الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

Mention Électronique

Spécialité Automatique et Informatique Industrielle

présenté par

AMMI ADEL

&

KHELIF MOHAMED

Fréquencemètre basé sur Microcontrôleur ATmega 328P

Proposé par : M. benakki abdelhakim

Année Universitaire 2017-2018

Remerciement

Avant tout, nous remercions Dieu qui nous a donné la force et le courage pour finir ce Travail.

Un merci spécial à nos parents pour avoir pris la peine de notre cours.

A notre aimable encadreur M. Benakki qui nous a dirigés et accompagnés pendant cette Période de travail.

A nos professeurs dont tout le mérite leur revient et dont la disponibilité et la Persévérance ont fait de nous ce que nous sommes.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Introduction générale

Le Fréquence mètre est un appareil dont l'usage touche a des domaines très varié, on le retrouve dans le Domaine industriel, les machines ou bien les moteur. Il est aussi utilisé dans les laboratoires d'enseignement et de recherche.

Ces fréquencemètres indiquent l'amplitude d'oscillation des impulsions dans un temps défini. Les domaines d'utilisation de ces fréquencemètres sont les suivants: les mesures dans des fréquences de réseau, des basses fréquences, hautes fréquences d'oscillations et radiofréquences. En combinaison avec des capteurs externes, il est possible de mesurer des fréquences mécaniques, des tours et d'autres processus physiques. Les fréquencemètres peuvent aussi s'utiliser comme compteurs d'impulsions.

Le circuit fréquencemètre de notre réalisation est base sur le Microcontrôleur Atmega 328P produit par la société Atmel corporation. A cet effet, on réparti ce travail en trois chapitres :

- Le premier chapitre : donne une vue global sur les différents éléments de base du microcontrôleur Atmega 328P, Suivi par des notions des caractéristiques et l'intérieur physique.

- Le deuxième chapitre : aborde les ports d'entrées, sorties, et en plus particulier les timers sur lesquels a été implémenté notre fréquencemètre, pour cela on verra leurs registres internes ainsi que leurs modes de fonctionnement.

- Le troisième chapitre : concerne l'implémentation du fréquencemètre et sa réalisation pratique, et comment faire le les différentes étapes de programmation.

ملخص :

الهدف من مشروعنا هذا هو انشاء جهاز قياس للتردد، بالاعتماد على المايكروكنترولر من نوع ATmega328p وقاعدة برمجة بلغة السي C باستخدام برنامج CodeVisionAVR ، بحيث يتم برمجة المؤقت رقم 1 على عد نبضات الاشارة المراد قياس ترددها، في حين يقوم المؤقت رقم 2 بعد المقاطعات التي تنتج عن كل تشبع للمؤقت رقم 1، يسمح هذا بتحديد زمن 1 ثانية. وبمعرفة عدد النبضات في كل 1 ثانية يمكننا حساب تردد الاشارة الداخلة الى المايكروكنترولر وعرضها على شاشة LCD .

كلمات المفاتيح : قياس التردد ; مؤقت 0 ; مؤقت 1 ;مؤقت 2 ; ميكروكنترولر AVR 328P

Résumé:

L'objectif de notre projet est d'établir un fréquencemètre, basé sur un microcontrôleur ATmega328p et langage de programmation C, en utilisant le programme CodeVisionAVR, afin que la minuterie 1 soit programmée sur la fréquence de l'impulsion à mesurer, Tandis que le temporisateur 2 suit les interruptions résultant de chaque saturation du temporisateur numéro, cela permet une limite de temps de 1 seconde. En connaissant le nombre d'impulsions par seconde, nous pouvons calculer la fréquence du signal entrant vers le microcontrôleur et les afficher sur l'écran LCD.

Mots clés : fréquencemètre avec C ; Compteur 0 ; Compteur 1 ; Compteur 2 ; Microcontrôleur AVR 328P ;

Abstract:

The objective of our project is to establish a frequency counter, based on an ATmega328p microcontroller and programming language C, using the CodeVisionAVR program, so that the timer 1 is programmed on the frequency of the pulse to be measured, while the Timer 2 tracks the interrupts resulting from each saturation of the timer number 1, this allows a time limit of 1 second. By knowing the number of pulses per second, we can calculate the frequency of the incoming signal to the microcontroller and display them on the LCD.

Keywords : Frequency meter ; Timer 0 ; Timer 1 ; Timer 2 ; Microcontrollers AVR 328P

Liste des figures

Chapitre I : Généralité sur ATmega 328P

Figure 1.1. Atmega 328P- 28 broches double ligne paquet (DIP) 4

Figure 1.2. Brochage typique d'un Atmega 328P en boitier (PDIP) 5

Figure 1.3. Architecture de l'Atmega 328P 9

Chapitre II : Organisation des entrées/sorties et des interruptions

Figure 2.1. Synoptique des ports E/S 17

Figure 2.2. Adresse de registre d'atmega 328P ports. 18

Figure 2.3. Le port d'E / S dans AVR 20

Figure 2.4. La résistance pull-up 21

Figure 2.5. Schéma bloc du principe de fonctionnement d'un Timer 22

Figure 2.6. Timer 0 Prescaler / Sélecteur 27

Figure 2.7. TCCR0 les Définitions de Prescaler 28

Figure 2.8. Timer/compteur contrôle registre 29

Figure 2.9. Timer/counter register 31

Figure 2.10. Timer/counter interrupt mask 31

Figure 2.11. Timer/counter interrupt flag register 33

Figure 2.12. Schéma interne du Timer1 36

Figure 2.13. Timer1 registres hauts et bas 37

Figure 2.14. TCCR1B définition de bit 37

Figure 2.15. TIFR {Timer/Counter Interrupt Flag Register) 38

Figure 2.16. TCCR1A (Timer 1 Control) Registre 39

Figure 2.17. TCCR1B (Timer 1 Control) Registre 39

Figure 2.18. Registre de capture d'entre (ICR) pour Timer1 40

Figure 2.19. TCCR2 Définitions de Bit 41

Figure 2.20. TCNT (Timer/Counter register) 41

Figure 2.21. TCCR2 42

Figure 2.22. TIMSK 42

Figure 2.23. TIFR 42

Figure 2.24. Timer 8 bits Diagramme 43

Figure 2.25. Timer 16 Bits Diagramme 43

Figure 2.26. Mode de fonctionnement Normal du timer (comptage) 45

Figure 2.27. PWM forme d'onde. 47

Figure 2.28. Timer/compteur CTC mode 48

Figure 2.29. Mode de fonctionnement CTC du timer. 48

Chapitre III : Réalisation de la Fréquencemètre

Figure 3.1. Schéma de principe d'un fréquencemètre numérique. 49

Figure 3.2. Schéma bloc du circuit de la réalisation 51

Figure 3.3. La structure interne d'un afficheur LCD 52

Figure 3.4. Schéma simplifié de la carte Arduino UNO 54

Figure 3.5. Cartographie des broches d'ATmega 328P 55

Figure 3.6. Simule le Project avec les appareils utilisés. 56

Figure 3.7. Organigramme du programme général. 59

Figure 3.8. Organigramme de Start-Count sous-programme 61

Figure 3.9. Organigramme du sous-programme Timer1 62

Figure 3.10. Organigramme du sous-programme Timer2. 63

Figure 3.11. 1er étape de l'implémentation 64

Figure 3.12. 2eme étape de l'implémentation 65

Figure 3.13.3eme étape de l'implémentation 65

Liste des tableaux

Chapitre I : Généralité sur ATmega 328P

Tableau 1.1. Les paramètres de l'Atmega 328P 4

Tableau 1.2. Description des Brochages de l'Atmega 328P 8

Tableau 1.3. Atmega 328P Registre générale 10

Chapitre II : Organisation des entrées/sorties et des interruptions

Tableau 2.1. Nombre de ports dans certains membres de la famille AVR 18

Tableau 2.2. Différents états d'une broche dans le microcontrôleur AVR. 21

Tableau 2.3. Différents registres de contrôle associes à chaque timer 25

Tableau 2.4. Les bits du registre 26

Chapitre III : Réalisation de la Fréquencemètre

Tableau 3.1. Brochage d'un afficheur LCD. 53

Tableau 3.2. Résultat des essais pour 1 seconde 66

Tableau 3.3. Résultat des essais pour 1/3 seconde 67

Listes des acronymes et abréviations :

ADC : convertisseurs analogiques-numériques

ALU : L'unité arithmétique et logique

AREF : est l'entrée de référence analogue pour le Convertisseur A/D

ASCII : L'American Standard Code for Information Interchange (American Standard Code for Information Interchange)

Atmel AVR : une famille de microcontrôleurs

AVCC : est une broche de tension d'alimentation pour le Convertisseur A/D

Clk : L'horloge

CTC : effacer le compteur de minuterie (Clear Timer Counter)

CPU : Un processeur (ou unité centrale de traitement)

CS : source d'horloge (clock source)

