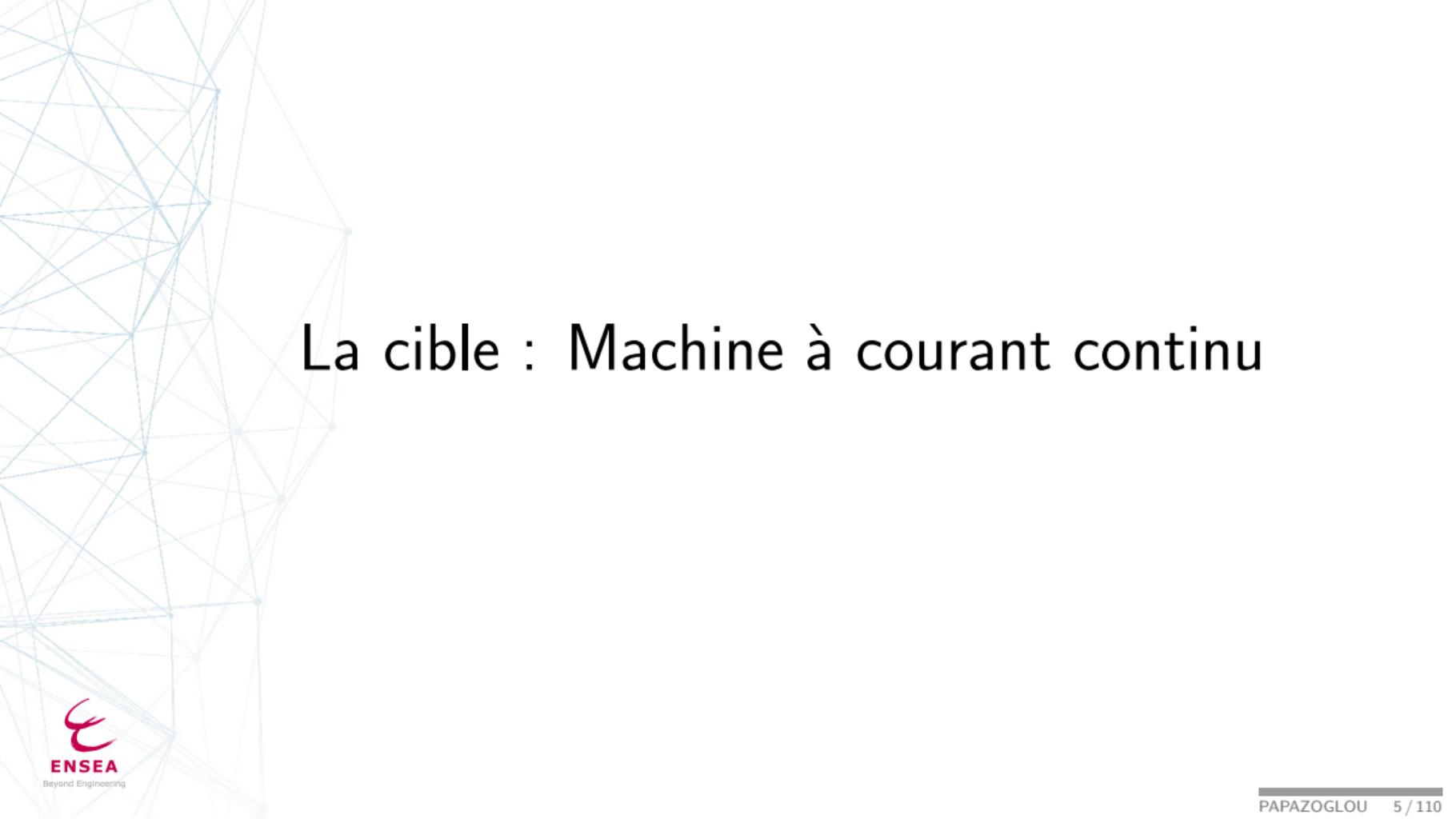
Ce cours a pour objectif de présenter de manière fonctionnelle les différents types d'actionneurs électriques pouvant être utilisés dans des applications industrielles embarquées.

- Les différents types de machines électriques (continu, alternatif, pas à pas, etc.).
- Les principaux capteurs associés : mesures de courant, vitesse, position.
- Modélisation des actionneurs électriques : ensembles formés de la machine, de son dispositif d'alimentation, des capteurs associés et de la commande.
- Réalisation d'un système de commande numérique directe et asservissement d'actionneur électrique par micro-controleur STM32.
- Les travaux pratiques mettront en œuvre la commande numérique pour moteur à courant continu avec asservissement.

Moyens :

- 14h de cours/TD
- 12h de TP

Séance 1 (Nicolas.P)	Séance 2 (Nicolas.P)	Séance 3 (Nicolas.P)	Séance 4 (Nicolas.P)	Séance 5 (Nicolas.P)	Séance 6 (Nicolas.P)	Séance 7 (Nicolas.P)	Séance 8 (Nicolas.P)
Présentation architecture micro-processeur, UART, Timer, PWM	Moteur pas à pas Commande et application	Commande de MCC Commande de hacheur temps morts	Commande complémentaire décalée	Capteurs des moteurs Analogiques (Courant, Hall)	Capteurs de moteurs numériques (roue codeuse)	Implémentation correcteur	Contrôle de moteur synchrone 6 step motor driver
Découverte de la carte, Clock, GPIO Application console UART	Moteur pas à pas	Génération de quatre PWM À partir du TIM 1	Génération de quatre PWM Complémentaire décalée	Capteur Analogiques Timer, ADC, DMA	Timer, Encoder interface mode	Test correcteur en temps réel	PWM 3 Commande de BLDC, MS



La cible : Machine à courant continu

La cible : Machine à courant continu

La machine à courant continu



La cible : Machine à courant continu

La machine à courant continu



$$u(t) = K\omega(t) + Ri(t)$$

$$C(t) = Ki(t)$$

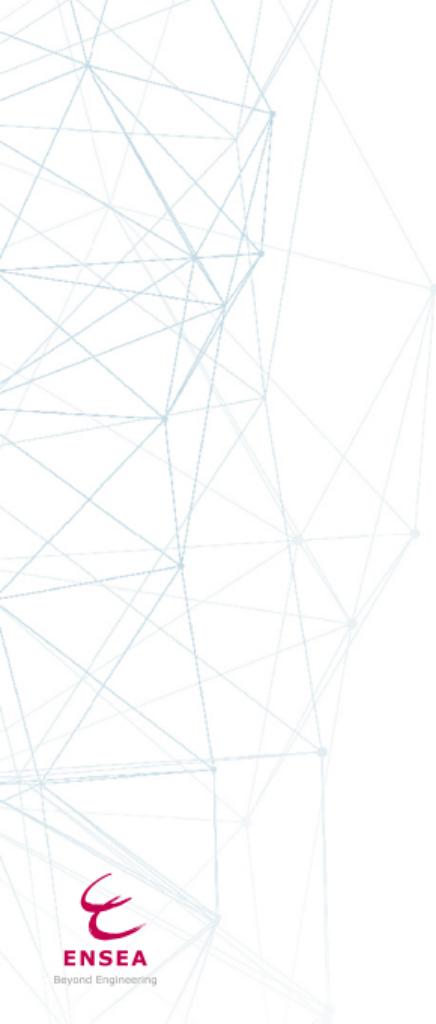
La cible : Machine à courant continu

La machine à courant continu



$$u(t) = K\omega(t) + Ri(t)$$
$$C(t) = Ki(t)$$

Pour contrôler un moteur, nous avons besoin de connaître ses grandeurs caractéristiques et asservir courant et tension.



Le micro-controleur

Capacité du micro-controleur

- Qu'avons nous besoin comme périphériques ?

Capacité du micro-controleur

- Qu'avons nous besoin comme périphériques ?
- Qu'avons nous besoin comme capacité de calcul ?

Capacité du micro-controleur

- Qu'avons nous besoin comme périphériques ?
- Qu'avons nous besoin comme capacité de calcul ?
- Comment l'interfacer avec l'homme ?

Micro-processeur



Kit P-NUCLEO-IHM03 :

- NUCLEO-G431RB board basé un micro-processeur STM32G431RB
- STSPIN830 driver pour moteur triphasé

STM32G431

Connectivity		Timers
3x SPI, 3x I ² C, 4x UxART		5x 16-bit timers
1x USB 2.0 FS, 1x USB-C PD3.0 (+PHY)	Arm® Cortex®-M4 Up to 170 MHz	2x 16-bit basic timer
1x CAN-FD	Floating Point Unit	2x 16-bit advanced motor control timers
2x I ² S half duplex, SAI	Memory Protection Unit	1x 32-bit timer
	Embedded Trace Macrocell	1x 16-bit LP timer
Accelerators	12-channel DMA + MUX	Analog
ART Accelerator™	Up to 128-Kbyte Flash memory / ECC Single Bank	2x 12-bit ADC
10-Kbyte CCM-SRAM	22-Kbyte SRAM	4x Comparators
Math Accelerators		4x DAC (2x buff + 2x non-buff)
Cordic (trigo...) Filtering		3x op-amps (PGA)
		1x temperature sensor
		Internal voltage reference

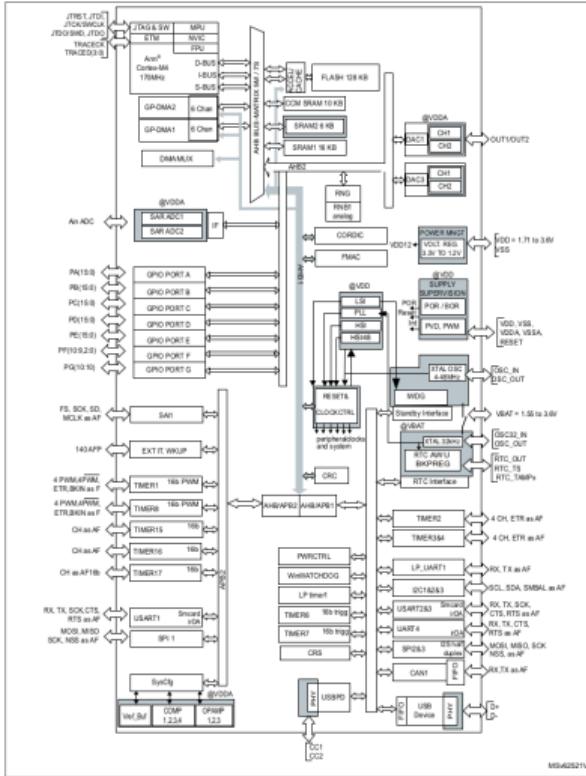
Documentation

Directement sur le site de ST : <https://www.st.com/>

- Documentation

Documentation

Figure 1. STM32G431x6/x8/xB block diagram

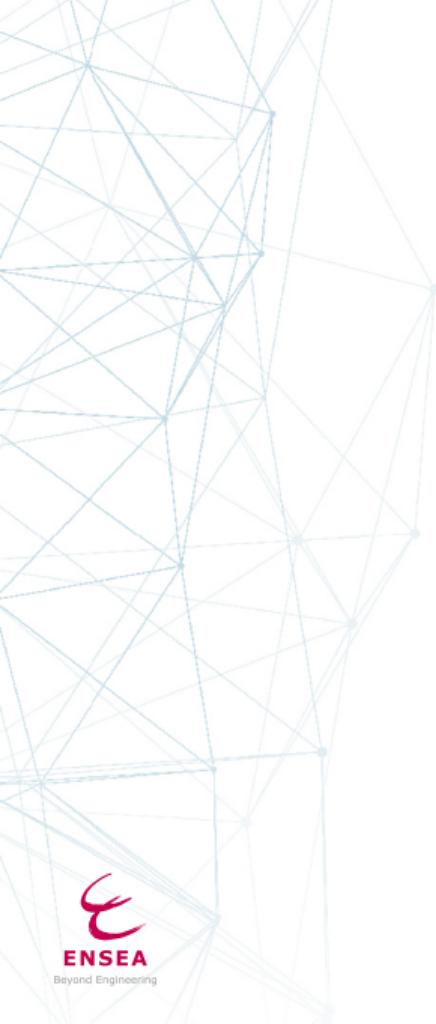




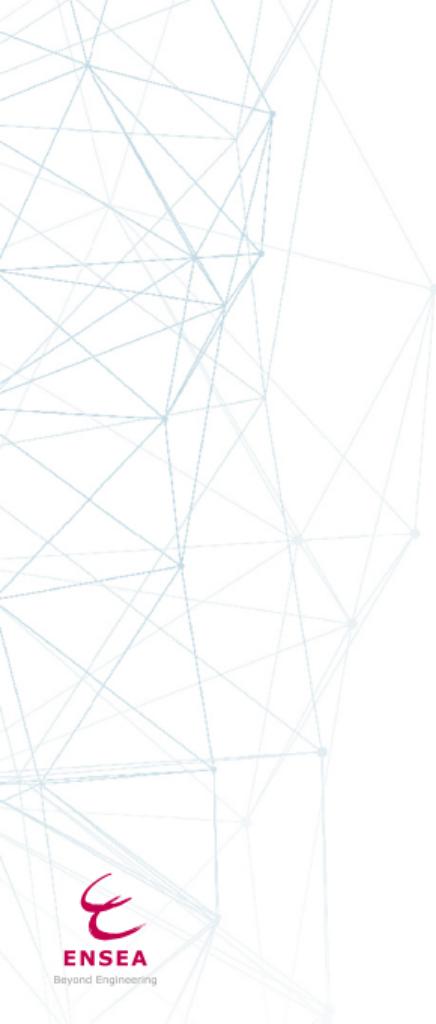
- IDE développé par ST avec des outils très puissants :
 - CubeMX
 - Librairie HAL
- Basé sur Eclipse
- Compilateur et débug intégré
- Disponible sous Windows, Mac OS et Linux

MOOC - STM32CubeIDE basics

MOOC by ST.



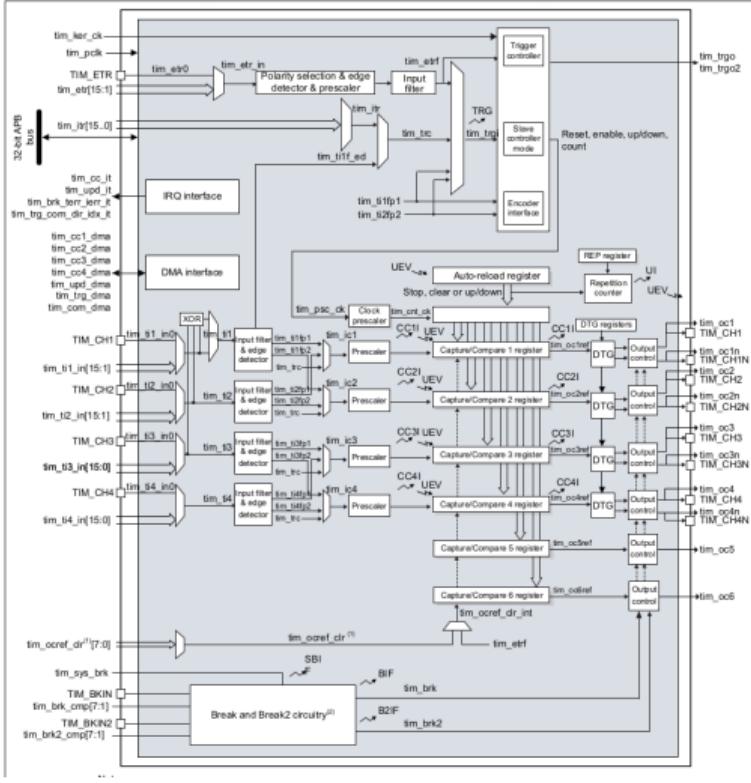
Timer, PWM, DMA et ADC



Timer et PWM

Timer

Figure 269. Advanced-control timer block diagram



Timer et PWM

28.3.13 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

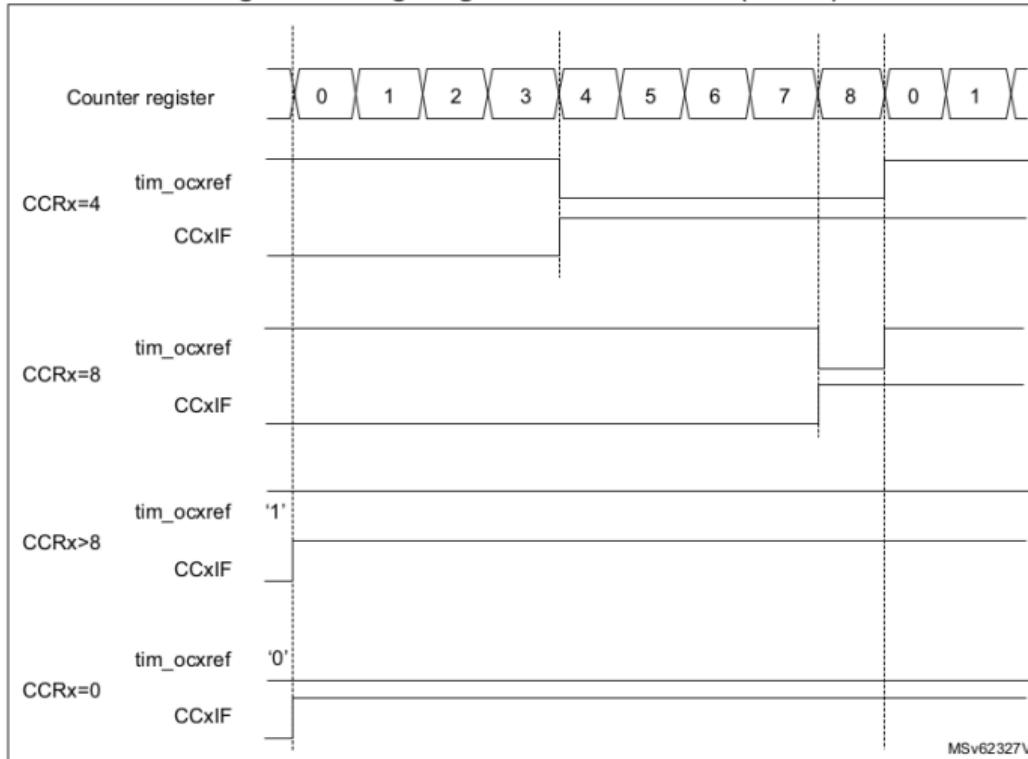
tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether $\text{TIMx_CCRx} \leq \text{TIMx_CNT}$ or $\text{TIMx_CNT} \leq \text{TIMx_CCRx}$ (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM commandé par un Timer

Figure 302. Edge-aligned PWM waveforms (ARR=8)



Il faut savoir quel type de signal vous voulez générer :

- Période de la PWM
- Résolution de la PWM
- Forme de la PWM (edge-aligned ou center-aligned)
- Symetric ou Asymmetric mode
- Etc...