DPRAM : la mémoire vive dynamique

DDRx : registre de direction

ICR : registre d'entrée de capture (Input Capture Register)

GND : Masse de l'alimentation

MIPS : Million d'instructions par seconde

OCF: l'indicateur d'interruption de la sortie (the Output Compare Interrupt Flag)

OCIE : Activation de l'interruption de la sortie du compteur du compteur (Timer Counter Output Compare Interrupt Enable)

OCR : registre de comparaison en sortie (Output Compare Register)

PORTx : registre de données

PIND : Registre de lecture des données

PWM : Un signal à modulation de largeur d'impulsion

RAM : la mémoire vive

RISC : Architecture d'un processeur à jeu d'instructions réduit

ROM : la mémoire morte

SPI : une interface périphérique série

SRAM *:* la mémoi*e* vive statique

TCCRn : Registre de contrôle du compteur de minuterie (Timer Counter Control Register)

TCNTn : registre de compteur de minuterie (Timer Counter Register)

TIFR : Registre d'interruption de minuterie (Timer Interrupt Register)

TIMSK : Registre de masque d'interruption de minuterie (Timer Interrupt Mask Register)

TOIE : Interruption de dépassement de la minuterie (Timer overflow interrupt Enable)

TOV : Débordement de la minuterie (Timer overflow)

UART : Universal Asynchrones Receiver Transmitter

VCC : Broches d'alimentation du microcontrôleur (+3 à +5V)

XTAL1 : Entrée de l'oscillateur externe ou libre pour l'horloge interne

XTAL2 : Production de l'amplificateur d'oscillateur

WGM : Bits de mode de génération d’onde (Wave Generation Mode Bits)

Table des matières

Remerciements

Résumé

Liste des abréviations et acronymes

Liste des figures

Liste des tableaux

Introduction générale 1

Chapitre I : Généralité sur ATmega 328P

1-1-Introduction 3

1-2-Définition 3

1-3- Présentation Physique 5

1-3-1 Brochages 5

1-3-2 Block Diagramme 9

1-3-3 les diffèrent modules 11

1-4 Mode d'adressage 13

Chapitre II : Organisation des entrées/sorties et des interruptions

2.1 Introduction 16

2.2 Les ports d'entrées/Sorties (PORTx) 17

2.3 Etude des Timers (Compteur programmables) de L'Atmega328p 22

2.3.1 DESCRIPTION 22

2.3.2 Les Registres de base des Timers 25

2.3.4 Timer/ COMPTEUR PRESCALER ET SÉLECTEURS D'ENTRÉE 27

2.3.5 Timer 1 35

2.3.6 Timer 2 40

2.4 Les Modes de Timer 44

2.4.1 Mode normal 44

2.4.2 Mode de capture d'entrée 45

2.4.3 Mode de capture de sortie 46

2.4.4 Mode modulateur de largeur d'impulsion(PWM) 46

2.4.5 Effacer sur le mode de comparaison de correspondance (CTC) 47

Chapitre III : Réalisation de la Fréquencemètre

3-1-Introduction 49

3.2 Principe de fonctionnement du fréquencemètre numérique 49

3.3 La partie réalisation 51

3.3.1 Le circuit LCD 51

3.3.2 Le Microcontrôleur 328P 53

3.4 La Partie Software 57

3.4.1 CodevisionAVR 57

3.4.2 Les organigrammes et le programme 57

3.4.3 Implémentation 64

3.5 Les Résultats obtenus 66

Chapitre I : Généralités sur Atmega 328P

1-1-Introduction :

Les microcontrôleurs ATMega appartiennent à la famille de microcontrôleurs AVR et sont fabriqués par Atmel Corporation.

Ceux sont des microcontrôleurs à 8 bits avec une architecture RISC (Harvard), il dispose de fonctionnalités standard telles que la mémoire ROM, la mémoire RAM, la mémoire EEPROM les minuteurs et les ports d'entrée / sortie, ainsi que des périphériques supplémentaires comme les convertisseurs analogiques-numériques (ADC), les ports d'interface série, etc. Ils ont 120 instructions dans leur répertoire et une plage de mémoire de programme allant de 4K à 256K octets.

1-2-Définition :

Le microcontrôleur ATMega 238P est basé sur une architecture de Harvard c'est-à-dire une mémoire de données séparée a une mémoire de programme.

La mémoire du programme, également appelée mémoire de code. La taille de la mémoire du programme est de 32 Ko en lecture-écriture, c'est une mémoire de type flash.

La mémoire de données est divisée en trois parties: 32 registres à usage général, 23 mémoires d'entrée / sortie et mémoire statique à accès aléatoire statique (SRAM) 2 Ko.

Trois timers / compteurs flexibles avec modes de comparaison, interruptions internes et externes, USART programmable en série, interface série 2 fils orientée octet, port série SPI, convertisseur A/N avec 6 canaux 10 bits (8 canaux dans TQFP et QFN / MLF packages), horloge de surveillance programmable avec oscillateur interne et cinq modes d'économie d'énergie sélectionnables par logiciel. L'appareil fonctionne entre 1,8-5,5 volts.

De plus, il possède certaines fonctionnalités constituées d'une architecture RISC avancée, bonne performance, compteur de minuterie réel ayant un oscillateur séparé, 6 broches PWM, verrouillage de la programmation pour la sécurité du logiciel.
En exécutant des instructions puissantes dans un seul cycle d'horloge, l'appareil atteint des débits proches de 1 MIPS par MHz, équilibrant la consommation d'énergie et la vitesse de traitement.

Chapitre I : Généralités sur Atmega 328P

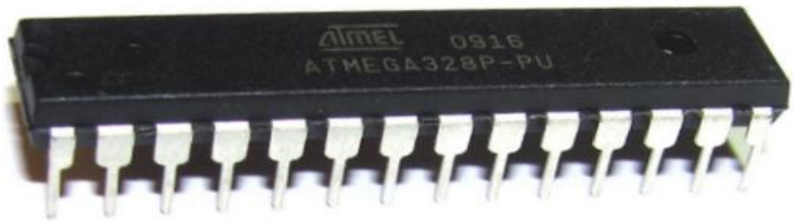


Figure 1.1. Atmega 328P- 28 broches double ligne paquet (DIP)

En résume les paramètres de l'Atmega 328P sont décrit par le tableau 1.1

Nom	**Value**
Type de CPU	AVR 8-bits
Type de mémoire de programme	Flash
Taille de mémoire de programme (KB)	32
CPU Speed (MIPS/DMIPS)	20
SRAM Bytes	2,048
Data EEPROM/HEF (bytes)	1024
Digital Communication Peripherals	1-UART, 2-SPI, 1-I2C
Capture/Compare/PWM Peripherals	1 Input Capture, 1 CCP, 6PWM
Timers	2 x 8-bit, 1 x 16-bit
Number of Comparators	1
Temperature Range (C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	32

Tableau 1.1. Les paramètres de l'Atmega 328P

1-3- Présentation Physique

1-3-1 Brochages

L'ATMEGA se présente sous la forme d'un circuit intégré à 28 broches pour le modèle ATMEGA 328P en boîtier PDIP.

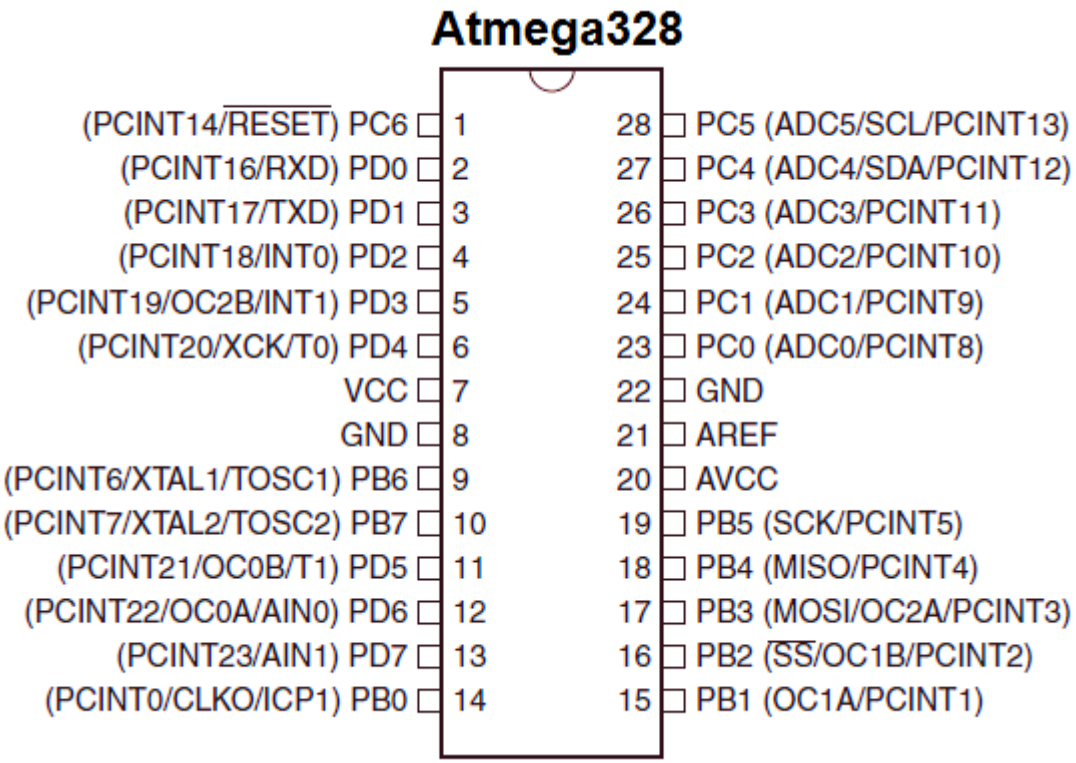


Figure 1.2. Brochage typique d'un Atmega 328P en boitier PDIP

- **Descriptions des broches**

1. **Port B (PB [7:0]) XTAL1/XTAL2/TOSC1/TOSC2**

Le port B est un port d'E / S bidirectionnel 8 bits avec des résistances de pull-up internes (sélectionnées pour chaque bit). Le Port B tampons de sortie ont des caractéristiques d'attaque symétriques avec une capacité élevée de puits et de source. En entrée, Les broches du port B qui sont tirées de l'extérieur vers le bas génèrent du courant si les résistances de pull-up sont activées. Le port des broches B sont tri-déclarées lorsqu'une condition de réinitialisation

devient active, même si l'horloge ne fonctionne pas. Selon les réglages des fusibles de sélection d'horloge, PB6 peut être utilisé comme entrée pour l'oscillateur inverseur amplificateur et entrée dans le circuit d'horloge interne. Selon les réglages des fusibles de sélection d'horloge, PB7 peut être utilisé comme sortie de l'oscillateur inverseur amplificateur.

2. **Port C (PC [5:0])**

Le port C, est un port d'E / S bidirectionnel à 7 bits avec des résistances pull-up internes (sélectionnées pour chaque bit). Le PC [5: 0] les tampons de sortie ont des caractéristiques d'attaque symétriques avec une capacité élevée de puits et de source.

En entrée, Les broches du port C qui sont tirées de l'extérieur vers le bas génèrent du courant si les résistances de pull-up sont activées. Le port Les broches C sont tri-déclarées lorsqu'une condition de réinitialisation devient active, même si l'horloge ne fonctionne pas.

3. **PC6/RESET**

Déclenché par un front descendant maintenue plus de 50 ns il produira le Reset du microcontrôleur, même si l'horloge ne court pas.

4. **Port D (PD [7:0])**

Le port D est un port d'E / S bidirectionnel 8 bits avec des résistances de pull-up internes (sélectionnées pour chaque bit). Le Port D Les tampons de sortie ont des caractéristiques d'attaque symétriques avec une capacité élevée de puits et de source. En entrée, Les broches du port D qui sont tirées de l'extérieur vers le bas génèreront du courant si les résistances de pull-up sont activées. Le port Les broches D sont tri-déclarées lorsqu'une condition de réinitialisation devient active, même si l'horloge ne fonctionne pas.

5. **XTAL1**

Entrée de l'oscillateur externe ou libre pour l'horloge interne.

6. **XTAL2**

Production de l'amplificateur d'oscillateur.

7. AVCC

est une broche de tension d'alimentation pour le Convertisseur A/D qui doit être connectée à VCC via un filtre passe-bas pour éviter les parasites.

8. AREF

est l'entrée de référence analogue pour le Convertisseur A/D avec une tension dans la gamme de 2 V à AVCC avec filtre passe bas.

9. VCC

Broches d'alimentation du microcontrôleur (+3 à +5V).

10. GND

Masse de l'alimentation.

Dans le tableau 1.2 montre la description pour chacune des broches, ainsi que leur fonction.

Pin Number	Description	Function
1	PC6	Reset
2	PD0	Digital Pin (RX)
3	PD1	Digital Pin (TX)
4	PD2	Digital Pin
5	PD3	Digital Pin (PWM)
6	PD4	Digital Pin
7	Vcc	Positive Voltage (Power)
8	GND	Ground
9	XTAL 1	Crystal Oscillator
10	XTAL 2	Crystal Oscillator
11	PD5	Digital Pin (PWM)
12	PD6	Digital Pin (PWM)
13	PD7	Digital Pin
14	PB0	Digital Pin
15	PB1	Digital Pin (PWM)
16	PB2	Digital Pin (PWM)
17	PB3	Digital Pin (PWM)
18	PB4	Digital Pin
19	PB5	Digital Pin
20	AVcc	Positive voltage for ADC (power)
21	Aref	Reference Voltage
22	GND	Ground
23	PC0	Analog Input
24	PC1	Analog Input
25	PC2	Analog Input
26	PC3	Analog Input
27	PC4	Analog Input
28	PC5	Analog Input

Tableau 1.2. Description des Brochages de l'Atmega 328P

1-3-2 Block Diagramme

Le synoptique suivant présente le fonctionnement général du microcontrôleur ATMEGA 328P.

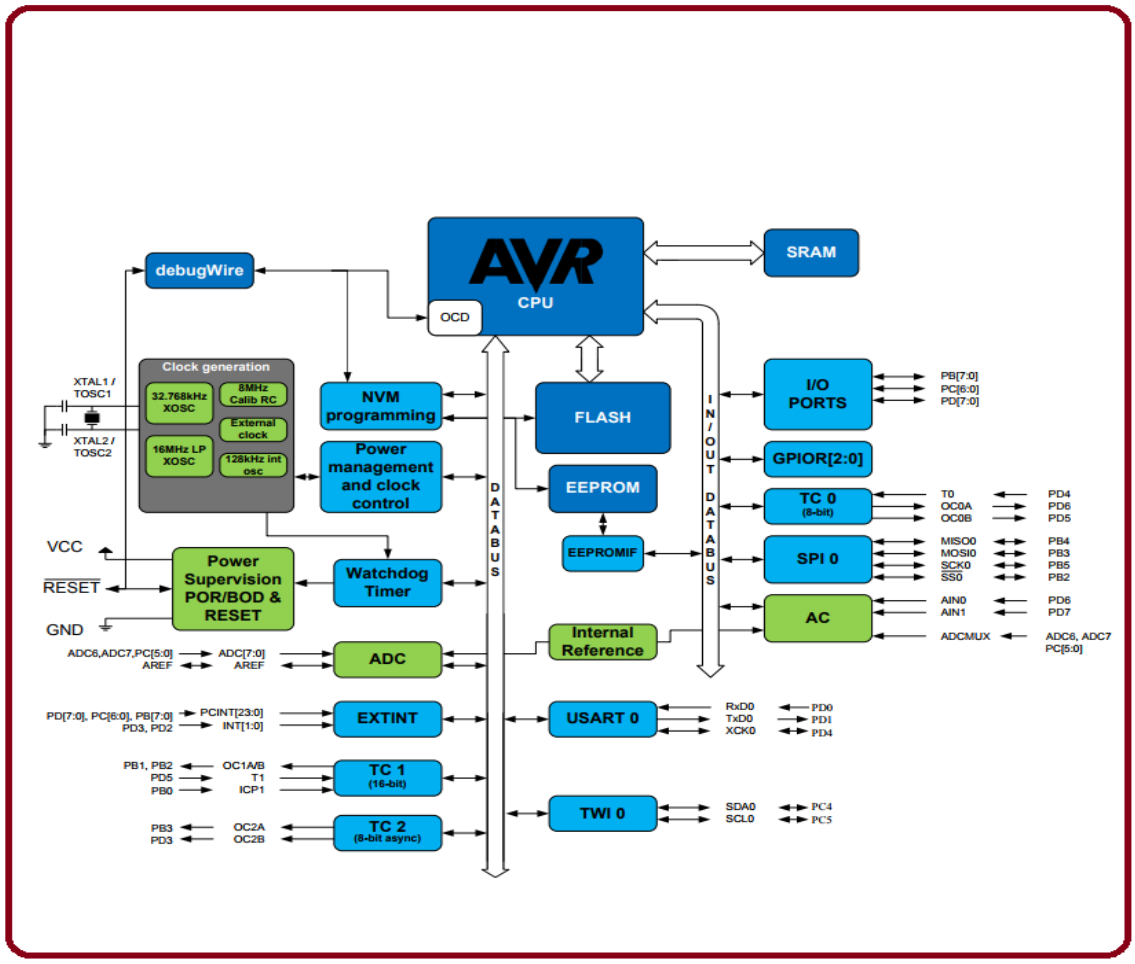


Figure 1.3. Architecture de l'Atmega 328P

Le cœur AVR combine un jeu de 131 instructions riches avec 32 registres spéciaux travaillants directement avec l'Unité Arithmétique de Logique ALU, qui représente le registre d'accumulateur dans les microcontrôleurs classiques.