Il faut savoir quel type de signal vous voulez générer :

- Période de la PWM
- Résolution de la PWM
- Forme de la PWM (edge-aligned ou center-aligned)
- Symetric ou Asymmetric mode
- Etc...

Ces contraintes imposeront d'autres paramètres :

- Fréquence de l'horloge en entrée du Timer
- Choix du timer (high résolution ou non)

Exercice en TD

Servo-moteur

→ | ← 1 ms



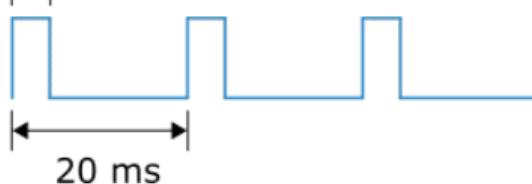
0 degrees

→ | ← 1.5 ms

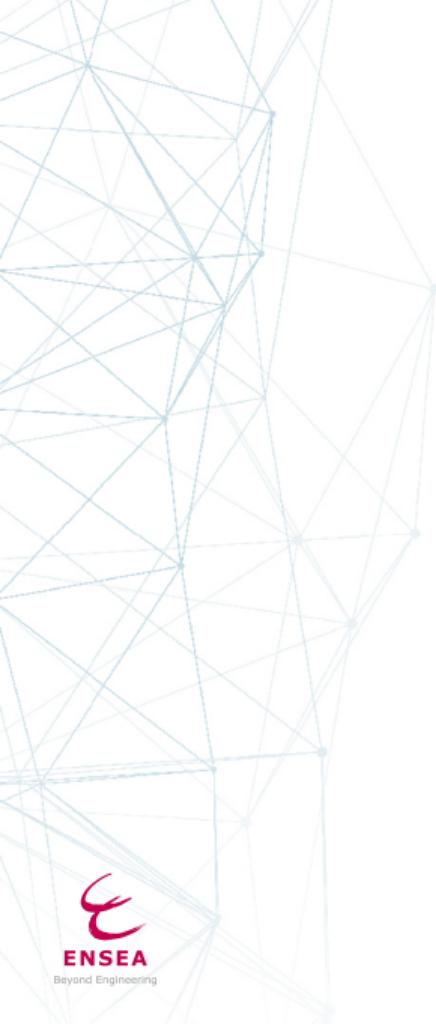


90 degrees

→ | ← 2 ms



180 degrees



ADC

Plusieurs façons de programmer le convertisseur analogique-numérique (ADC) :

- Par pooling,
- Par déclenchement software et interruption sur l'acquisition des données,
- Par déclenchement software et interruption par le DMA,
- Par déclenchement hardware (Timer) et interruption par le DMA.

ADC par pooling

Acquisition par pooling :

- Un déclenchement a lieu par le micro-processeur (soft)
- Après un certain temps....
- Le processeur va récupérer les valeurs converties par l'ADC (en espérant que la conversion ait eu le temps de finir)

ADC par pooling

Acquisition par pooling :

- Un déclenchement a lieu par le micro-processeur (soft)
- Après un certain temps....
- Le processeur va récupérer les valeurs converties par l'ADC (en espérant que la conversion ait eu le temps de finir)

Fonctions HAL utilisées :

- `HAL_StatusTypeDef HAL_ADCEx_Calibration_Start (ADC_HandleTypeDef * hadc, uint32_t SingleDiff)`
- `HAL_StatusTypeDef HAL_ADC_Start (ADC_HandleTypeDef * hadc)`
- `HAL_StatusTypeDef HAL_ADC_PollForConversion (ADC_HandleTypeDef * hadc, uint32_t Timeout)`
- `uint32_t HAL_ADC_GetValue (ADC_HandleTypeDef * hadc)`
- `void HAL_Delay (uint32_t Delay)`

Par déclenchement software, interruption en fin d'acquisition

Par déclenchement software et interruption sur l'acquisition des données :

- Un déclenchement à lieu par le micro-processeur (soft)
- Une fois l'acquisition réalisé par l'ADC, une interruption est mise en place,
- Le processeur va récupérer les valeurs converties par l'ADC

Par déclenchement software, interruption en fin d'acquisition

Par déclenchement software et interruption sur l'acquisition des données :

- Un déclenchement à lieu par le micro-processeur (soft)
- Une fois l'acquisition réalisé par l'ADC, une interruption est mise en place,
- Le processeur va récupérer les valeurs converties par l'ADC

Fonctions HAL utilisées :

- HAL_StatusTypeDef HAL_ADCEx_Calibration_Start (ADC_HandleTypeDef * hadc, uint32_t SingleDiff)
- uint32_t HAL_ADC_GetValue (ADC_HandleTypeDef * hadc)
- HAL_StatusTypeDef HAL_ADC_Start_IT (ADC_HandleTypeDef * hadc)
- void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc);

Par déclenchement software et interruption par le DMA

Par déclenchement software et interruption par le DMA

- Un déclenchement à lieu par le micro-processeur (soft)
- Une fois l'acquisition réalisé par l'ADC, les données sont transmises au DMA,
- Une fois le buffer rempli par le DMA, un interruption est générée.

Par déclenchement software et interruption par le DMA

Par déclenchement software et interruption par le DMA

- Un déclenchement à lieu par le micro-processeur (soft)
- Une fois l'acquisition réalisé par l'ADC, les données sont transmises au DMA,
- Une fois le buffer rempli par le DMA, un interruption est générée.

Fonctions HAL utilisées :

- HAL_StatusTypeDef HAL_ADCEx_Calibration_Start (ADC_HandleTypeDef * hadc, uint32_t SingleDiff)
- HAL_StatusTypeDef HAL_ADC_Start_DMA (ADC_HandleTypeDef * hadc, uint32_t * pData, uint32_t Length)
- void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)

Par déclenchement hardware (Timer) et interruption par le DMA

Par déclenchement hardware (Timer) et interruption par le DMA :

- Crédit d'une base de temps (Timer) avec *Event* régulier qui déclenche une conversion (ou succession de conversion),
- Une fois l'acquisition réalisé par l'ADC, les données sont transmises au DMA,
- Une fois le buffer rempli par le DMA, un interruption est générée.

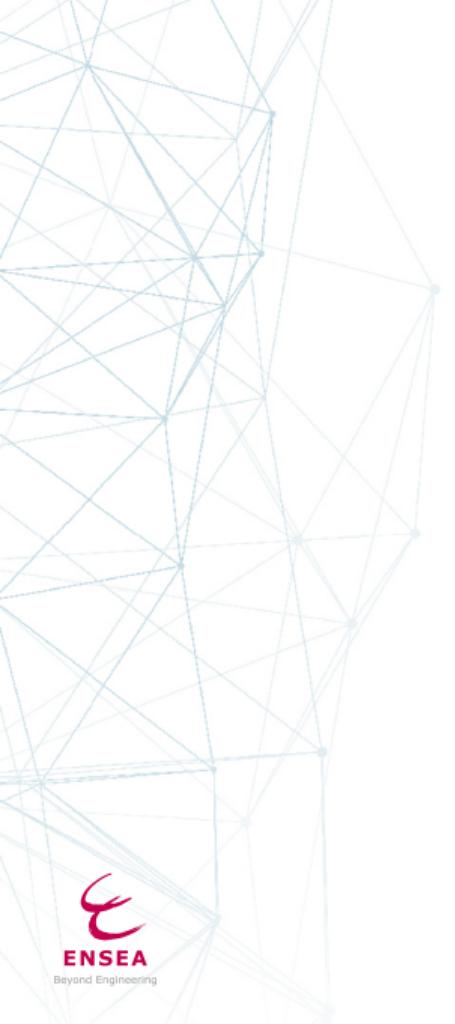
Par déclenchement hardware (Timer) et interruption par le DMA

Par déclenchement hardware (Timer) et interruption par le DMA :

- Crédation d'une base de temps (Timer) avec *Event* régulier qui déclenche une conversion (ou succession de conversion),
- Une fois l'acquisition réalisé par l'ADC, les données sont transmises au DMA,
- Une fois le buffer rempli par le DMA, un interruption est générée.

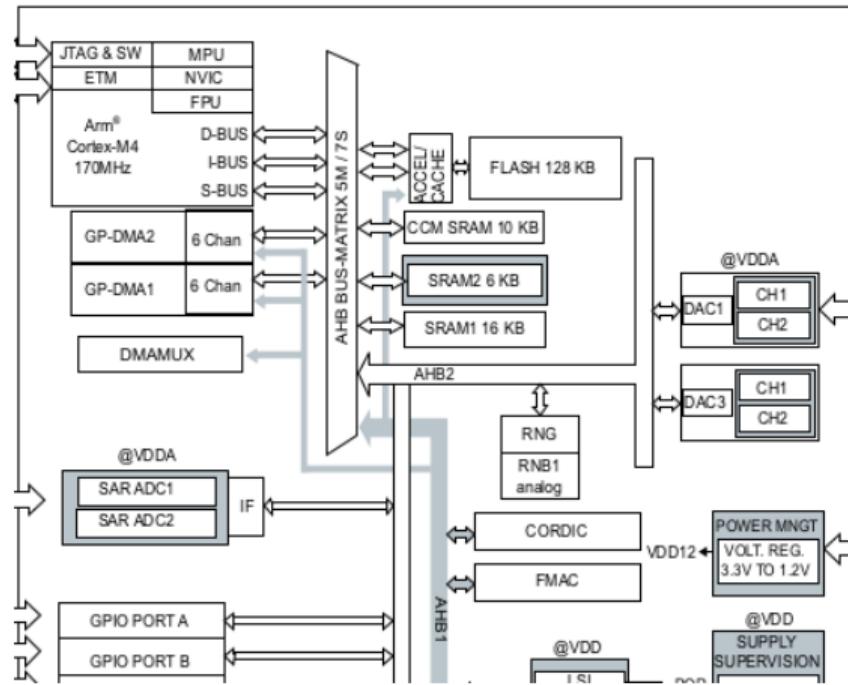
Fonctions HAL utilisées :

- HAL_StatusTypeDef HAL_ADCEx_Calibration_Start (ADC_HandleTypeDef * hadc, uint32_t SingleDiff)
- HAL_StatusTypeDef HAL_ADC_Start_DMA (ADC_HandleTypeDef * hadc, uint32_t * pData, uint32_t Length)
- HAL_StatusTypeDef HAL_TIM_Base_Start (TIM_HandleTypeDef * htim)



Direct Memory Access (DMA)

Direct Memory Access : Mécanisme qui permet l'accès direct à la mémoire vive sans passer par le processeur permettant ainsi une accélération assez importante des performances pour les bus d'entrées/sorties (E/S ou I/O). (Wikipedia)



Paramètres d'un DMA

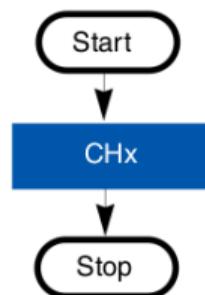
- Direction
 - de périphérique vers mémoire
 - de mémoire vers périphérique
 - de mémoire vers mémoire
- 2 modes :
 - Normal
 - Circulaire
- Incrément d'adresse
- Largeur des données (8, 16 ou 32 bits)



ADC, pour aller plus loin

Plusieurs modes de conversion

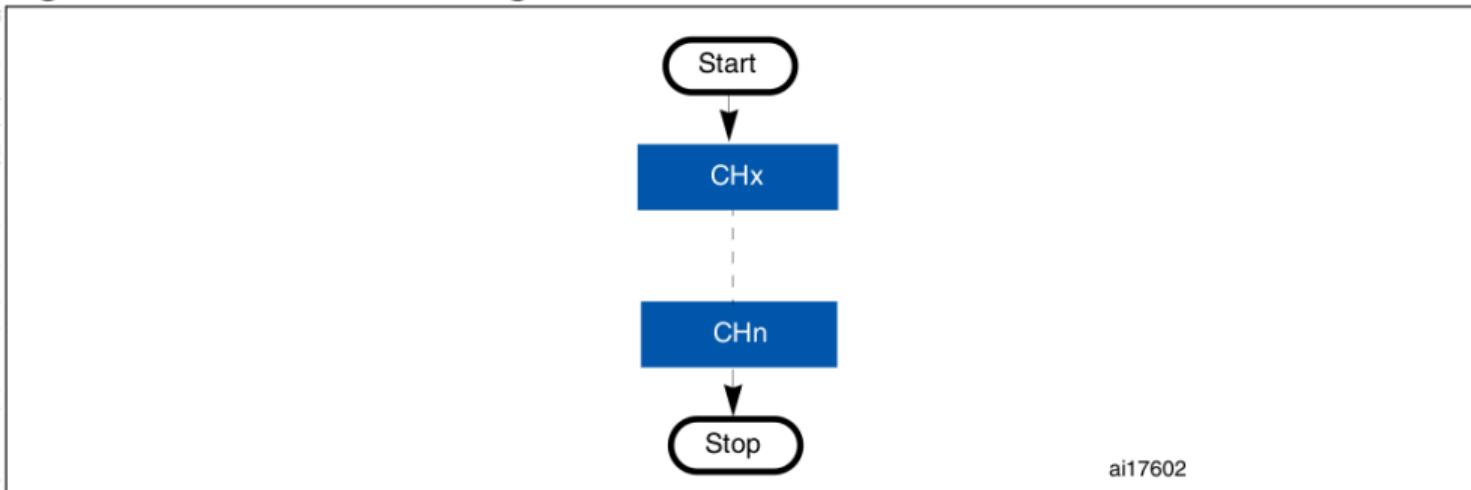
Figure 1. Single-channel, single conversion mode



ai17600

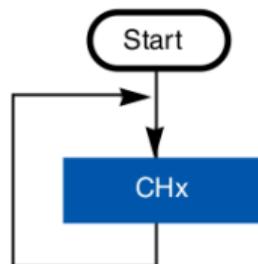
Plusieurs modes de conversion

Figure 3. Multichannel, single conversion mode



Plusieurs modes de conversion

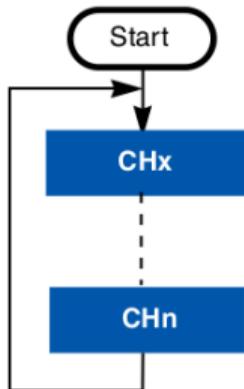
Figure 4. Single-channel, continuous conversion mode



ai17603

Plusieurs modes de conversion

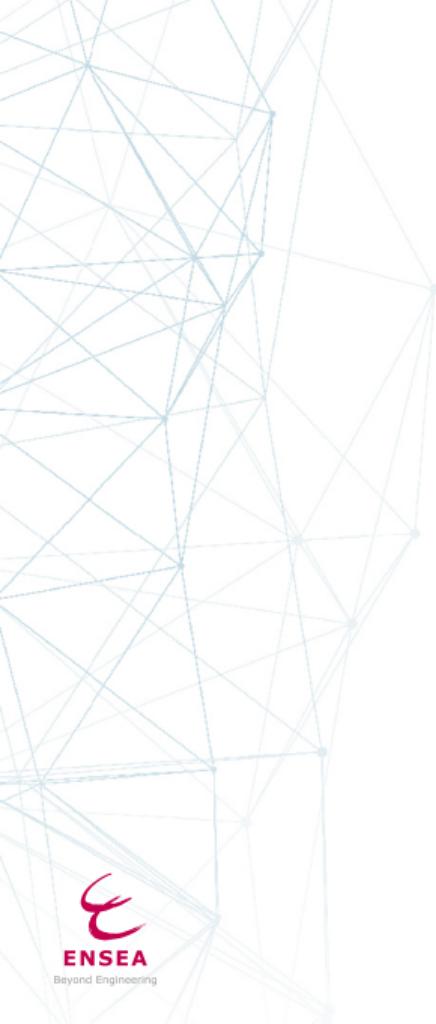
Figure 5. Multichannel, continuous conversion mode



ai17604

Autres paramètres à prendre en compte

- Temps d'acquisition et de conversion des données (voir horloge et pre-scaler)
- Mode Single Ended ou Differential