Ces registres spéciaux permettent à deux registres indépendants d'être en accès directs par l'intermédiaire d'une simple instruction et exécutée sur un seul cycle d'horloge. Cela signifie que pendant un cycle d'horloge simple l'Unité Arithmétique et Logique ALU exécute l'opération

et le résultat est stocké en arrière dans le registre de sortie, le tout dans un cycle d'horloge. L'architecture résultante est plus efficace en réalisant des opérations jusqu'à dix fois plus rapidement qu'avec des microcontrôleurs conventionnels CISC, Les registres spéciaux sont dit aussi registre d'accès rapide et 6 des 32 registres peuvent être employés comme trois registre d'adresse 16 bits pour L'adressage indirect d'espace de données (X, Y & Z). Le troisième Z est aussi employé comme indicateur d'adresse pour la fonction de consultation de table des constantes.

Les 32 registres sont détaillés dans le tableau qui suit avec l'adresse effective dans la mémoire SRAM.

Bit 7 à 0	Adresse	Registre Spéciaux
R0	$00	
R1	$01	
Rn	$xx	
R26	$1A	Registre X Partie Basse
R27	$1B	Registre X Partie Haute
R28	$1C	Registre Y Partie Basse
R29	$1D	Registre Y Partie Haute
R30	$1E	Registre Z Partie Basse
R31	$1F	Registre Z Partie Haute

Tableau 1.3. Atmega 328P Registre générale

Les informations sont diffusées par un bus de donnée à 8 bits dans l'ensemble du circuit. Le microcontrôleur possède aussi un mode sommeil qui arrêter l'unité centrale en permettant à la SRAM, les Timers/Compteurs, l'interface SPI d'interrompre la veille du système pour reprendre le fonctionnement. Lors de l'arrêt de l'énergie électrique, le mode économie sauve le contenue

des registres et gèle l'oscillateur, mettant hors de service toutes autres fonctions du circuit avant qu'une éventuelle interruption logicielle ou matérielle soit émise. Dans le mode économie, l'oscillateur du minuteur continue à courir, permettant à l'utilisateur d'entretenir le minuteur RTC tandis que le reste du dispositif dort. Le dispositif est fabriqué en employant la technologie de mémoire à haute densité non volatile d'ATMEL. La mémoire FLASH est reprogrammable par le système avec l'interface SPI ou par un programmateur de mémoire conventionnel non volatile.

1-3-3 les diffèrent modules

1. Registres à usage général :

Les microcontrôleurs ATMega ont une architecture basée sur un registre, c'est-à-dire que les opérandes et le résultat des opérations sont stockés dans les registres, colloqués avec l'unité centrale de traitement (CPU). Les registres à usage général sont couplés à l'unité logique arithmétique (ALU) du processeur.

Ces registres sont utilisés pour stocker temporairement des informations lors de l'exécution d'un programme. Ceux-ci consomment 32 octets d'espace de mémoire de données et prennent l'emplacement d'adresse - $ 00 à $ FF. Ces registres sont de nomenclature R0 à R31 et ont chacun une largeur de 8 bits.

2. Mémoire d'entrée / sortie :

Cette mémoire est également appelée mémoire SFR (Registre des fonctions spéciales) car elle est dédiée aux fonctions spéciales telles que les registres d'état, les timers, les communications série, les ports d'E / S, les compteurs analogiques à numériques (ADC), etc.

Le nombre d'emplacements occupés par cette mémoire dépend du nombre de broches et des fonctions périphériques supportées par la puce. Alors que 64 octets d'emplacement d'E / S sont fixés pour toutes les puces, certains microcontrôleurs ATMega ont une mémoire d'E / S étendue qui contient des registres liés aux ports et périphériques supplémentaires.

3. SRAM interne:

Ceci est également appelé bloc-notes et est utilisé pour stocker des données et des paramètres par les programmeurs et les compilateurs. Chaque emplacement est accessible directement par son adresse. Ceci est utilisé pour stocker les données des entrées / sorties et des ports série dans la CPU.

4. EEPROM Flash:

C'est une mémoire programmable intégrée au système utilisée pour stocker les programmes. Il est effaçable et programmable en tant qu'unité unique. Comme il est non volatile, le contenu de la mémoire est conservé même en cas de mise hors tension.

Pour chaque microcontrôleur ATMega, le numéro à la fin du nom indique la capacité de la mémoire flash.

5. Ports:

Les microcontrôleurs ATMega contiennent quatre ports 8 bits - Port A, Port B, Port C et Port D. Chaque port est associé à trois registres - Registre de données (écrit les données de sortie sur le port), Registre de direction des données (définit une broche de port spécifique en sortie ou entrée) et l'adresse de la broche d'entrée (lit les données d'entrée du port).

6. Clock:

L'horloge du microcontrôleur est utilisée pour fournir une base de temps aux sous-systèmes périphériques. Nous pouvons régler l'horloge en interne en utilisant le condensateur de résistance sélectionnable par l'utilisateur ou en utilisant des oscillateurs externes.

7. Timers et compteurs:

Les microcontrôleurs ATMega contiennent généralement 3 Timers/ Compteurs. Bien que deux timer de 8 bits puissent également être utilisés comme compteurs, le troisième est un compteur de 16 bits. Ils sont utilisés pour générer des signaux de sortie de précision, compter des événements externes ou mesurer des paramètres du signal numérique d'entrée.

8. Systèmes de communication série:

La puce du microcontrôleur ATMega contient un récepteur et un transmetteur série synchrone et asynchrone universel (USART), une interface périphérique série (SPI) et une interface série à deux fils (TWI).

9. Analog to Digital Converters:

Les microcontrôleurs ATMega contiennent un sous-système ADC (Analog to Digital Converter) multicanal. L'ADC a une résolution de 10 bits et fonctionne selon le principe de l'approximation successive. Il est associé à trois registres: registre de sélection de multiplexeur ADC, contrôle ADC et registre d'état, et registre de données ADC.

10. Interruptions :

Il y a 21 périphériques d'interruption dans les microcontrôleurs ATMega. Alors que 3 sont utilisés pour des sources externes, les 19 restants sont utilisés pour les sous-systèmes internes.

Ceux-ci sont utilisés pour interrompre la séquence normale des événements en cas d'urgence de haute priorité.

1-4 Mode d'adressage

On appelle « mode d'adressage » la manière dont la donnée est spécifiée dans une instruction. Selon le mode d'adressage la taille de l'instruction peut varier de 1 à 4 octets.

Il existe 5 modes d'adressage :

1. Le mode d'adressage implicite
2. Le mode d'adressage immédiat
3. Le mode d'adressage relatif
4. Le mode d'adressage direct
5. Le mode d'adressage indirect

1. Le mode d'adressage implicite

Le mode d'adressage implicite correspond à une instruction ne comportant pas d'opérande. L'instruction est composée du code opération uniquement et sa taille peut varier entre 1 octet et 2 octets selon l'opération.

code opération (1 ou 2 octets)

Ce type d'instruction porte généralement sur des registres. Les opérations d'incrémentation ou de décrémentation d'un registre ont un mode d'adressage implicite.

2. Le mode d'adressage immédiat

On parle de mode d'adressage immédiat lorsque le code opérande contient une donnée. La taille de la donnée peut varier entre 1 et 2 octets.

code opération (1 ou 2 octets)	code opérande (1 ou 2 octets)

Ce type d'instruction met en jeu un registre et une valeur (qu'il s'agisse d'une affectation, une addition, une soustraction ou bien même une comparaison), la taille de l'opérande dépendra donc du type de registre mis en jeu (1 octet pour un registre 8 bits, 2 pour un registre de 16 bits).

3. Le mode d'adressage relatif

Ce type de mode d'adressage met en jeu un champ opérande contenant un entier relatif (sa taille est donc un octet).

code opération (1 octet)	code opérande (1 octet)

On l'utilise pour les opérations de saut, l'entier relatif est appelé *déplacement*, il correspond à la longueur du saut que le processeur doit effectuer dans les instructions.

4. Mode d'adressage direct

Le code opérande d'une instruction en mode d'adressage direct, contrairement au mode d'adressage immédiat, contient l'adresse d'une donnée en mémoire (au lieu de contenir la donnée). Une adresse étant codée sur 16 bits, la taille du champ opérande est donc de 2 octets.

code opération (1 ou 2 octets)	code opérande (2 octets)

Il peut s'agir par exemple de l'affectation à un registre d'une donnée contenue dans une case mémoire. Ce mode d'adressage provoque un temps d'exécution de l'instruction plus long car l'accès à la mémoire principale est plus long que l'accès à un registre.