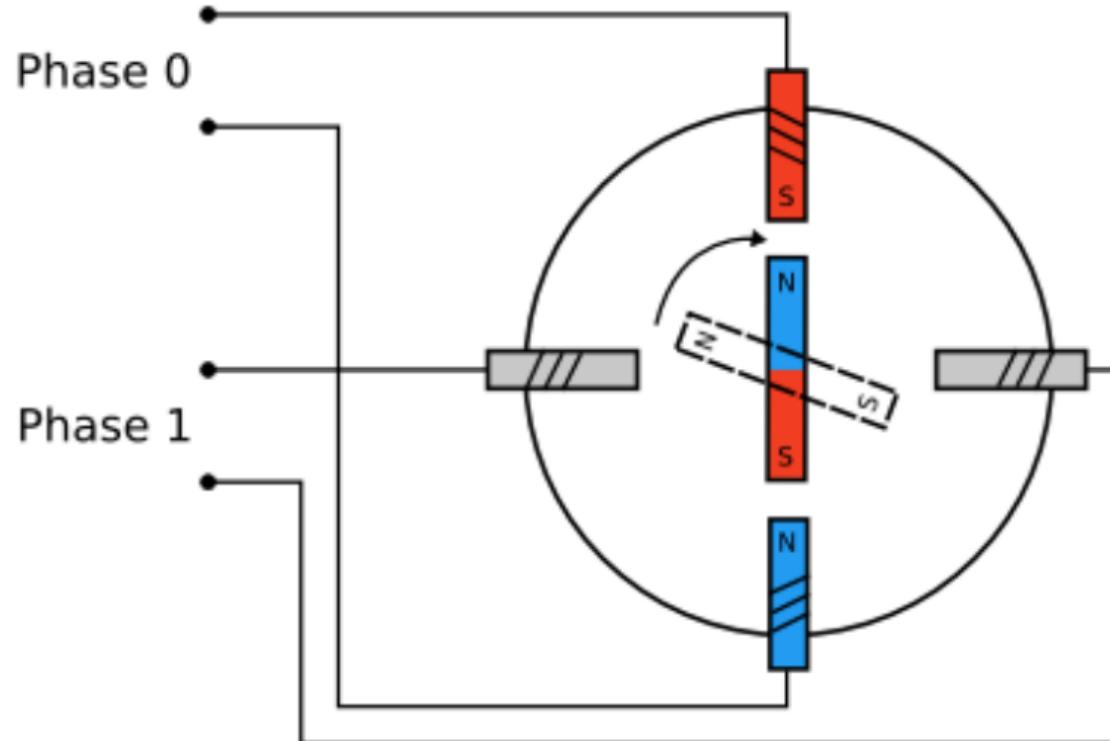


Moteur pas à pas

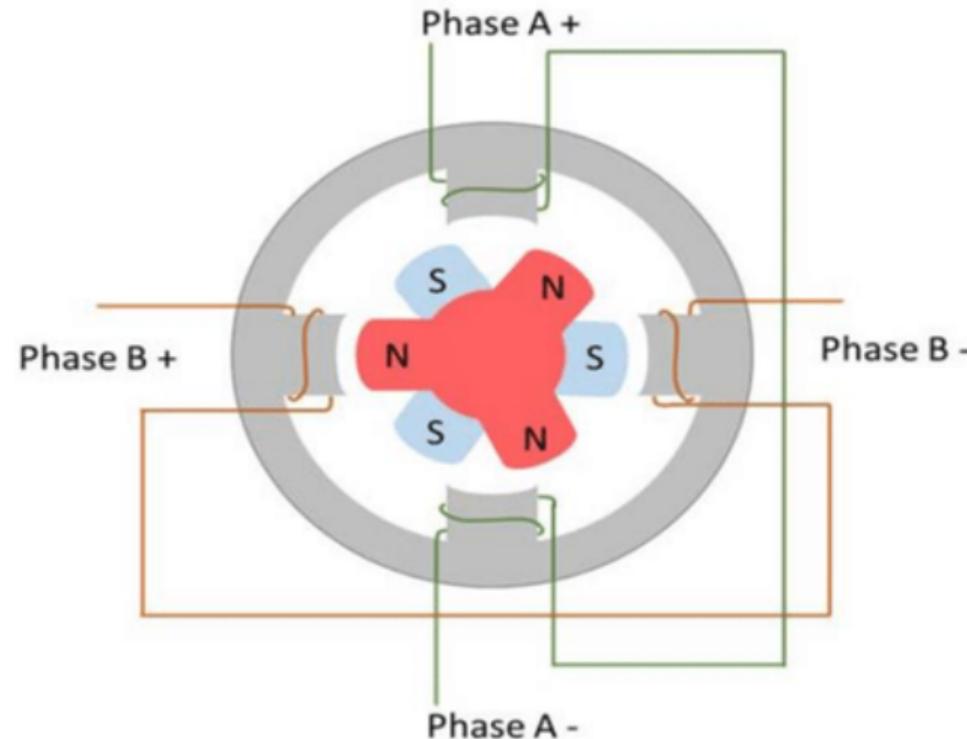


Principe du moteur pas à pas

Principe du moteur pas à pas unipolaire

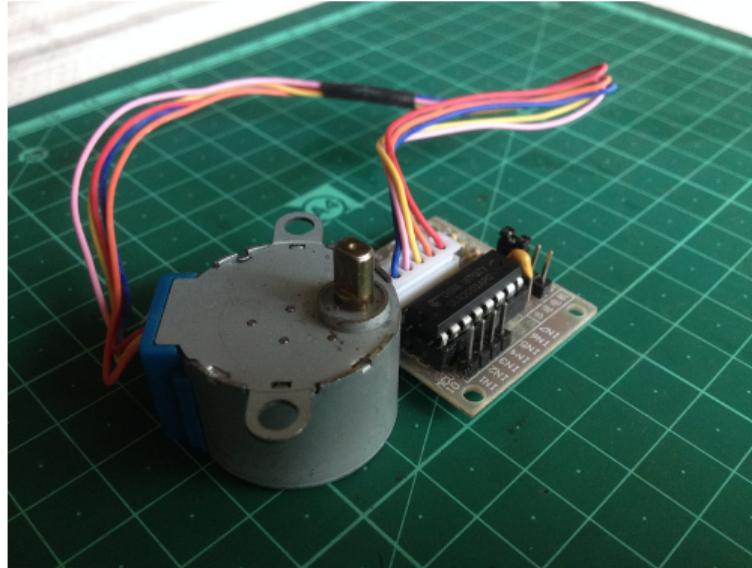


Principe du moteur pas à pas

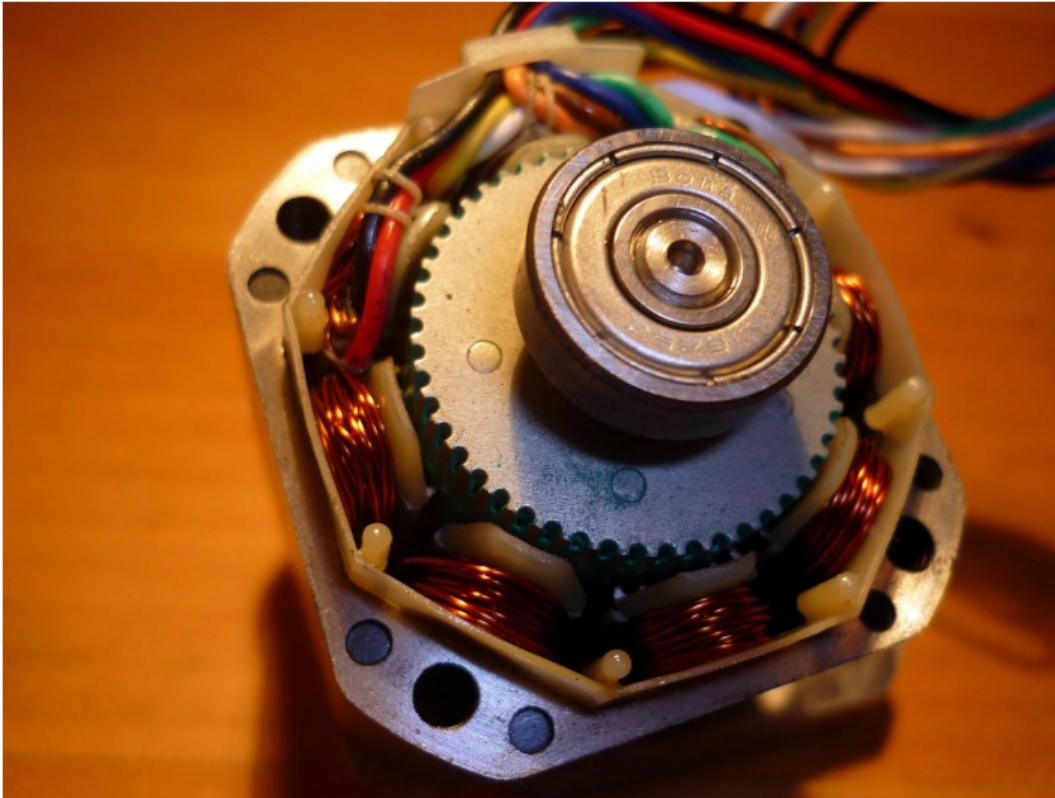


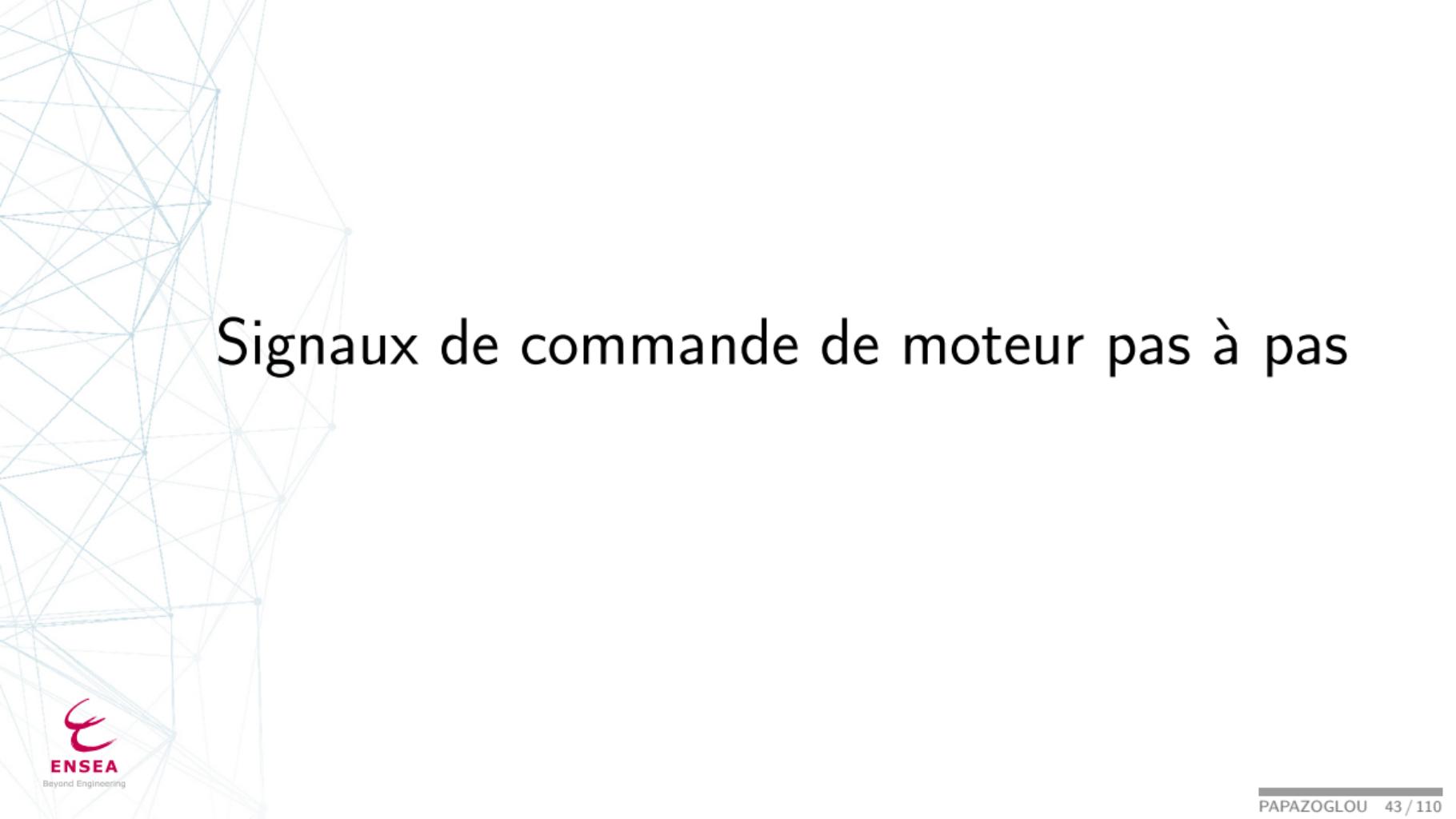
Exemple de matériel

- Driver ULN2003
- Moteur pas à pas 28BYJ48



Vue interne d'un moteur pas à pas





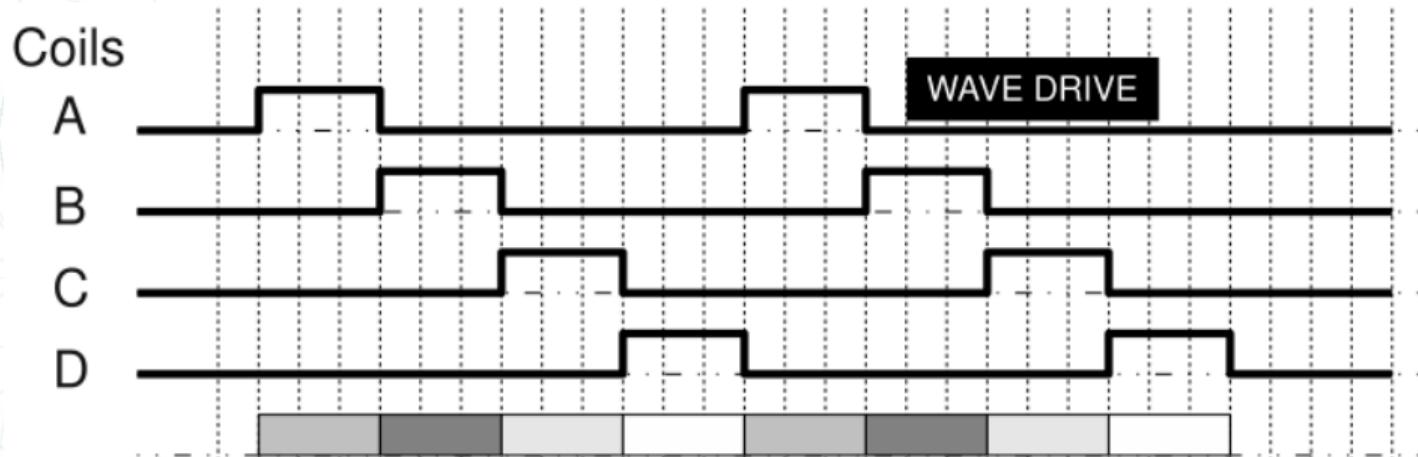
Signaux de commande de moteur pas à pas

Commande du moteur pas à pas

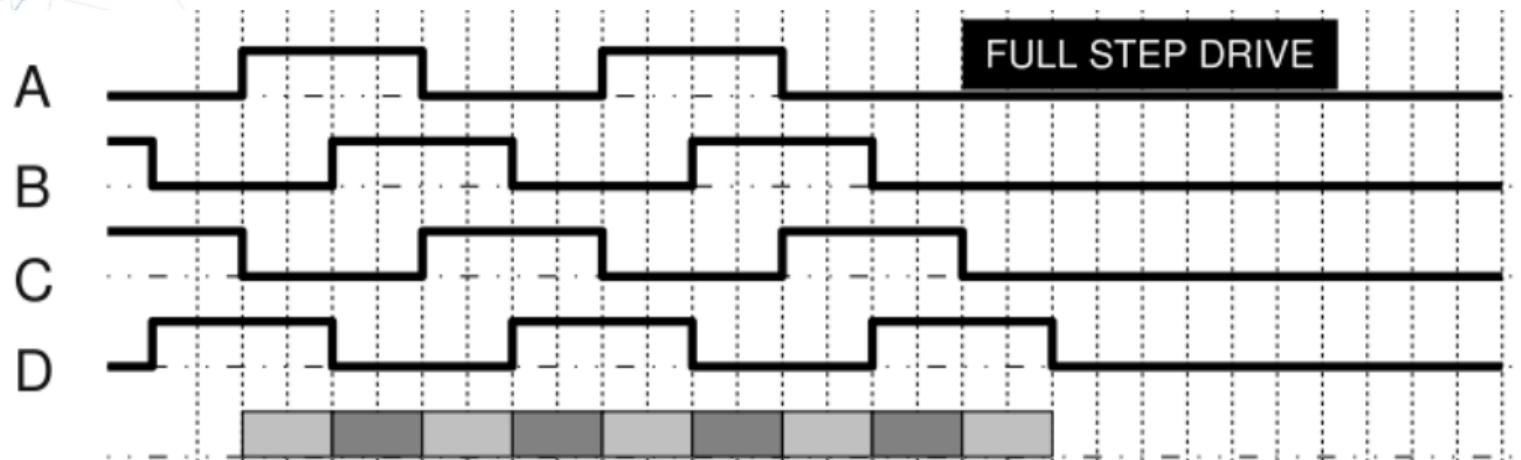
4 types de commande

- Wave drive
- Full step drive
- Half step drive
- Microstepping

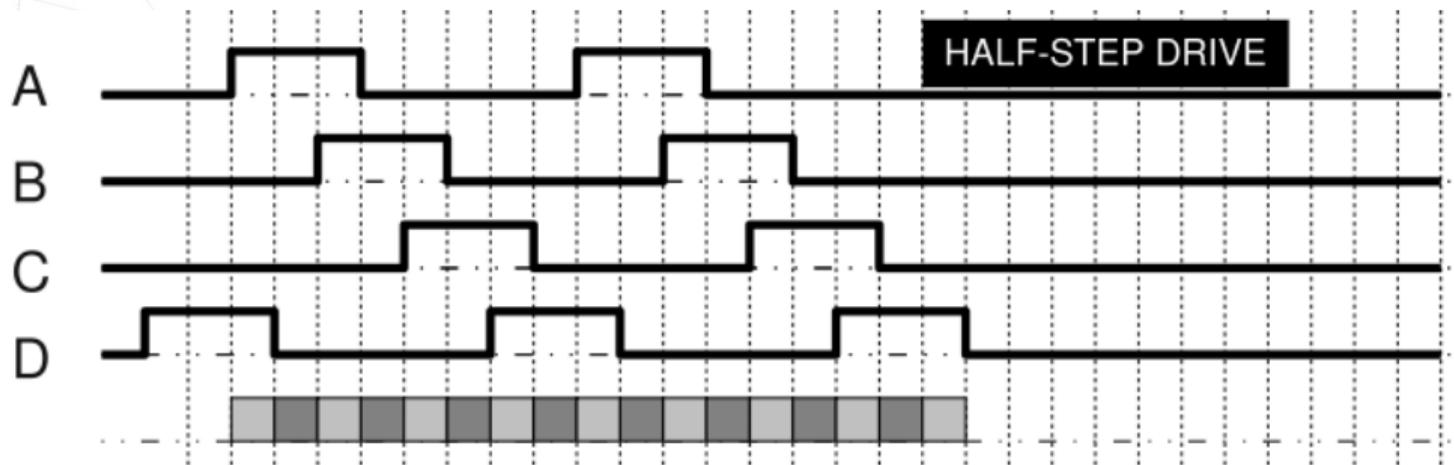
Commande du moteur pas à pas



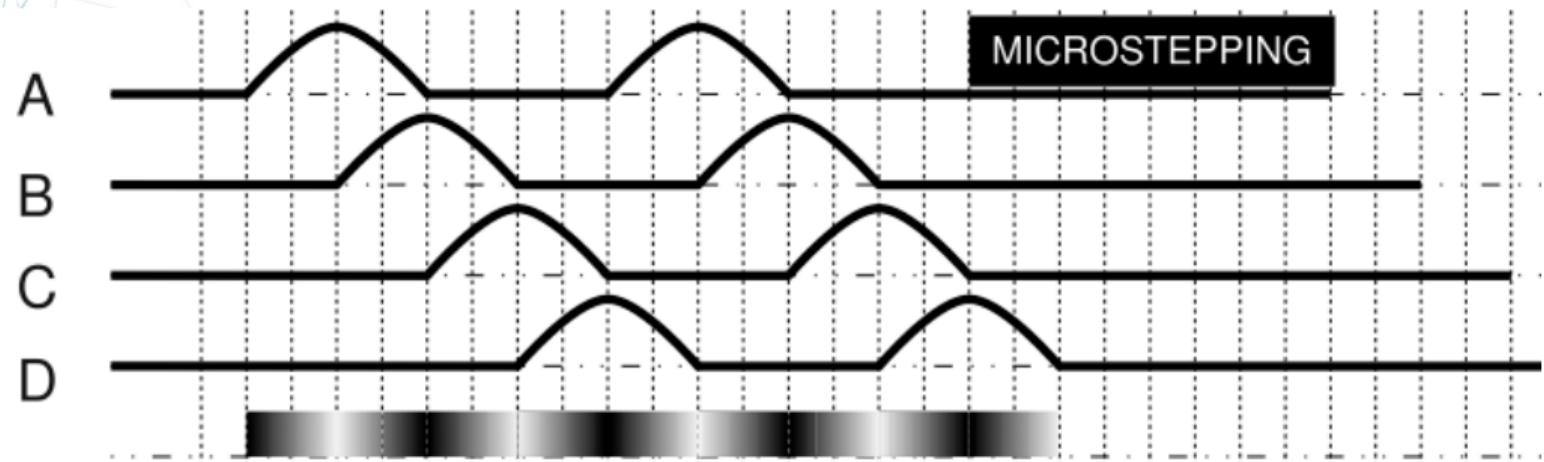
Commande du moteur pas à pas



Commande du moteur pas à pas



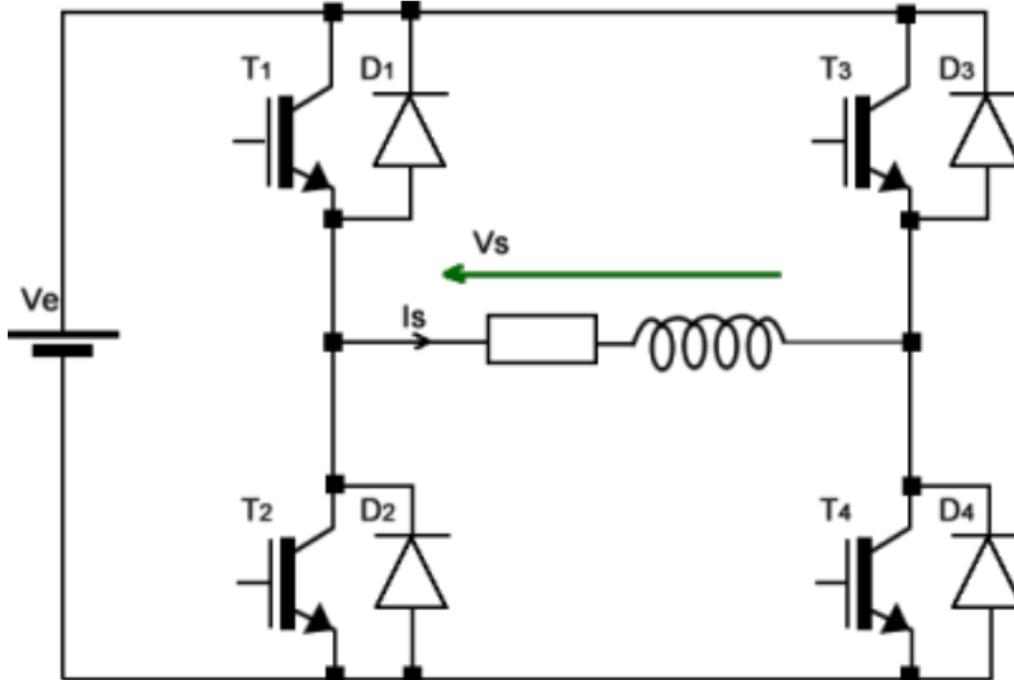
Commande du moteur pas à pas





Commande de MCC, commande de hacheur, gestion des temps morts

Hacheur 4 cadrants



Advanced-timer

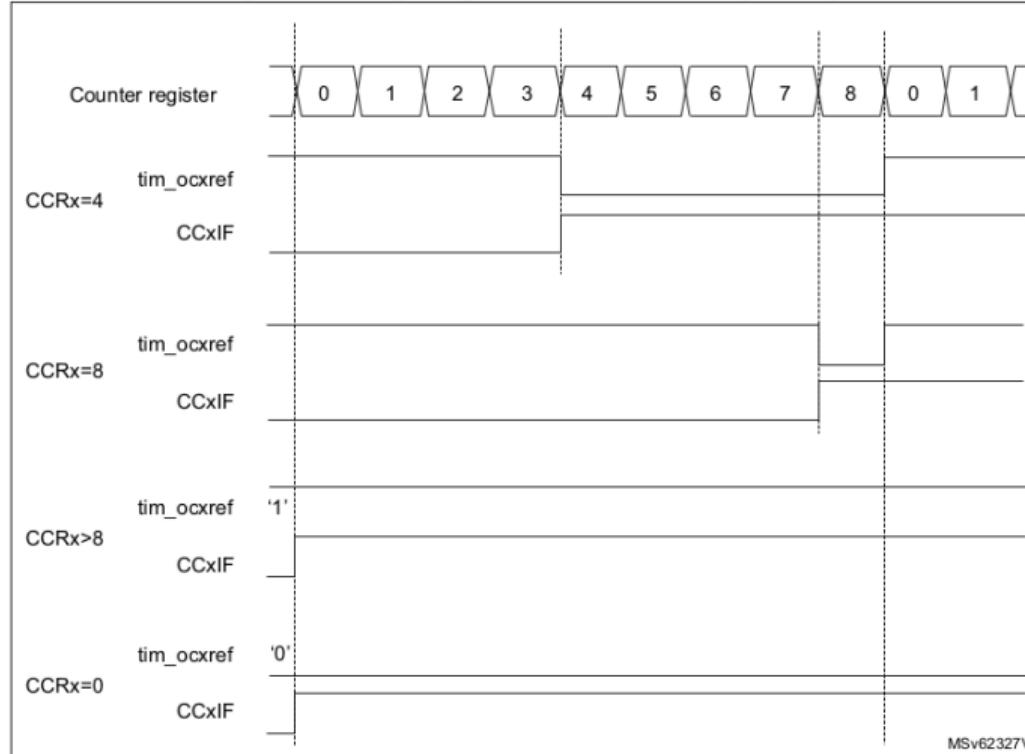
28.2 TIM1/TIM8/TIM20 main features

TIM1/TIM8/TIM20 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also "on the fly") the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
 - Input capture (but channels 5 and 6)
 - Output compare
 - PWM generation (Edge and Center-aligned Mode)
 - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer's output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
 - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
 - Trigger event (counter start, stop, initialization or count by internal/external trigger)
 - Input capture
 - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

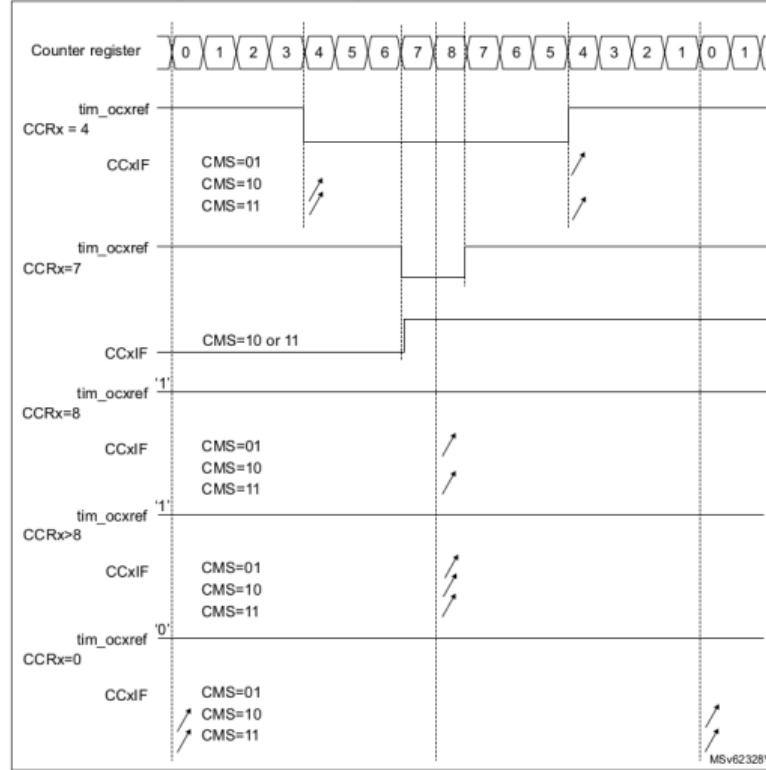
PWM edge aligned

Figure 302. Edge-aligned PWM waveforms (ARR=8)



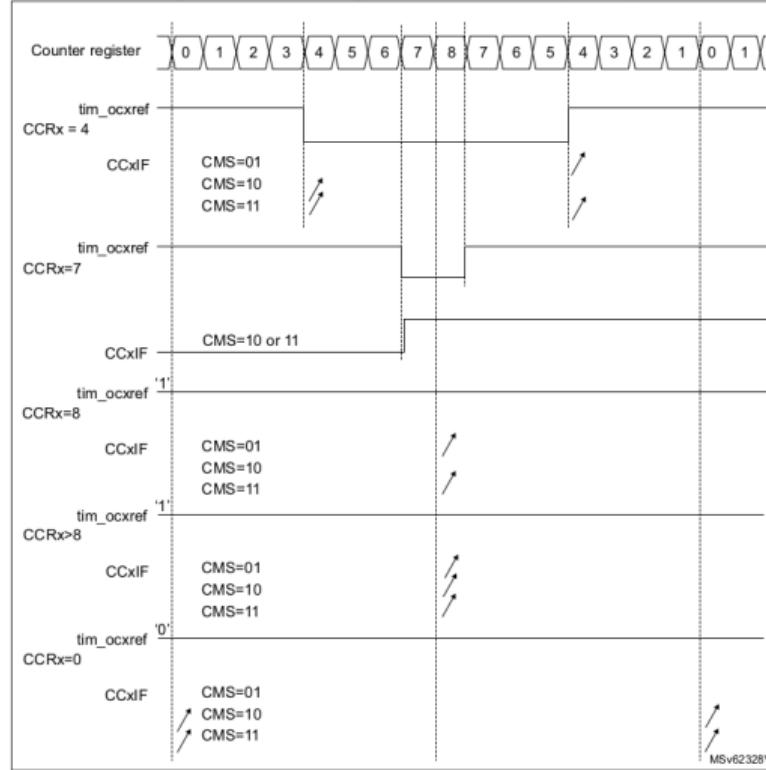
PWM center aligned

Figure 303. Center-aligned PWM waveforms (ARR=8)



PWM center aligned

Figure 303. Center-aligned PWM waveforms (ARR=8)



PWM Deadtime configuration

28.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8, 20)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		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
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 7:0 DTG[7:0]: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with t_{dtg}=t_{DTS}.

DTG[7:5]=10x => DT=(64+DTG[5:0])x t_{dtg} with T_{dtg}=2x t_{DTS}.

DTG[7:5]=110 => DT=(32+DTG[4:0])x t_{dtg} with T_{dtg}=8x t_{DTS}.

DTG[7:5]=111 => DT=(32+DTG[4:0])x t_{dtg} with T_{dtg}=16x t_{DTS}.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

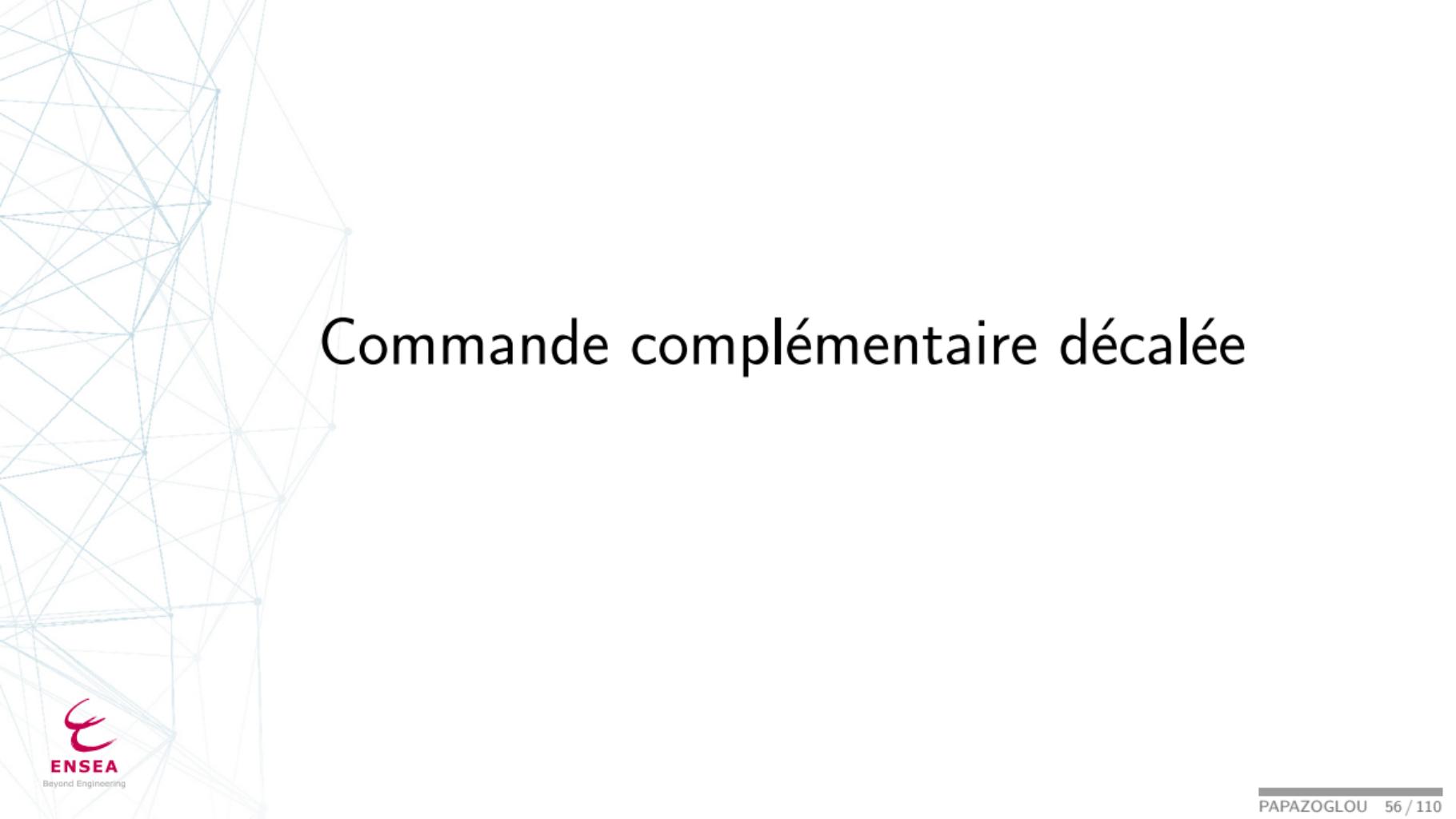
0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

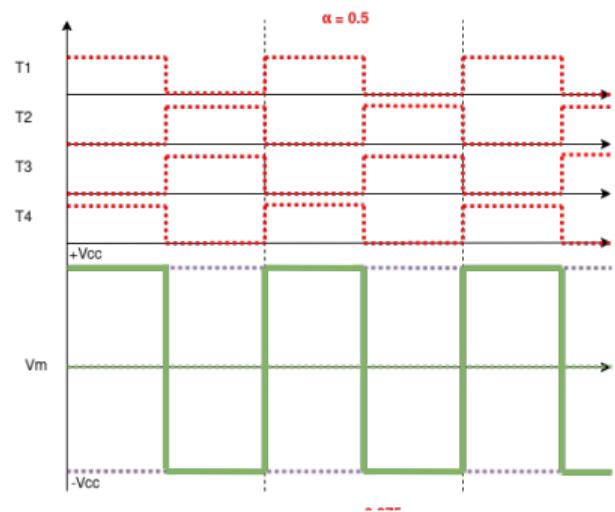
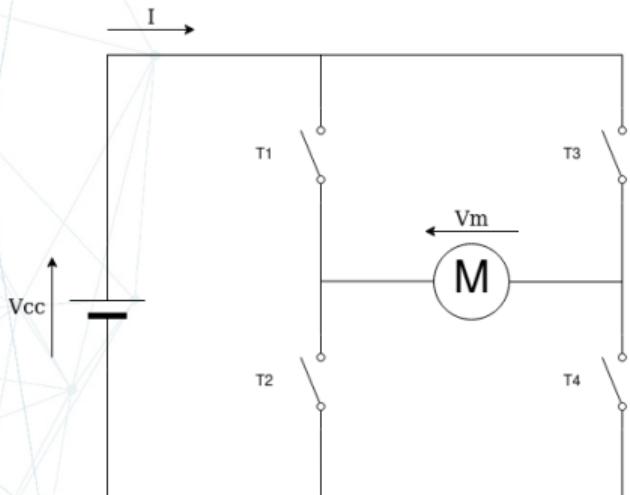
64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