5. Mode d'adressage indirect

Le mode d'adressage indirect permet d'accéder à une donnée par l'intermédiaire d'un registre qui contient son adresse. Son utilité n'est pas apparente à ce stade, mais l'adressage indirect est très utile lors de l'utilisation de tableaux (parcours des cases d'un tableau) car il suffit d'incrémenter Rn de la taille d'une case pour passer d'une case à une autre.

Chapitre II : Organisation des entrées/sorties et des Interruptions

2.1 Introduction

L'Atmega328P dispose de 2 ports d'interfaces de 8 bits chacun (B et D) et d'un port C de 7bits, tous accessibles à travers 3 registres de 8 bits. Cependant certaines broches de ces ports peuvent être partagées avec d'autres ressources internes (Timer, ADC …) . C'est la phase d'initialisation qui permet de déterminer quelles seront les ressources mises en œuvre. Chaque port dispose de :

Les registres PINx permettent au microcontrôleur de lire l'état logique des broches. Les registres PORTx permettent au microcontrôleur de forcer à un niveau logique les broches des ports (commande d'actionneurs « tout ou rien »).

Les 3 ports regroupent 23 broches parmi les 28 broches au total, les 5 qui restent sont utilisées pour l'alimentation du microcontrôleur (Vcc et GND) et la tension de référence de l'ADC (Vref).

Donc on constate que le microcontrôleur communique avec l'environnement extérieur à travers ces différents ports, ce qui fait que ces derniers jouent un rôle essentiel dans les opérations d'entrées / sorties.

2.2 Les ports d'entrées/Sorties (PORTx)

Les microcontrôleurs ATMEL sont pourvus des ports d'entrées/sorties numériques pour communiquer avec l'extérieur. Ces ports sont multidirectionnels et configurable broche à broche soit en entrée, soit en sortie.

D'autres modes sont aussi utilisables comme des entrées analogiques, des fonctions spéciales de comparaison, de communication synchrone, … Mais pour le moment nous allons voir la fonction Numérique des ports.

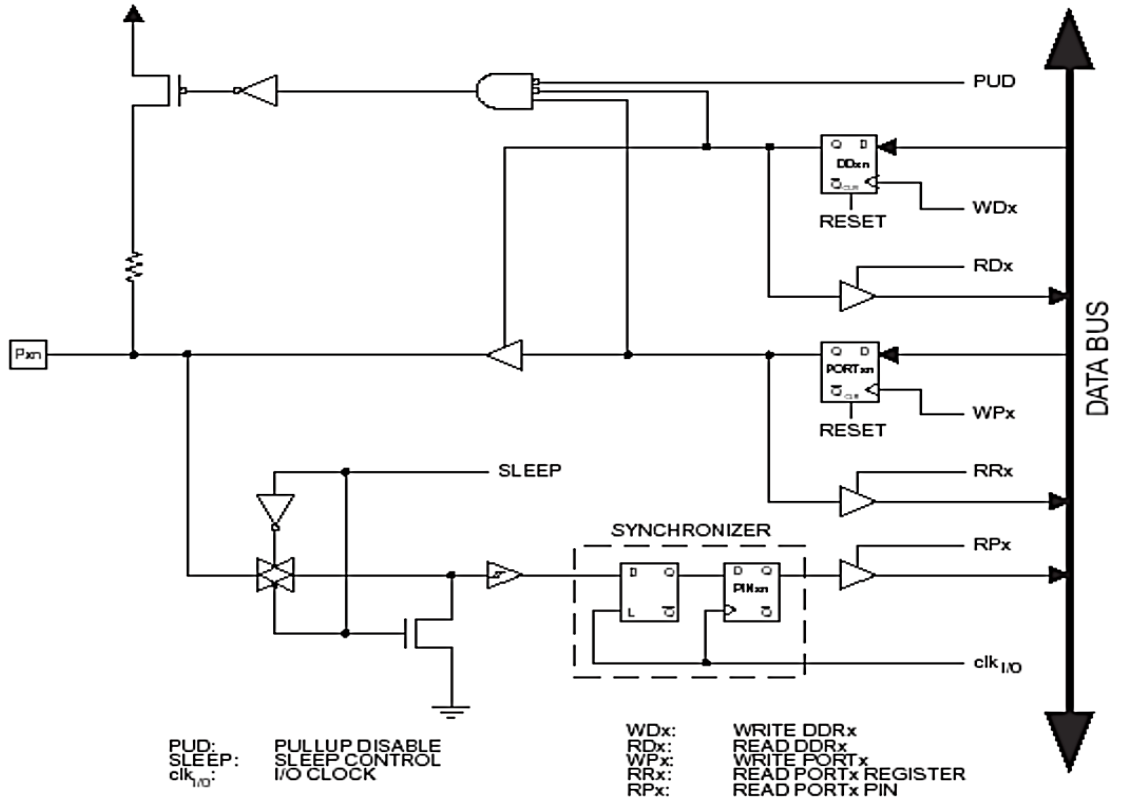


Figure 2.1. Synoptique des ports E/S.

Chapitre II : Organisation des entrées/sorties et des Interruptions

Le nombre de ports dans la famille AVR varie en fonction du nombre de broches sur la puce.

Pins	8-pin	28-pin	40-pin	64-pin	100-pin
Chip	ATtiny25/45/85	ATmega8/48/88	ATmega32/16	ATmega64/128	ATmega1280
Port A			X	X	X
Port B	6 bits	X	X	X	X
Port C		7 bits	X	X	X
Port D		X	X	X	X
Port E				X	X
Port F				X	X
Port G				5 bits	6 bits
Port H					X
Port J					X
Port K					X
Port L					X

Tableau 2.1. Nombre de ports dans certains membres de la famille AVR

Note : le X indique que le port disponible

Pour utiliser l'un de ces ports comme port d'entrée ou de sortie, il doit être programmé, en plus d'être utilisé pour des E / S simples, chaque port a d'autres fonctions, ADC, minuteries, interruptions, communication série.

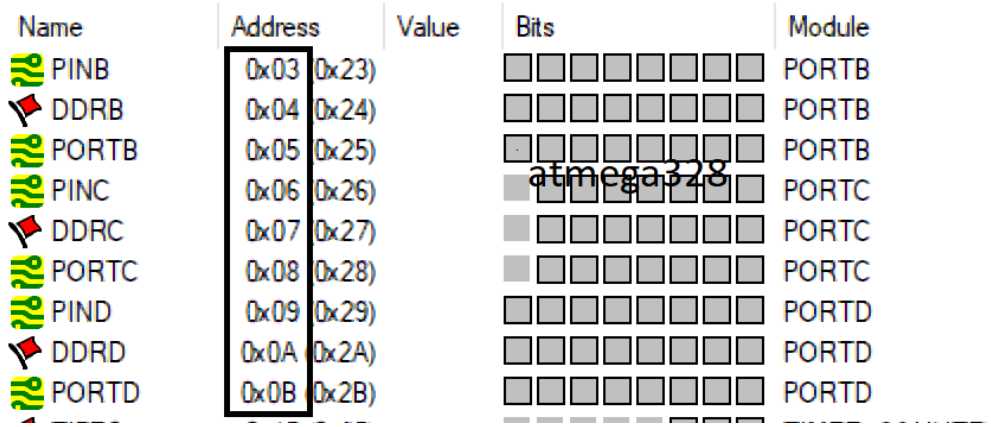


Figure 2.2. Adresse de registre d'atmega 328P ports

Chapitre II : Organisation des entrées/sorties et des Interruptions

DDRx: Registre de direction (Mode Entrée, Sortie)
PORTx: Registre de sortie de donnés
PINx: Registre de lecture des données

X Représente le nom de port (B, C ou D).

Chaque registre et configurable bit à bit, c'est à dire que sur l'on peut utiliser sur le même port des fonctions en entrée et/ou en sortie simultanément. Par exemple, on peut avoir les quatre premiers bits en entrée et les quatre derniers bits en sortie sur un même port Notez que chacun des registres d'E / S a une largeur de 8 bits, et que chaque port a un maximum de 8 broches, là pour chaque bit des registres d'E / S affecte l'une des broches, le contenu du bit 0 de DDRB représente la direction de la broche PB0, et ainsi de suite).

- Chaque port de 8 bits est limité à un courant total de 200 mA.
- Le microcontrôleur lui-même peut supporter au maximum un courant de 400 mA.

- ***DDRx :*** *enregistrer le rôle dans la sortie des données*

  Chacun des ports B-D de l'ATmega328P peut être utilisé pour l'entrée ou la sortie. Le registre d'E / S DDRx est utilisé uniquement dans le but de faire d'un port donné un port d'entrée ou de sortie. Par exemple, pour écrire un port, nous écrivons 1 dans le registre DDRx. En d'autres termes, pour envoyer des données à toutes les broches du port B, nous avons d'abord mis 0bi1111111 dans le registre DDRB pour que toutes les broches soient sorties.