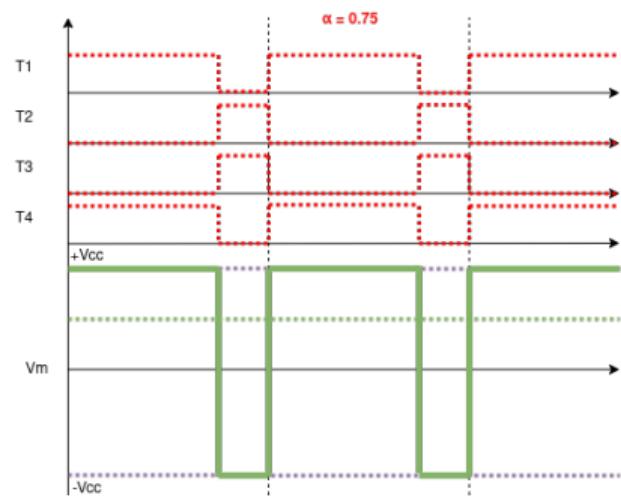
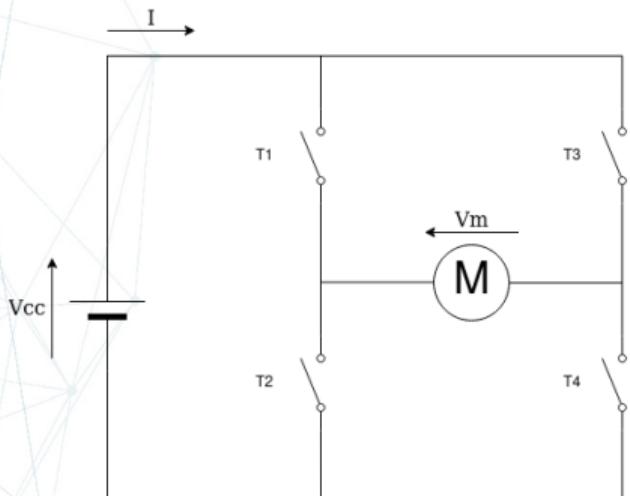


Commande complémentaire décalée

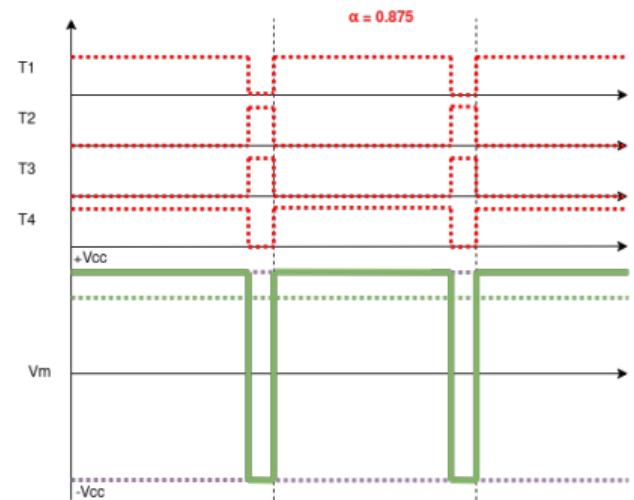
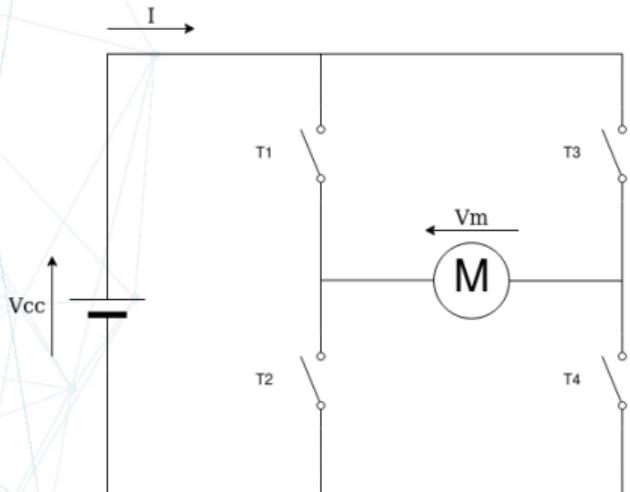
Commande classique : $\alpha = 0.5$



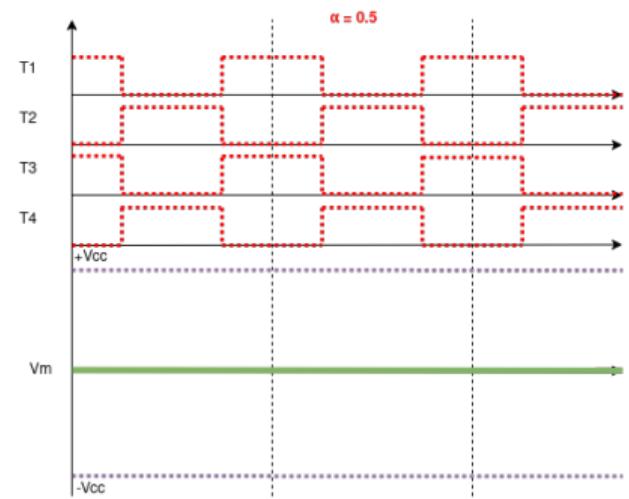
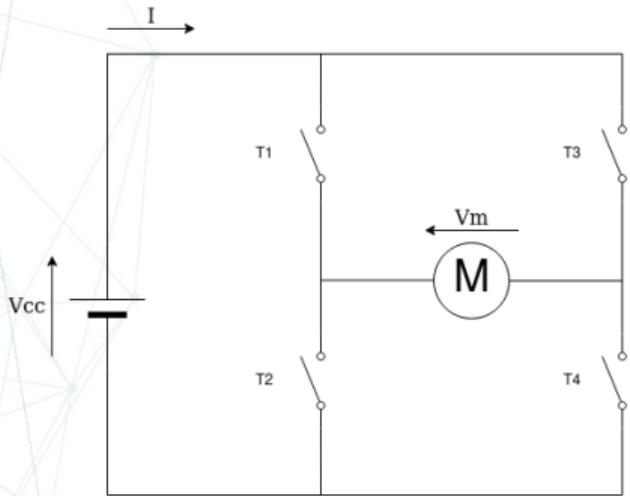
Commande classique : $\alpha = 0.75$



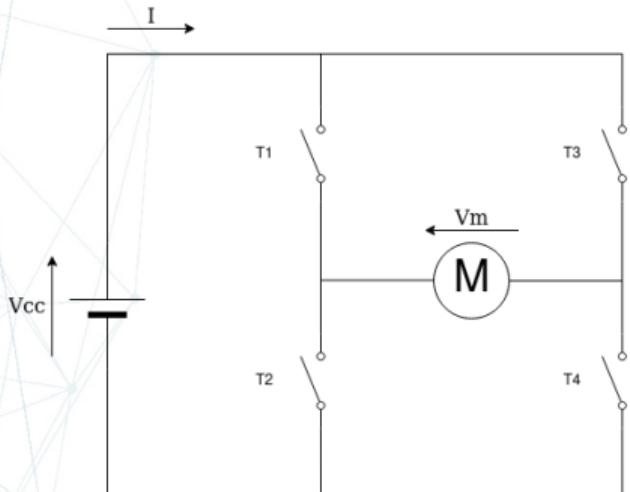
Commande classique : $\alpha = 0.875$



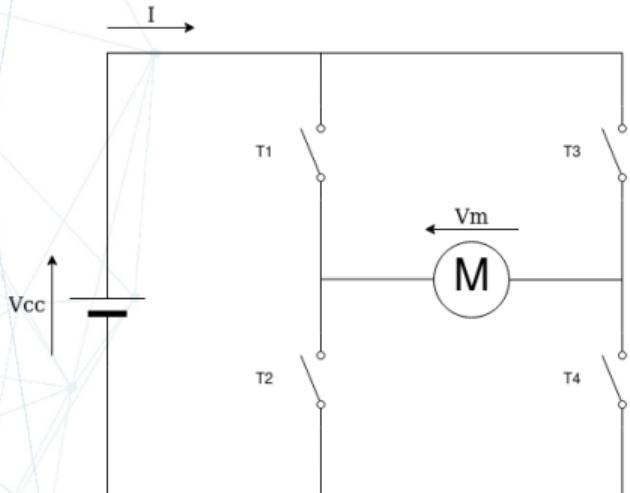
Commande complémentaire décalée : $\alpha = 0.5$



Commande complémentaire décalée : $\alpha = 0.75$



Commande complémentaire décalée : $\alpha = 0.875$





Cf. TD / TDm



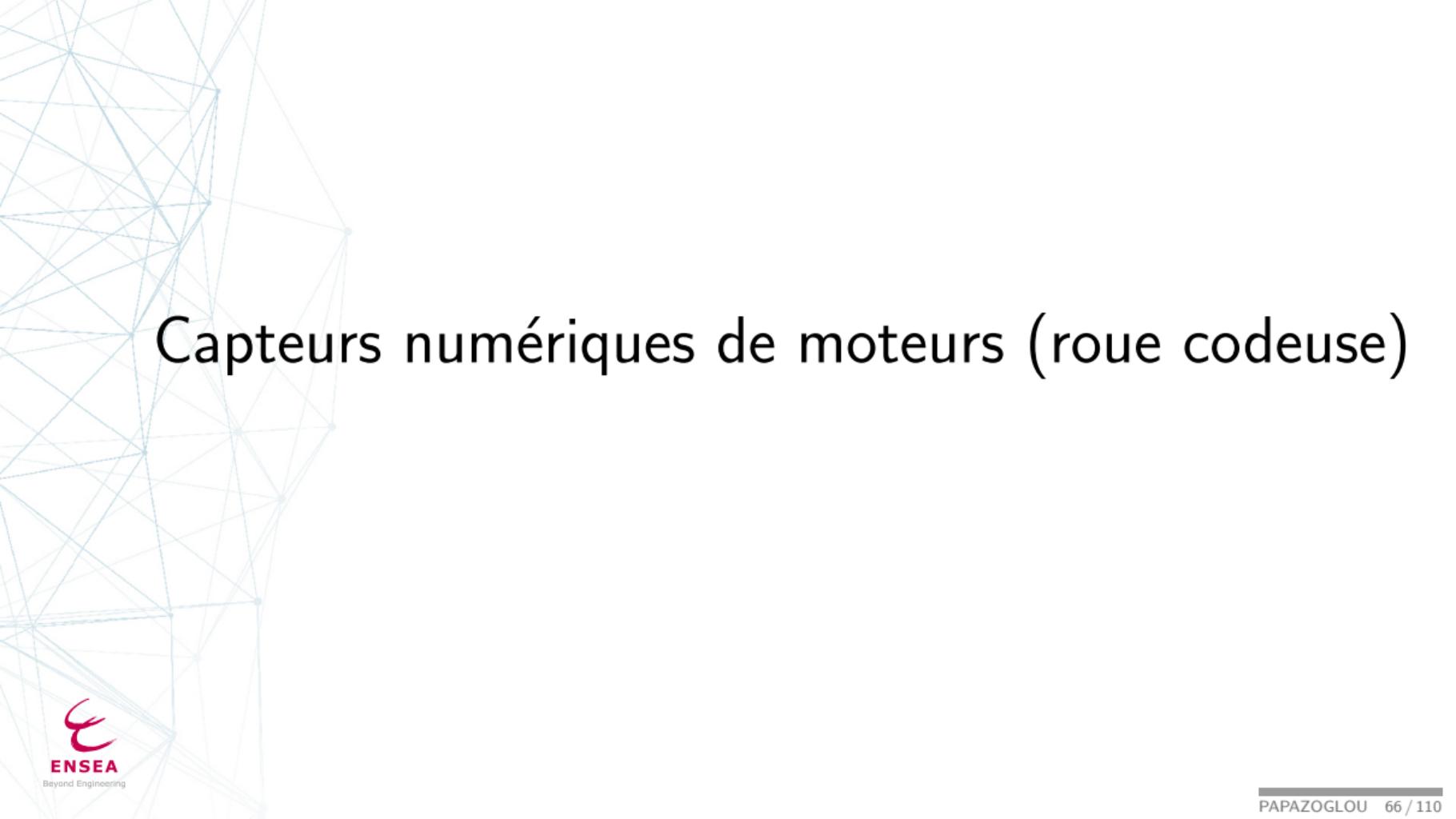
Capteurs des moteurs (Courant, Hall)

Pour l'asservissement des moteurs :

- Nécessite la mesure de plusieurs grandeurs analogiques :
 - Mesure de courant (capteur hall)
 - Mesure de vitesse (tachymétrie)
 - Capteur à effet hall (position moteur synchrone)

Pour l'asservissement des moteurs :

- Nécessite la mesure de plusieurs grandeurs analogiques :
 - Mesure de courant (capteur hall)
 - Mesure de vitesse (tachymétrie)
 - Capteur à effet hall (position moteur synchrone)
- A intervalle de temps régulier

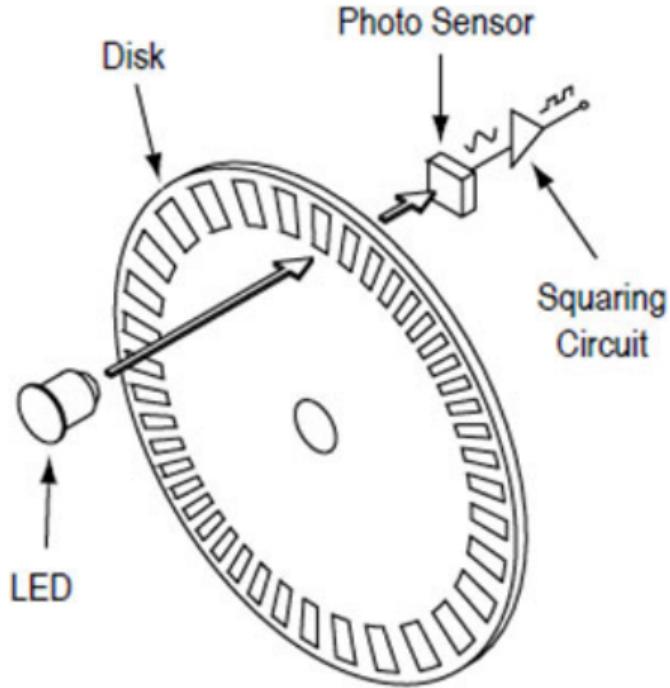


Capteurs numériques de moteurs (roue codeuse)

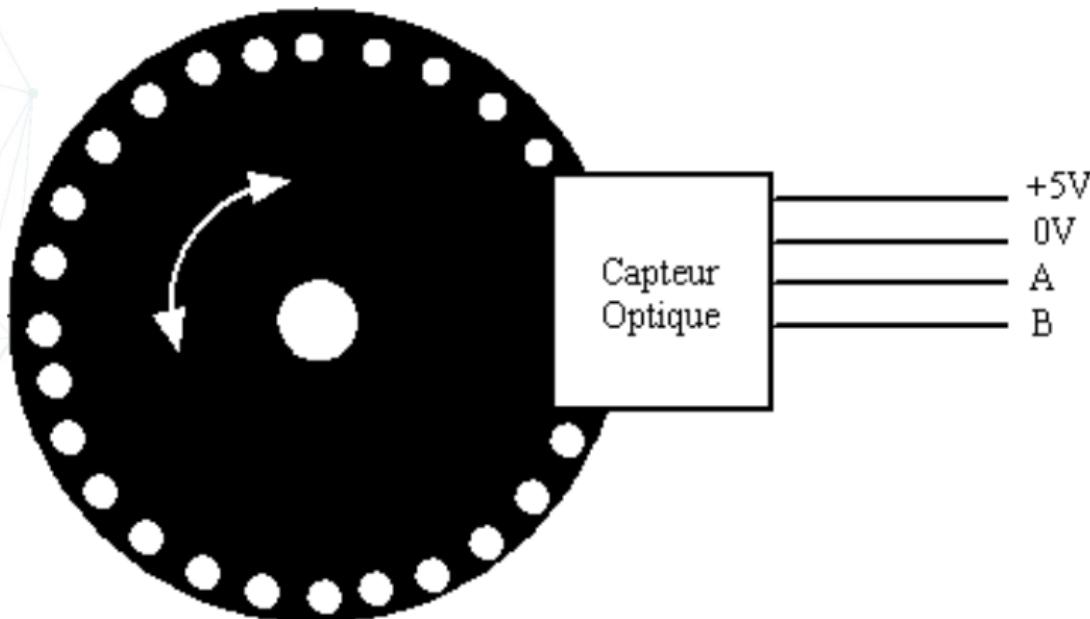
Objectif

- Asservissement du moteur en vitesse
- Asservissement du moteur en position
- Capteur optique, n'intervient pas sur le couple résistif du moteur
- Capteur "numérique", n'est pas sensible au bruit

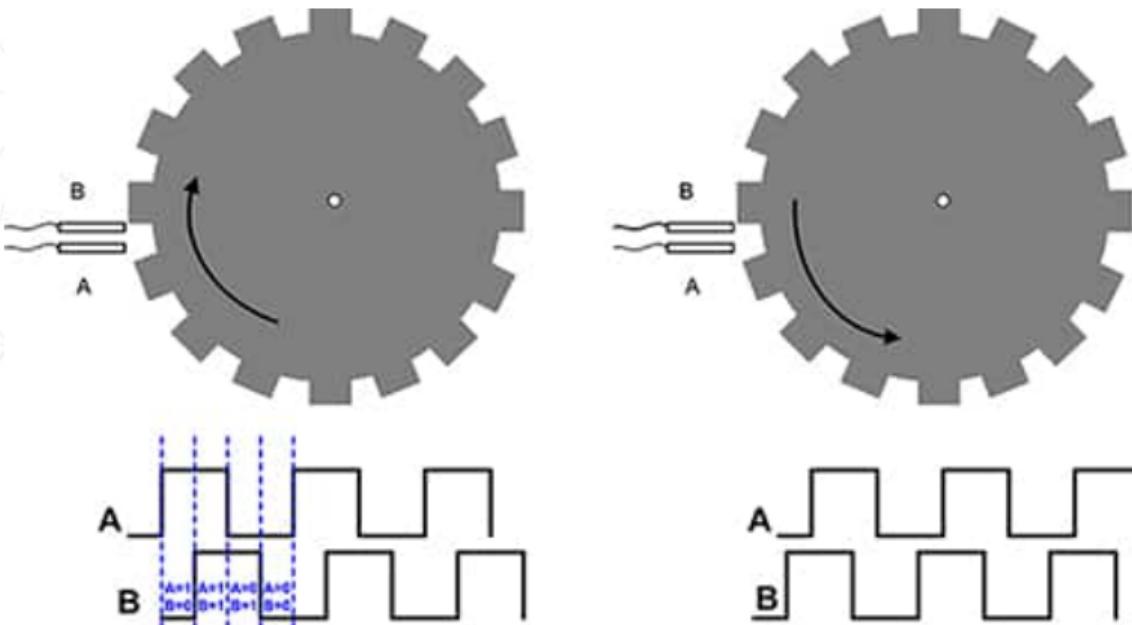
Principe mécanique



Principe mécanique



Signaux générés



Signaux générés

Caractéristiques des signaux générés :

- Très semblables à des PWM,
- Signaux en quadrature de phase,
- Il faut compter pour connaître la vitesse du moteur : $\omega(t) = \alpha \frac{\Delta n}{\Delta t}$

Comment compter...