- ***DDRx :*** *inscrire le rôle dans la saisie des données*

  Pour faire d'un port un port d'entrée, nous devons d'abord mettre 0 dans le registre DDRx pour ce port, puis introduire (lire) les données présentes sur les broches. Pour vous rappeler que le port est entré lorsque les bits DDR sont à 0, le registre ne peut obtenir que des données, et non le donner. Lorsque DDR contient O, le port obtient des données.

- ***PINx :*** *inscrire le rôle dans la saisie des données*

  Pour lire les données présentes sur les broches, nous devrions lire le registre PINx. Il faut noter que pour lire des données dans la CPU à partir de broches, nous lisons le contenu du registre PINx, alors que pour envoyer des données aux broches, nous utilisons le registre PORTx.

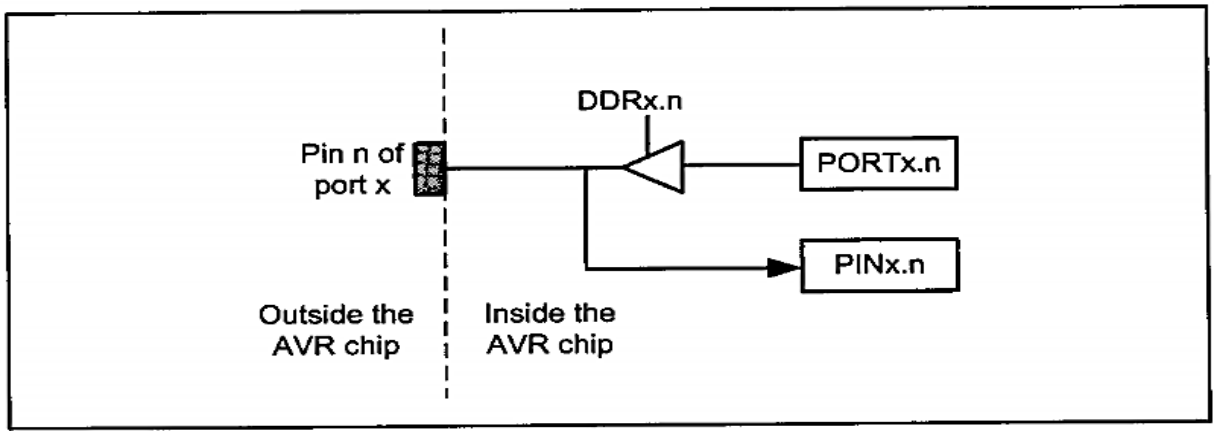


Figure 2.3. Le port d'E / S dans AVR

- ***PORTx:*** *enregistrer le rôle dans l'ecriture des données*

Il y a une résistance de pull-up pour chacune des broches de l'AVR. Si nous mettons 1 dans des bits du registre PORTx, les résistances pull-up sont activées. Dans les cas où rien n'est connecté à la broche ou si les appareils connectés ont une haute impédance, la résistance tire la broche. Voir la figure 2-4.

Si nous mettons O dans les bits du registre PORTx, la résistance de pull-up est inactive.

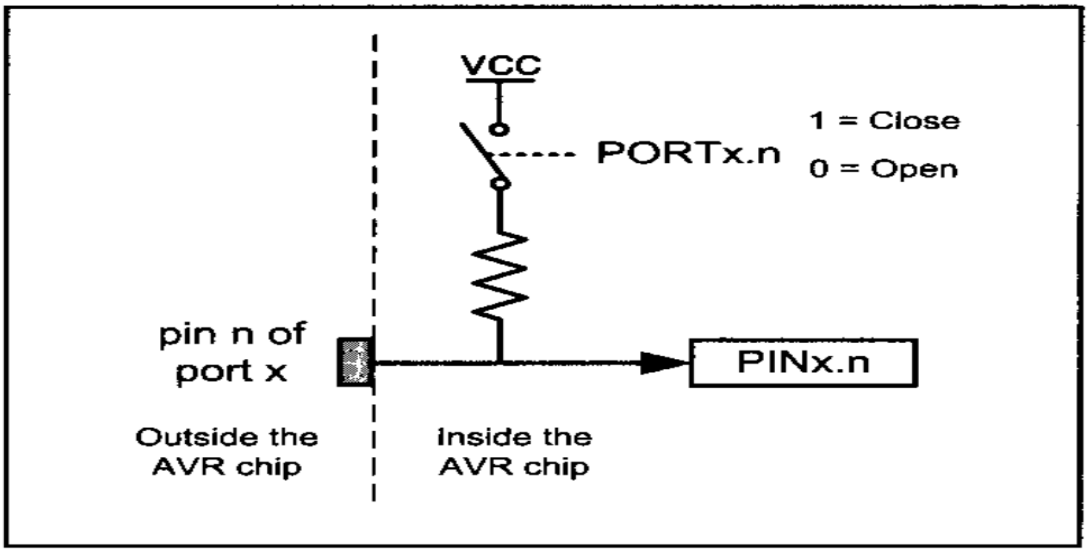


Figure 2.4. La résistance pull-up

Les broches des microcontrôleurs AVR peuvent être dans quatre états différents en fonction des valeurs de PORTx et DDRx, comme indiqué sur la Figure 2.5.

PORTx / DDRx	0	1
0	Input & high impedance	Out 0
1	Input & pull-up	Out 1

Tableau 2.2. Différents états d'une broche dans le microcontrôleur AVR

2.3 Etude des Timers (Compteur programmables) de L'Atmega328p

2.3.1 DESCRIPTION

Le fonctionnement d'un Timer/Compteur repose sur un compteur **TCNTn** (**T**imer **C**ou**NT**er), à 8 ou 16 bits, incrémenté en permanence avec les fronts actifs d'une horloge **clk_{Tn}** et dont le contenu est comparé en permanence au registre **OCRn** (**O**utput **C**ompare **R**egister). Il peut générer des actions sur une sortie **OCn** (**O**utput **C**ompare pin) à chaque égalité entre TCNTn et OCRn ou à chaque débordement, etc.

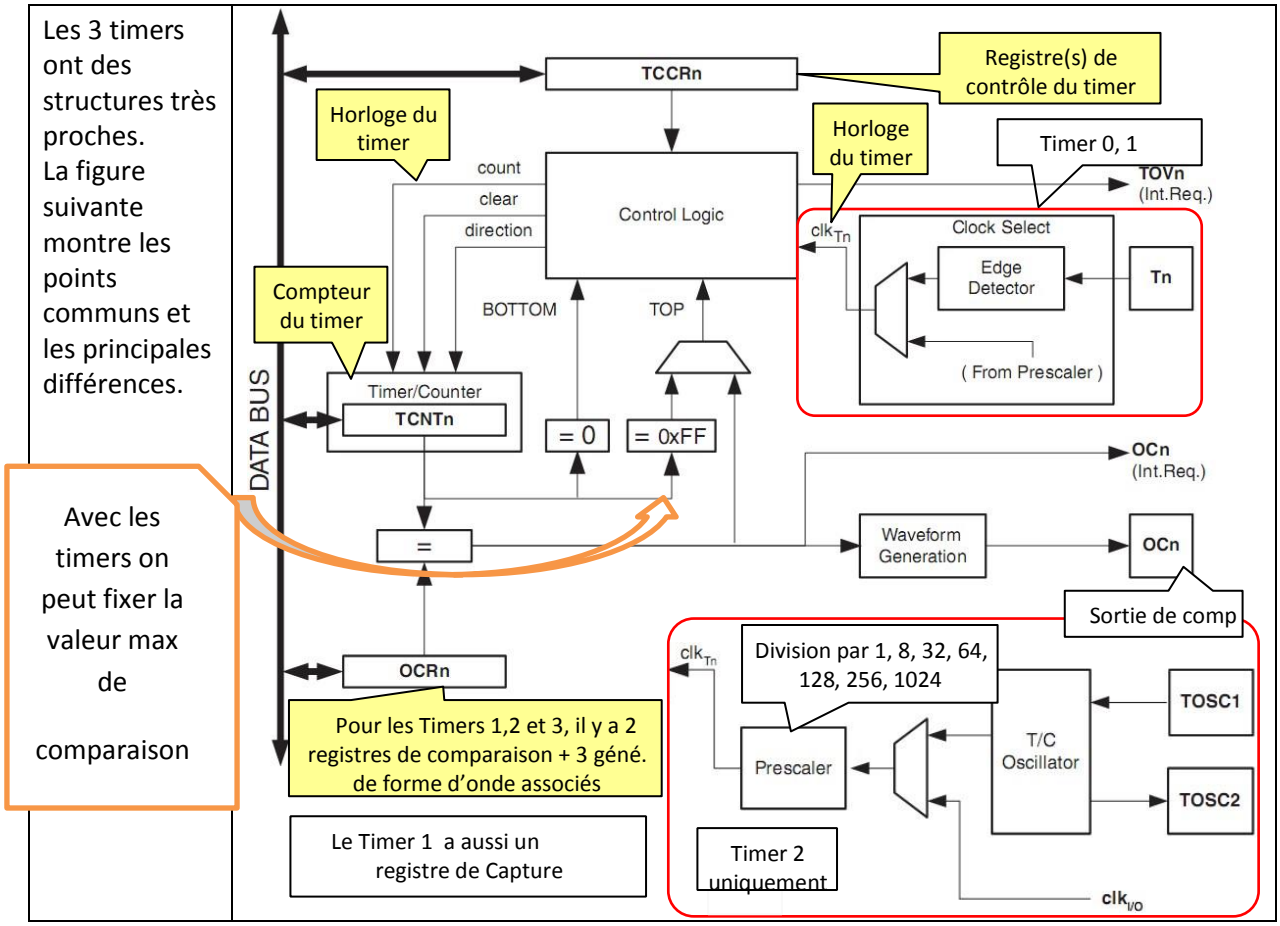


Figure 2.5. Schéma bloc du principe de fonctionnement d'un Timer

Chapitre II : Organisation des entrées/sorties et des Interruptions

Les Timers sont probablement les périphériques complexes les plus couramment utilisés dans un microcontrôleur, leur Data Sheet (documentation technique) peut comprendre jusqu'à 200 pages. Cependant on va essayer de développer les points les plus importants.