2 cas :

- Vitesse rapide :
 - Il ne faut pas saturer le compteur
 - Utilisation d'un timer précis avec Δt fixé en amont
- Vitesse lente :
 - Si Δt trop petit, on ne verra aucun cran passé,
 - Δt non fixé, on lance un timer (pour compter le temps) et on attend d'arriver à un n fixé.
 - Le timer pourra être alors moins précis mais peut compter plus longtemps sans saturer (PSC plus élevé).

Il faut choisir la meilleure des deux solutions en fonction des vitesses mises en oeuvre dans votre moteur.

Comment compter...

Utilisation d'un timer pour compter les pas : Encoder Interface mode

La source n'est plus l'horloge mais 2 GPIO connectés à l'encoder.

28.3.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS='0001' in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS='0010' if it is counting on tim_ti2 edges only and SMS='0011' if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to [Table 260](#). The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1=tim_ti1 if not filtered and not inverted, tim_ti2fp2=tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Comment compter...

Plusieurs vitesses possibles

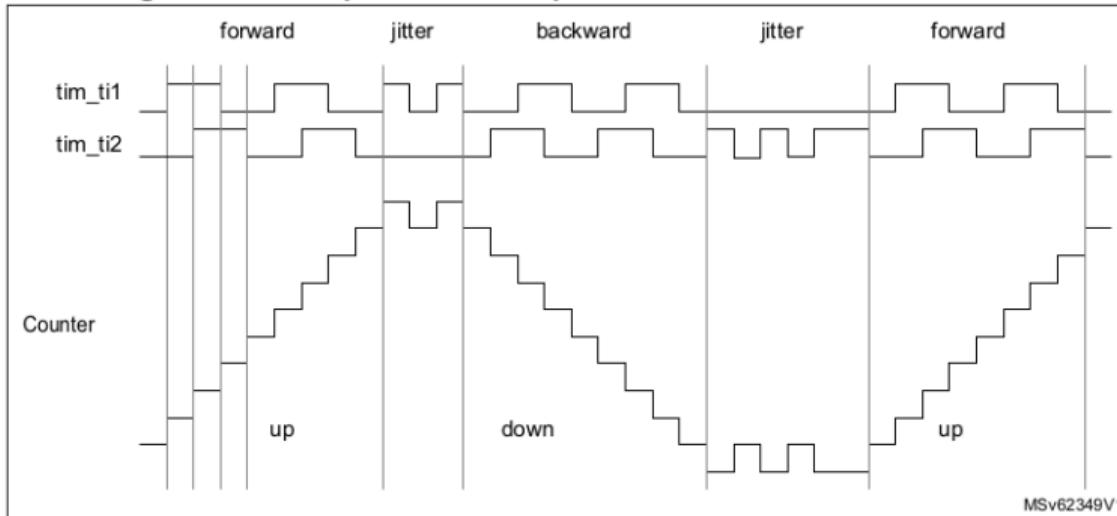
Table 260. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

Comment compter...

Compter et décompter

Figure 329. Example of counter operation in encoder interface mode.



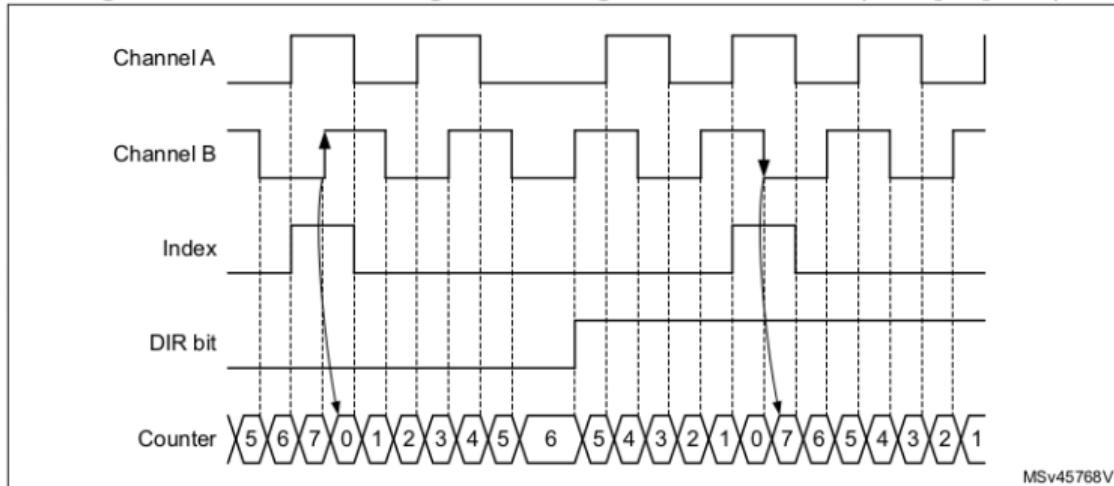
Remise à zéro du compteur

Deux possibilités Index : cran particulier sur le disque qui spécifie la position avec un angle nul.

- Sans index
 - On ne connaît que la position relative par rapport à celle au temps zéro.
 - On doit remettre la valeur du compteur à 0 manuellement
- Avec index
 - Une entrée spécifique est utilisée pour remettre automatiquement le compteur à 0

Remise à zéro du compteur

Figure 338. Counter reading with index gated on channel A (IPOS[1:0] = 11)



Possibilier de gate sur l'index sur la channel A ou B, ou les deux, ou sans gate

Pour aller plus loin...



Test sur un codeur rotatif manuel



Les signaux générés sont les mêmes.

Test sur un codeur rotatif manuel

TIM4 Mode and Configuration

Mode

- Slave Mode: Disable
- Trigger Source: Disable
- Internal Clock
- Channel1: Disable
- Channel2: Disable
- Channel3: Disable
- Channel4: Disable
- Combined Channels: Encoder Mode
- XOR activation
- One Pulse Mode

Configuration

Reset Configuration

NVIC Settings DMA Settings GPIO Settings
 Parameter Settings User Constants

Configure the below parameters :

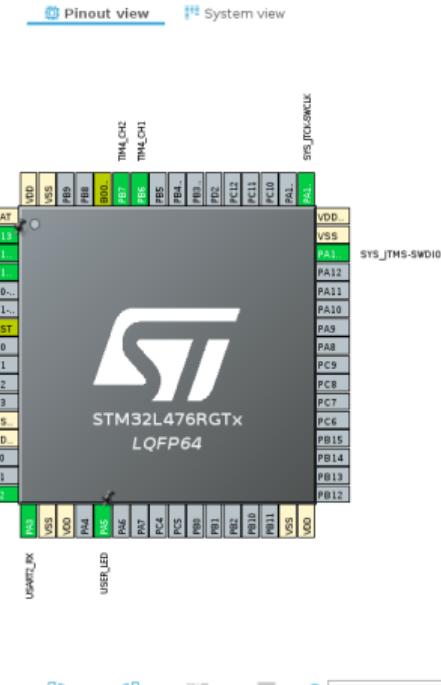
Search (Ctrl+F)

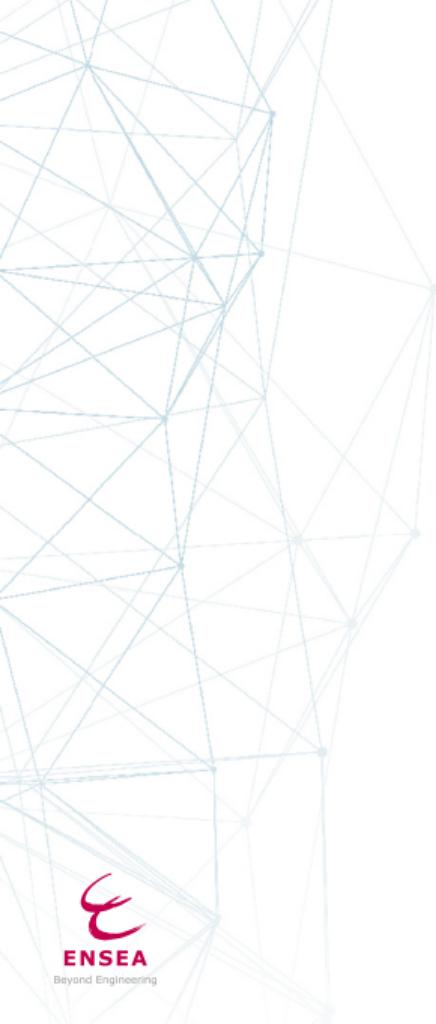
Counter Settings

- Prescaler (PSC - ... 0)
- Counter Mode Up
- Counter Period (A... 65535)
- Internal Clock Divi... No Division
- auto-reload preload Disable

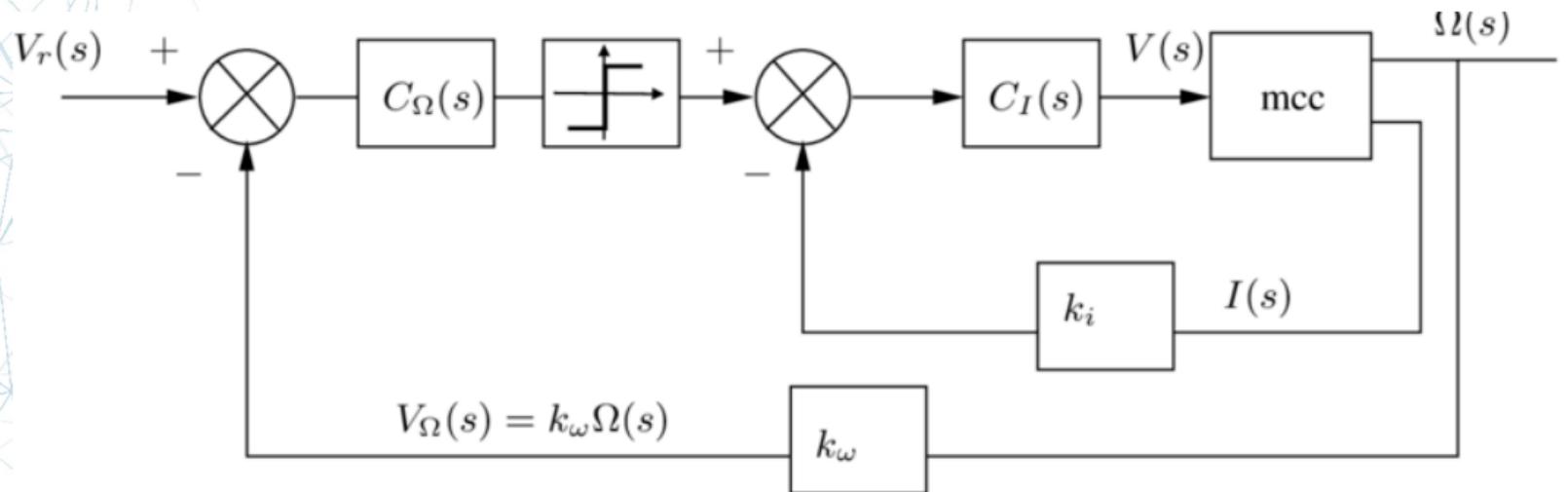
Trigger Output (TRGO) P...

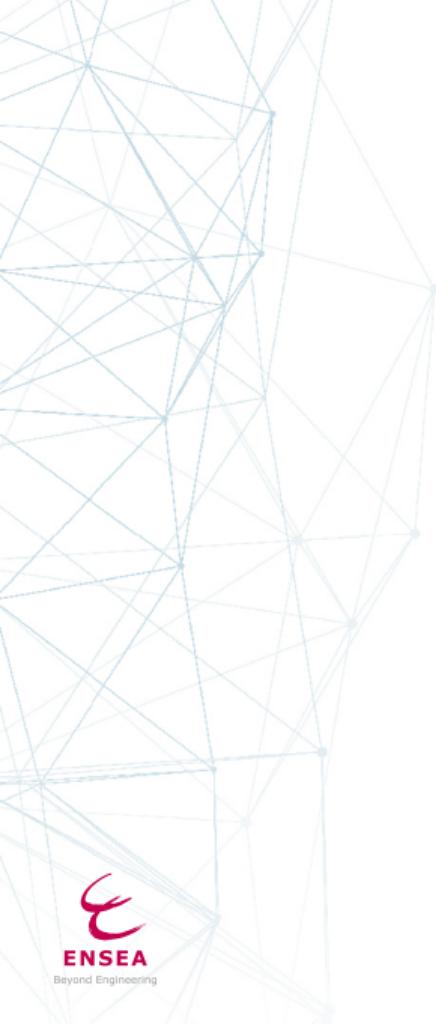
Master/Slave Mod... Disable (Trigger input effec...





Implémentation Correcteur





Transformée bilinéaire

Objectifs

Partir de l'asservissement analogique pour le transformer en asservissement numérique en minimisant l'erreur due à la transformation.

Transformation bilinéaire

L'opérateur d'intégration permet de définir une correspondance entre domaine numérique et domaine analogique. En fait,

- En analogique, le filtre intégrateur est $H_A(p) = \frac{1}{p}$
- Donc, dans le domaine temporelle, on a

$$y(t_1) - y(t_0) = \int_{t_0}^{t_1} x(\tau)d\tau \text{ avec } x(t) \text{ le signal dérivé de } y(t)$$

- Le pont entre le domaine analogique et domaine numérique est l'approximation de l'intégrale par la méthode des trapèzes :

$$\int_{t_0}^{t_1} x(\tau)d\tau \sim (t_1 - t_0) \frac{x(t_1) + x(t_0)}{2}$$

Transformation bilinéaire

- Pour $t_1 = nT_e$ et $t_0 = (n - 1)T_e$, avec T_e est la période d'échantillonnage, on a

$$y[n] - y[n - 1] = \frac{T_e}{2} (x[n] + x[n - 1]);$$

- Appliquons la transformée en \mathcal{Z} de la relation précédente, on trouve

$$H_N(z) = \frac{Y(z)}{X(z)} = \frac{T_e}{2} \frac{1 + z^{-1}}{1 - z^{-1}};$$

- Donc, par comparaison, on a :

$$\frac{1}{p} = \frac{T_e}{2} \frac{1 + z^{-1}}{1 - z^{-1}} \iff z = \frac{1 + \frac{T_e}{2}p}{1 - \frac{T_e}{2}p}.$$

Transformation bilinéaire

Donc le passage du plan p au plan Z peut se faire par la correspondance suivante:

$$z = \frac{1 + \frac{T_e}{2}p}{1 - \frac{T_e}{2}p}.$$

Ainsi, l'homologue numérique d'un filtre analogique $H_A(p)$ est obtenue par la relation suivante

$$H_N(z) = H_A \left(p = \frac{2}{T_e} \frac{1 - z^{-1}}{1 + z^{-1}} \right).$$

Impact dans le domaine fréquentiel

Pour $z = e^{j2\pi \frac{f_N}{f_e}}$ et $p = j2\pi f_A$, on a :

$$\begin{aligned} p &= \frac{2}{T_e} \frac{1 - z^{-1}}{1 + z^{-1}} \\ j2\pi f_A &= \frac{2}{T_e} \frac{1 - e^{-j2\pi f_N T_e}}{1 + e^{-j2\pi f_N T_e}} \\ j\pi f_A &= \frac{1}{T_e} \frac{e^{-j\pi f_N T_e}}{e^{-j\pi f_N T_e}} \frac{e^{+j\pi f_N T_e} - e^{-j\pi f_N T_e}}{e^{+j\pi f_N T_e} + e^{-j\pi f_N T_e}} \\ j\pi f_A &= \frac{1}{T_e} \frac{2j \sin(\pi f_N T_e)}{2 \cos(\pi f_N T_e)} \\ f_A &= \frac{1}{\pi T_e} \tan(\pi f_N T_e) \end{aligned}$$

Impact dans le domaine fréquentiel

$$f_A = \frac{1}{\pi T_e} \tan(\pi f_n T_e)$$

- La relation f_A et f_N est non linéaire d'où une distorsion en réponse fréquentielle.
- Dans le cas où $f_N T_e \ll 1$ (c'est à dire $f_N \ll f_e$) :

$$\begin{aligned} f_A &= \frac{1}{\pi T_e} \tan(\pi f_N T_e) \\ &\approx \frac{1}{\pi T_e} (\pi f_N T_e) \\ &\approx f_N \end{aligned}$$

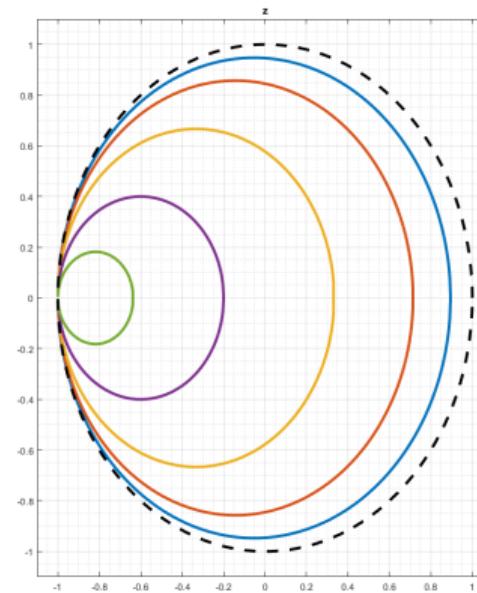
- La distorsion fréquentielle se fait d'avantage ressentir à haute fréquence qu'à basse fréquence.
- Cette méthode de transformation peut être utilisée pour tout type de filtre : passe-bas, passe-bande, passe-haut et coupe-bande. La nature du filtre sera conservée.

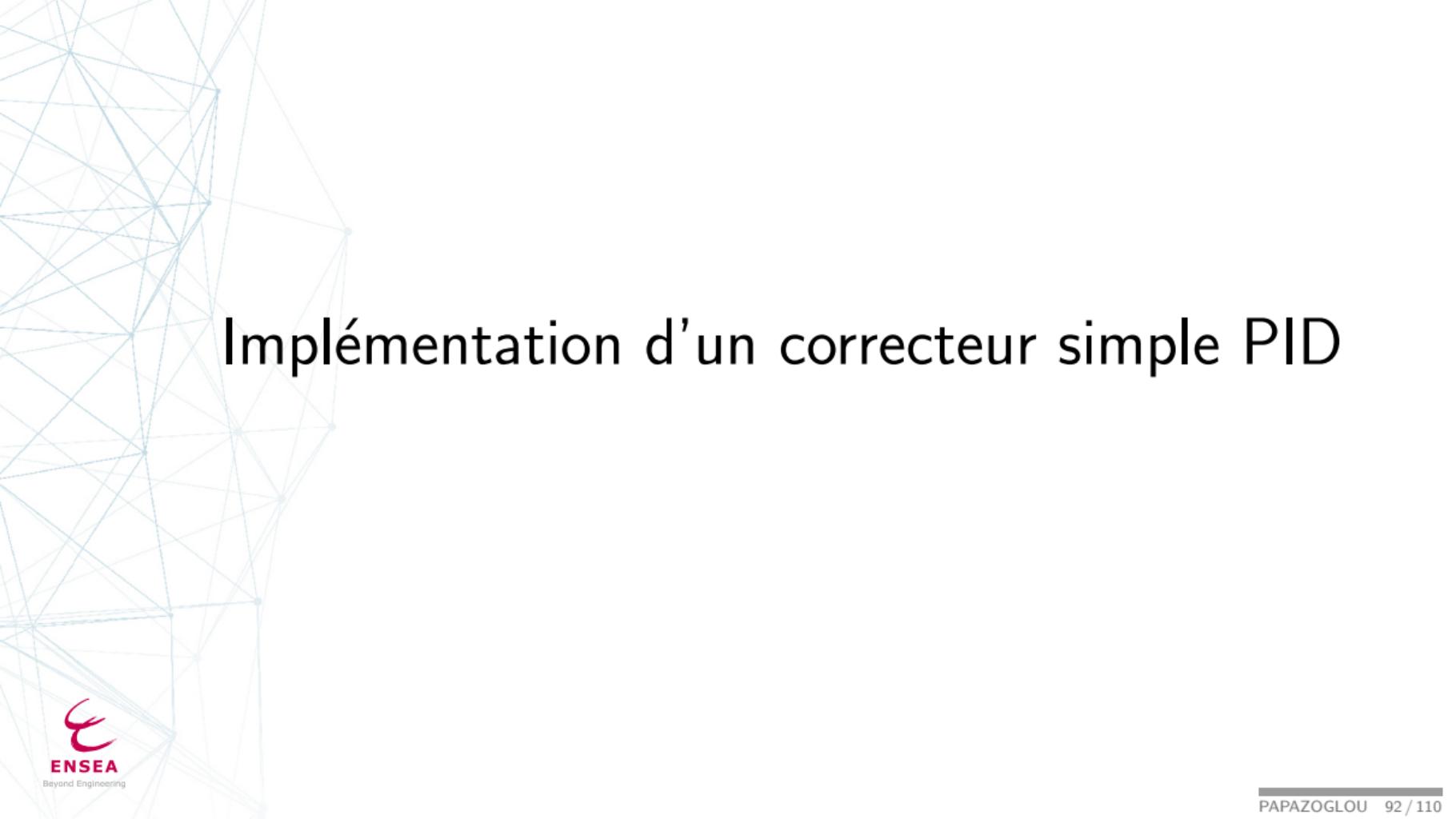
Conservation de la stabilité par transformation bilinéaire

Propriété

- Un système analogique stable donne un système numérique stable car les pôles dans le demi-plan gauche du plan p sont projetés dans le cercle unité du plan Z : si $\text{Re}(p) < 0$ alors $|z| < 1$

Transformation du demi-plan p en disque z





Implémentation d'un correcteur simple PID

Méthode d'implémentation d'un correcteur

- Etablissement des équations du moteur
- Calcul théorique de l'asservissement en temporel continu (Laplace)
- Détermination des fréquences maximales en jeu
- Détermination des fréquences d'échantillonnage
- Application des transformées bilinéaires
- Implementation des filtres

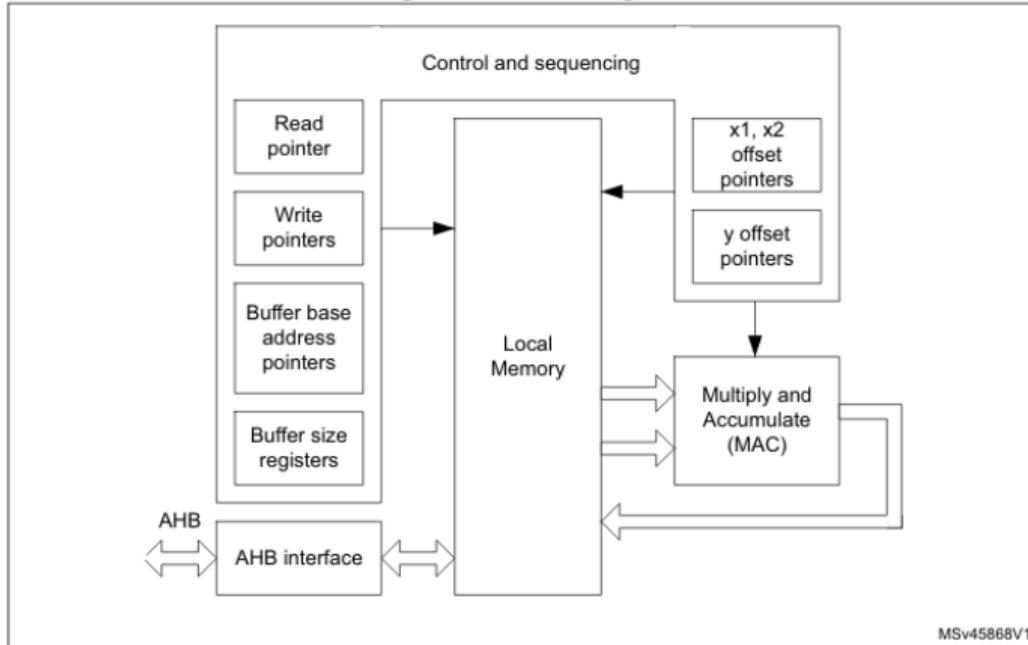
Côté micro-processeur

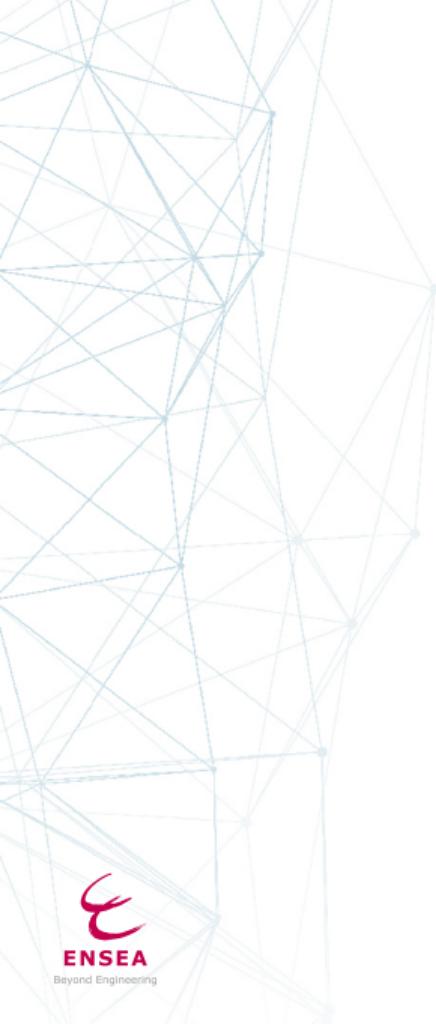
- Mise en place de timer pour l'acquisition des signaux :
 - Timer
 - ADC
 - DMA
 - Interrupt
- Sur chaque interruption, le calcul des nouvelles valeurs doit avoir lieu.

Pour aller plus loin

Filter math accelerator (FMAC)