L'horloge $\mathbf{clk_{Tn}}$ de TCNTn peut être :

- l'horloge interne, via un pré-diviseur de fréquence (prescaler)
- une horloge externe appliquée sur la broche **Tn**
- aucune horloge (timer non utilisé, permet de diminuer la consommation du µC).

Le ou les registres de commande **TCCRn** permettent de choisir parmi les modes de fonctionnement.

La demande d'interruption (Interrupt Request) positionne un bit à un dans le registre **TIFR** (**T**imer **I**nterrupt **F**lag **R**egister)**.** Toutes les interruptions sont individuellement masquées dans **TIMSK** (**T**imer **I**nterrupt **M**ask **R**egister).

OCR Output Compare Register (registre de comparaison en sortie) :Generation de signaux

ICR Input Capture Register (registre d'entrée de capture) : Mesure de signaux

L'ATmega328p intègre 3 timers possédant les caractéristiques suivantes :

- **Timer 0 et Timer 2** : Résolution : 8 bits, 2 registre de comparaison, utilisation en mode normal, CTC ou PWM (MLI).
- **Timer 1** : Résolution : 16 bits, 2 registres de comparaison, utilisation en mode normal, CTC, PWM et 1 registre de capture.

L'Atmega328p dispose de modules de temporisation/comptage internes suivants, deux compteurs 8 bits (Timer 0 et 2) et d'un compteur 16 bits (Timer1). Ils peuvent être utilisés pour générer des temporisations précises, compter des évènements, mesurer des périodes ou

fréquences de signaux, générer des signaux PWM pour le contrôle en vitesse de moteurs à courant continu par exemple …

Dans tous les cas, chaque événement de comptage conduit à une modification du registre de comptage (+1 ou -1). L'événement de comptage peut être un "tick" de l'horloge du microcontrôleur, ce qui revient à mesurer l'écoulement du temps. L'évenement de comptage peut aussi être un front sur une broche d'entrée du microcontrôleur (les broches T0 et T1)

Fonction de Temporisation : lorsque l'on compte des "ticks" de l'horloge qui cadence le microcontrôleur, on mesure du temps. Les modules Timers permettent de compter les ticks du signal d'horologe ou un signal de fréquence plus faible obtenu par un diviseur appelé prescaler.

Fonction de Comptage : lorsque l'on compte des fronts sur une entrée de comptage (broches T0 ou T1), on utilise alors la fonction "compteur" du module.

Le choix entre fonction de temporisation (avec pré diviseur de fréquence ou non) et fonction de comptage se fait par paramétrage de registres dédiés à la gestion des modules Timers/Counters dans ce cas c'est le registre de contrôle TCCR.

Lorsqu'un Timer est utilisé comme compteur d'évènement, les événements à compter sont appliqués à l'entrée du compteur binaire, et le nombre d'événements survenant est compté. Par exemple, le compteur pourrait être utilisé pour compter le nombre d'objets descendant d'une ligne d'assemblage en appliquant une impulsion à l'entrée du compteur pour chaque objet. À tout moment, le compteur pouvait être lu pour déterminer combien d'objets étaient descendus d'une chaîne de montage.

Les compteurs 0 et 1 évoluent sous contrôle d'une horloge externe ou interne. Dans ce dernier cas l'horloge est un dérivé de l'horloge du quartz comme représenté sur la figure ci-dessous

Les microcontrôleurs AVR fournissent à la fois des compteurs / Timer a 8 bits et 16 bits Dans les deux cas, un problème important pour le programme est de savoir quand le compteur atteint son nombre maximum et se retourne. Dans le cas d'un compteur à 8 bits, cela se produit

lorsque le compte atteint 255, auquel cas l'impulsion suivante fera basculer le compteur à 0. Dans le cas d'un compteur à 16 bits, la même chose se produit à 65 535.

2.3.2 Les Registres de base des Timers

Le tableau suivant donne les différents registres de contrôle associés à chaque timer ; nous verrons le rôle de chaque registre tout en nous limitant à ce qui est vraiment essentiel à connaître pour comprendre le fonctionnement.

Timer 0	Timer 1	Timer 2	Rôle
TCNT0	TCNT1L	TCNT2	Timer (bit 0 à 7)
-	TCNT1H	-	Timer (bit 8 à 15)
TCCR0A	TCCR1A	TCCR2A	Registre de contrôle
TCCR0B	TCCR1B	TCCR2B	Registre de contrôle
-	TCCR1C	-	Registre de contrôle
OCR0A	OCR1AL	OCR2A	Output Compare (bit 0 à 7)
-	OCR1AH	-	Output Compare (bit 8 à 15)
OCR0B	OCR1BL	OCR2B	Output Compare (bit 0 à 7)
-	OCR1BH	-	Output Compare (bit 8 à 15)
-	ICR1L	-	Input Capture (bit 0 à 7)
-	ICR1H	-	Input Capture (bit 8 à 15)
TIMSK0	TIMSK1	TIMSK2	Interrupt Mask
TIFR0	TIFR1	TIFR2	Interrupt Flag

Tableau 2.3. Différents registres de contrôle associes à chaque timer

TIMSK et **TIFR** servent pour que le timer puisse générer des interruptions

Lors de la réinitialisation, le TCNTn contient la valeur zéro par default. Il compte à chaque impulsion. Le contenu des compteurs peut être consulté en utilisant une instruction de lecture du registre TCNTn. Vous pouvez charger une valeur dans le registre TCNTn ou lire sa valeur. Chaque temporisateur possède également un indicateur de depassement TOVn (Timer overflow). Quand le registre compteur TCNTn déborde, son flag TOVn sera activer.

Chaque timer possède également le registre TCCRn (registre de contrôle du compteur) pour définir les modes de fonctionnement. Par exemple, vous pouvez spécifier Timer0 pour fonctionner en tant que timer ou compteur en chargeant des valeurs appropriées dans le TCCRO. Chaque timer possède également un registre OCRn (Registre de comparaison de sortie). Le contenu de l'OCRn est comparé au contenu du TCNTn. Quand ils sont égaux, le drapeau OCFn (Flag de comparaison de sortie) sera défini.

Les registres de temporisation sont situés dans la mémoire du registre d'E/S (voir tableau ci-dessous). Par conséquent, vous pouvez lire ou écrire à partir des registres de minuterie en utilisant les instructions IN et OUT, comme les autres registres d'E / S.

Adresse	Registre	Nbr bits	Contenu des Bits du registre concerne							
0 x 23	PINB	0 – 7	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0 x 2D	PORTD	0 – 7	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0 x 35	TIFR0	0 – 7						OCFB	OCFA	TOV
0 x 36	TIFR1	0 – 7			ICF			OCFB	OCFA	TOV
0 x 44	TCCR0A	0 – 7	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM0
0 x 45	TCCR0B	0 - 7	FOC0A	FOC0B			WGM02		CS0[2:0]	
0 x 46	TCNT0	0 - 7				TCNT0[7:0]				
0 x 84	TCNT1L	0 - 7				TCNT1L[7:0]				
0 x 85	TCNT1H	0 - 7				TCNT1H[7:0]				
0 x 86	ICR1L	0 - 7				ICR1L[7:0]				
0 x 87	ICR1H	0 – 7				ICR1H[7:0]				
0 x B0	TCCR2A	0 – 7	COM2A1	COM2A0	COM2B1	COM2B0			WGM21	WGM20

Tableau 2.4. Les bits du registre

2.3.4 Timer/ COMPTEUR PRESCALER ET SÉLECTEURS D'ENTRÉE

Les unités de minuterie / comptage peuvent utiliser une variété de fréquences internes dérivées de l'horloge du système en tant qu'entrée, ou elles peuvent obtenir leur entrée à partir d'une broche externe. Le registre de commande de compteur de temporisation (TCCRx) associé au temporisateur contient les bits de sélection de compteur (CSx2, CSx1, CSx0) qui contrôlent quelle entrée est utilisée avec un compteur spécifique. La Figure 2.9 illustre la configuration du sélecteur de présélection et d'entrée pour un registre de contrôle du compteur de temporisation utilisé dans la plupart des microcontrôleurs AVR.