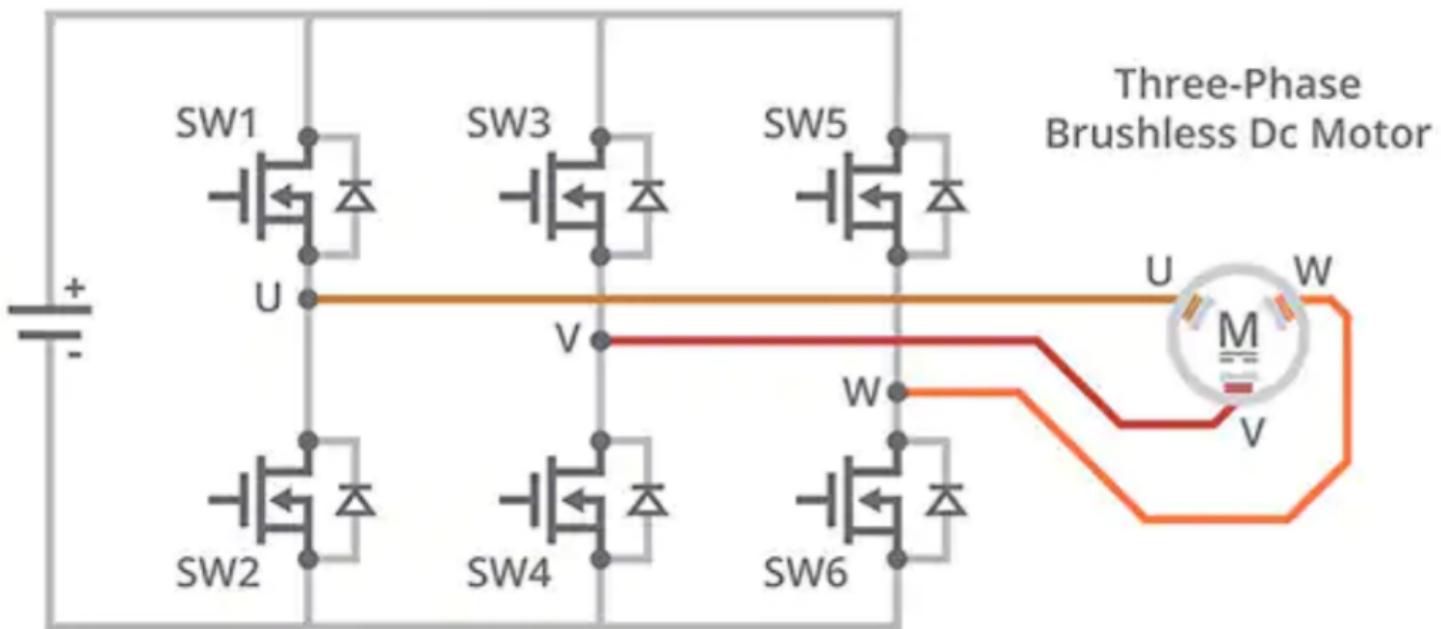
Figure 41. Block diagram



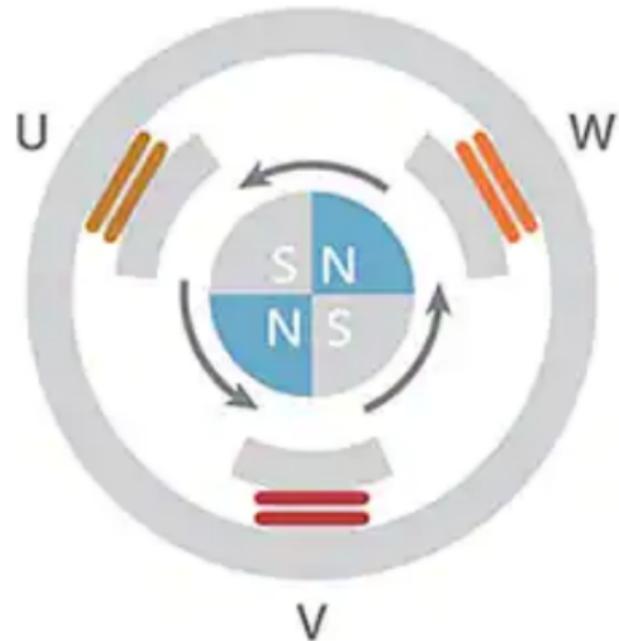


Controle de moteur synchrone

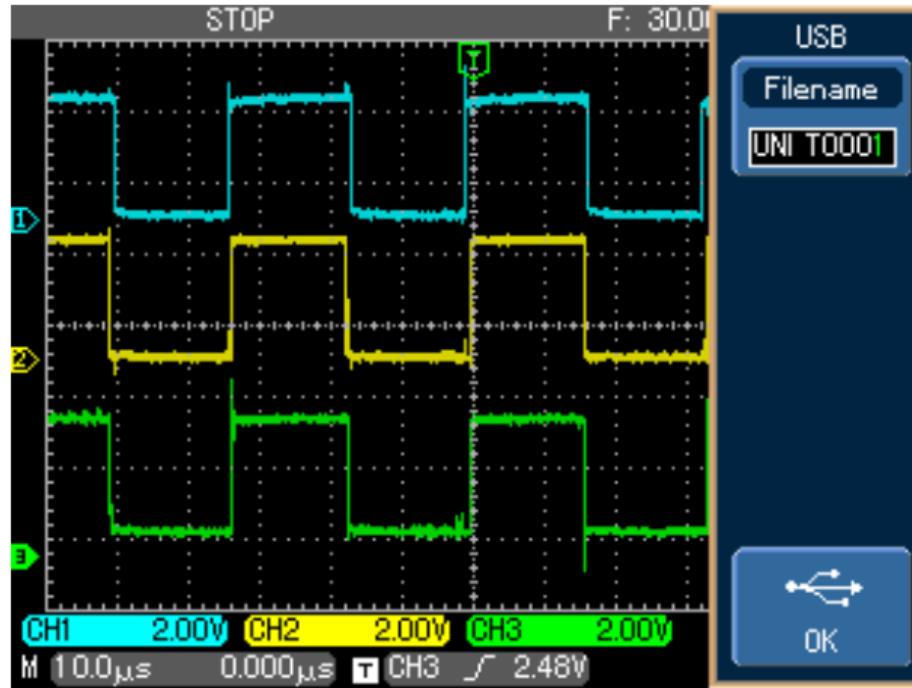
Schéma 3 phases



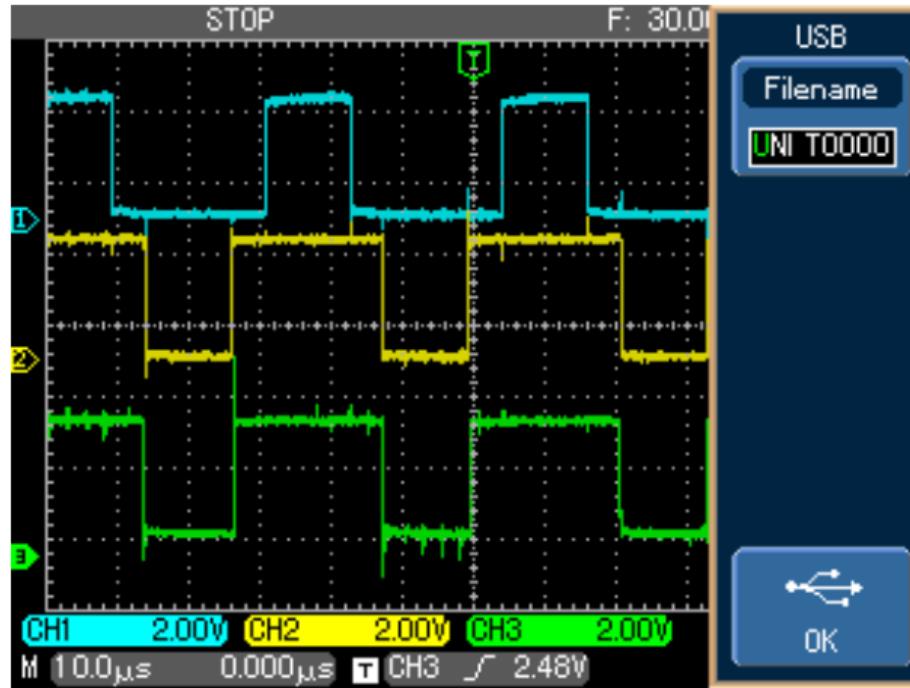
Moteur synchrone



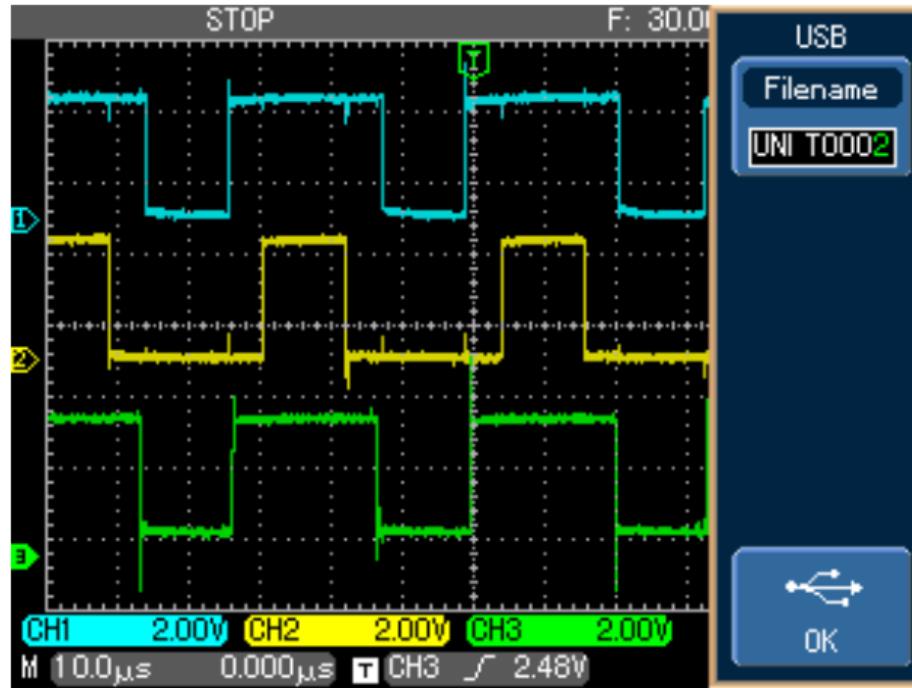
Commande



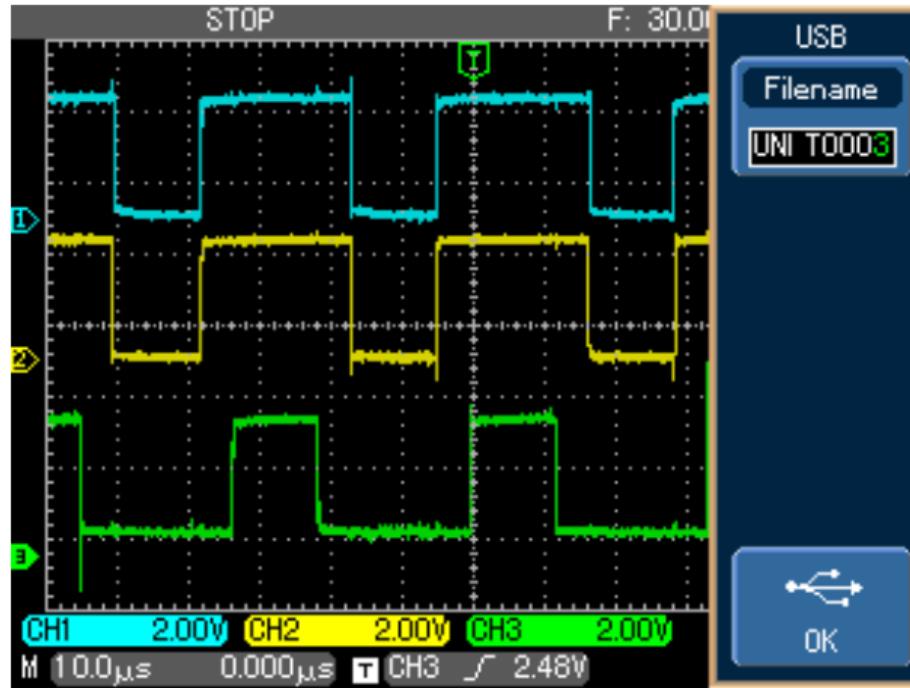
Commande

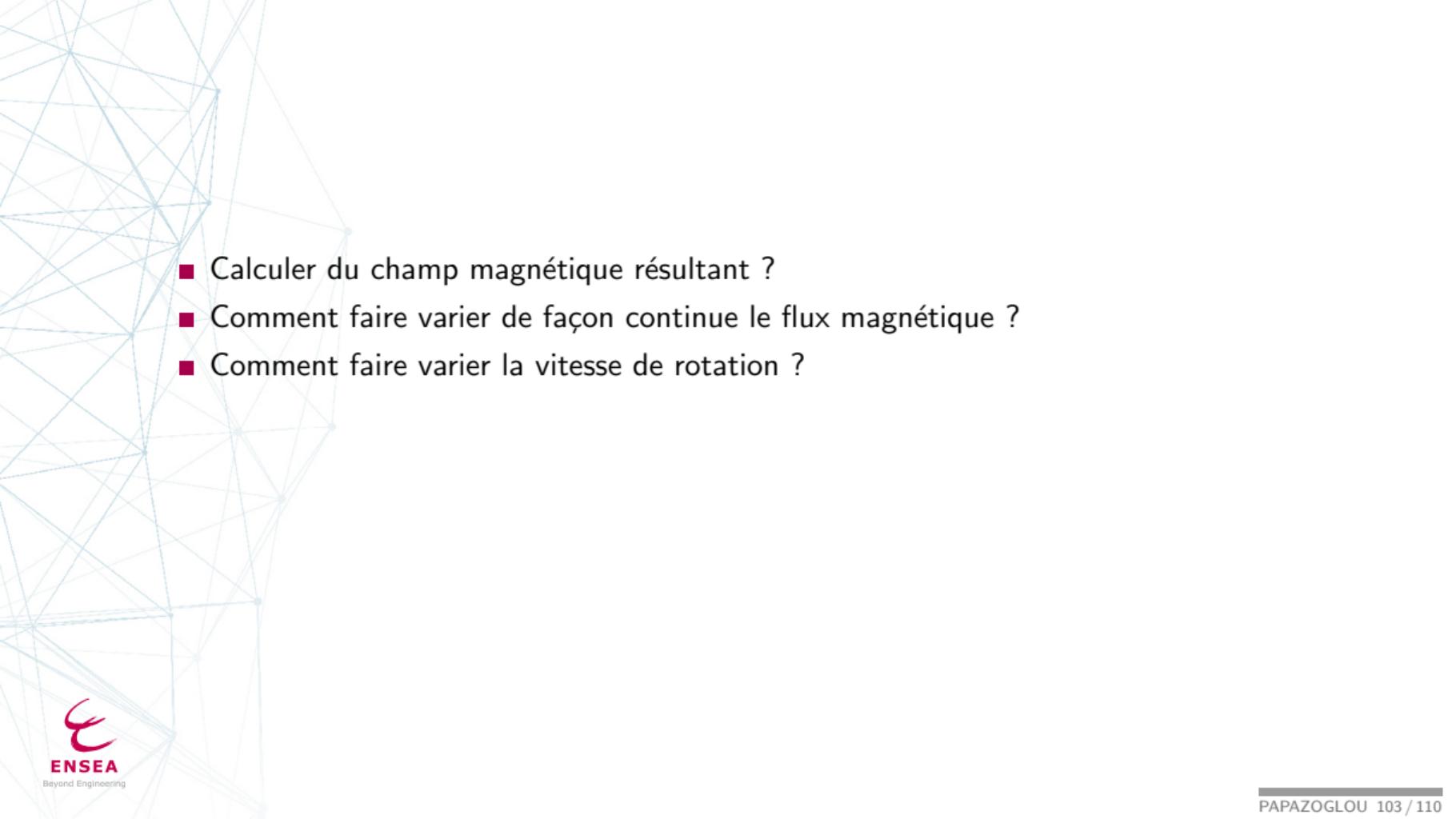


Commande



Commande

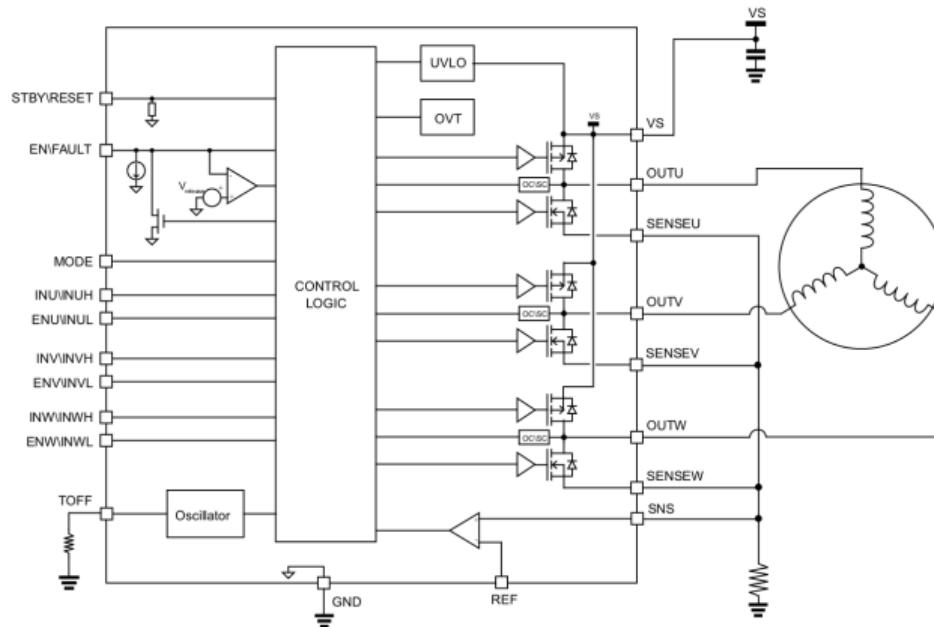


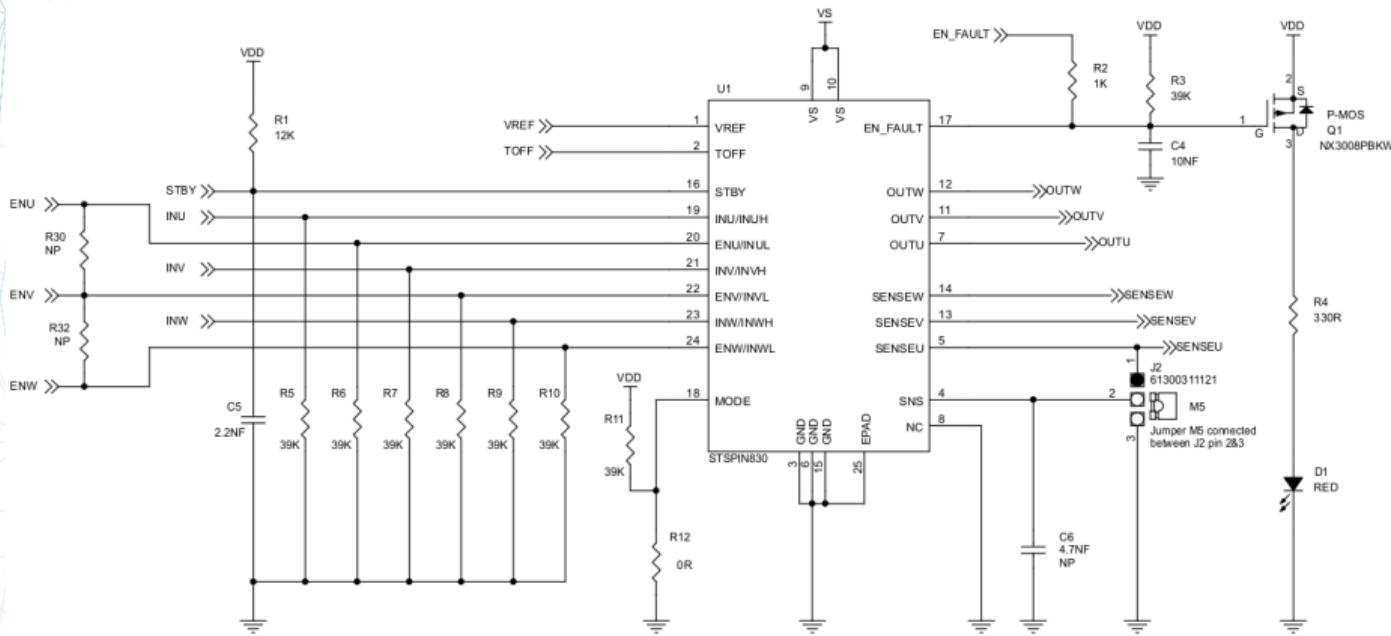
- 
- Calculer du champ magnétique résultant ?
 - Comment faire varier de façon continue le flux magnétique ?
 - Comment faire varier la vitesse de rotation ?

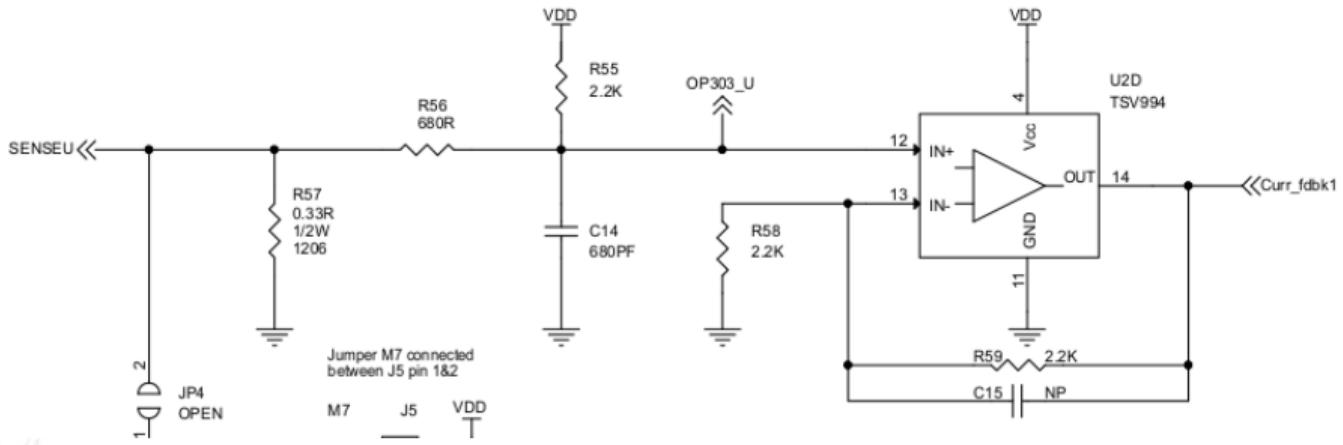
Il faut régénérer les 3 phases sinusoïdales :

- La valeur du rapport cyclique représente la valeur moyenne instantanée de la phase associée.
- Utilisation du "Math Accelerators" pour les calculs trigonométriques.
- L'amplitude (rapport cyclique) commande le couple du moteur, une augmentation de l'amplitude accélère le moteur.
- Une accélération du moteur implique une augmentation de la fréquence des 3 phases (diminution de la valeur du registre ARR).

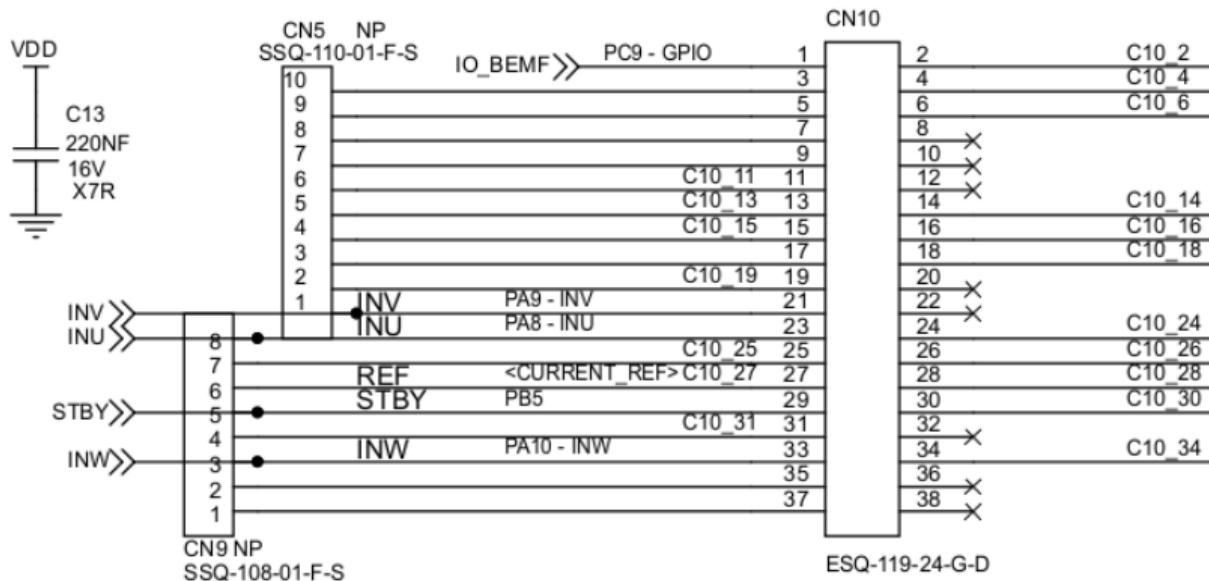
Figure 1. STSPIN830 block diagram







X-NUCLEO-IHM16M1 pinout



Pour aller plus loin...

- Mise en place d'un asservissement (cf TP)
- BEMF : Back Electro-motive Force

Les outils développés par ST