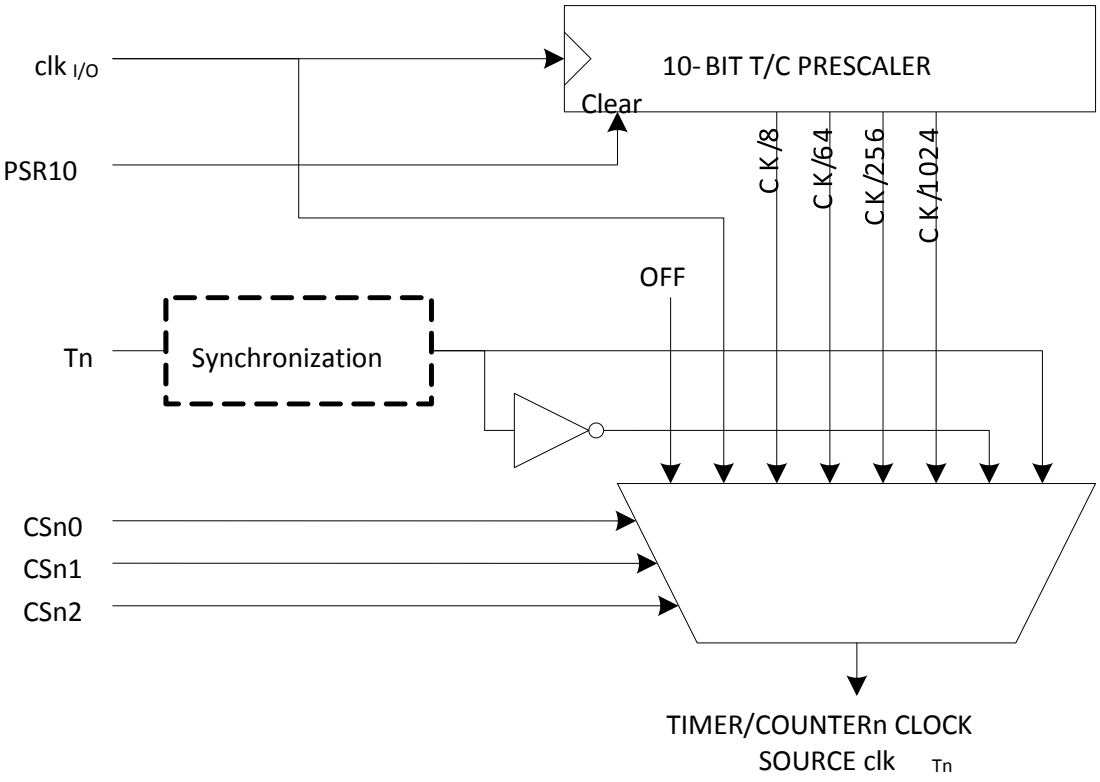
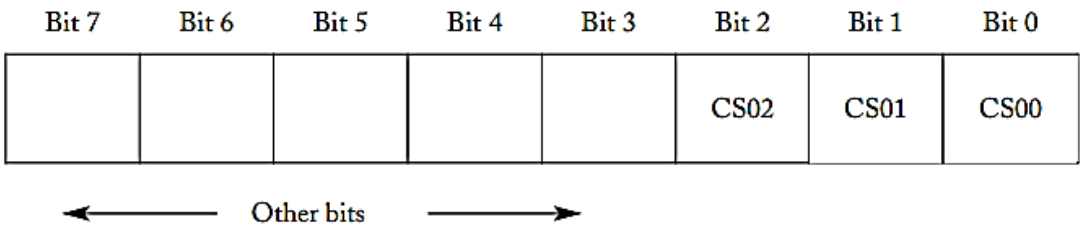


Figure 2.6. Timer 0 Prescaler / Sélecteur

Le Timer 0 est généralement un Timer de 8 bits, mais cela varie selon le type de processeur spécifique. Il est capable des fonctions habituelles de minuterie / comptage, mais sert le plus souvent à créer une base de temps ou un tique pour le programme.

Le registre 0, TCCR0, contrôle la fonction du Timer 0 en sélectionnant la source d'horloge appliquée au Timer 0.

La Figure 2.11 montre les définitions de bits du prédicteur d'horloge pour TCCR0. D'autres bits de TCCR0 commandent des fonctions additionnelles de Timer 0 d'une manière similaire aux bits de contrôle pour Timer 1.



CS02	*CS01*	*CS00*	*Interrupt function*
0	0	0	Stop, Timer 0 is stopped
0	0	1	System Clock, CK
0	1	0	System Clock / 8, CK/8
0	1	1	System Clock / 64, CK/64
1	0	0	System Clock / 256, CK/256
1	0	1	System Clock / 1024, CK/1024
1	1	0	External Pin T0, counts a falling edge
1	1	1	External Pin T0, counts a rising edge

Figure 2.7. TCCR0 les Définitions de Prescaler

Une coche de programme, comme la coche d'une horloge, fournit un événement de synchronisation très précis. Le schéma général est qu'un nombre est sélectionné et chargé dans la Timer. La Timer compte de ce nombre jusqu'à 255 et se retourne. Chaque fois qu'il survole, il crée une interruption. La routine de service d'interruption recharge le même numéro dans la minuterie, exécute toutes les activités urgentes qui peuvent être nécessaires, puis retourne au programme. Le cycle se répète alors, le compteur comptant à partir du nombre qui a été

chargé à 255, et se retourne, créant une autre interruption. L'interruption se produit donc régulièrement lorsque chaque période s'est écoulée. Le nombre chargé dans le compteur détermine la longueur de la période. Plus le chiffre est bas, plus la minuterie prendra de temps pour atteindre 255 et se renversera, et plus la période de la coche sera longue.

- **TCCR0 (registre de contrôle de Timer / compteur)**

TCCR0 est un registre de 8 bits utilisé pour le contrôle de Timer0.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
écriture/lecture	E	E/L	E/L	E/L	E/L	E/L	E/L	E/L	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.8. Timer/compteur contrôle registre

- **CS02:CS00 (Tlmer0 clock source)**

Ces bits dans le registre TCCR0 sont utilisés pour choisir la source d'horloge. Si CS02; CS00 = 000, le compteur est arrêté. Si CS02-CS00 a des valeurs comprises entre 001 et 101, l'oscillateur est utilisé comme source d'horloge et le timer/ compteur agit comme un timer. Dans ce cas, les minuteurs sont souvent utilisés pour la génération de retard.

Si CS02-CS00 est 110 ou 111, la source d'horloge externe est utilisée et agit comme un compteur.

- D7 : FOC0 — Force compare match: Ceci est un bit en écriture seule, qui peut être utilisé lors de la génération d'une onde. En écrivant 1, le générateur d'ondes agit comme si une comparaison avait eu lieu.

D6 WGM00	D3 WGM01	bits de sélection de mode Timer 0
0	0	Normal
0	1	CTC (Clear Timer on Compare Match)
1	0	PWM, phase correcte
1	1	PWM rapide

D5 COM01	D4 COM00	Compare Output Mode: Ces bits contrôlent le générateur de forme d'onde

D2 CS02	D1 CS01	D0 CS00	Timer0 clock selector
0	0	0	No clock source (Timer)Counter stopped)
0	0	1	Clk (pas de Prescale)
0	1	0	Clk/8
0	1	1	Clk/64
1	0	0	Clk/256
1	0	1	Clk/1024
1	1	0	Source d'horloge externe sur la broche TO. Horloge sur le front descendant
1	1	1	Source d'horloge externe sur la broche TO. Horloge sur le front montant.

- **WGM01:00**

Timer0 peut fonctionner dans quatre modes différents: Normal, PWM phase correcte, CTC et PWM rapide. Les bits WGMO1 et WGMOO sont utilisés pour en choisir un.

- **TOV0 (Timer0 Overflow)**

Le drapeau (flag) est activé lorsque le compteur déborde, passant de $ FF à $ 00, lorsque le temporisateur revient de $ FF à 00, le fanion TOV0 est mis à 1 et il reste activé jusqu'à ce que le logiciel l'efface. Pour l'effacer il faut y écrire 1. En effet cette règle s'applique à s tous les drapeaux de la puce AVR. En AVR, quand nous voulons effacer un drapeau donné d'un registre, nous lui écrivons 1 et 0 aux autres bits.

- **TCNT0 (Timer/Counter Register)**

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.9. Timer/counter register

C'est là que réside le compteur de 8 bits de la minuterie. La valeur du compteur est stockée ici et augmente / diminue automatiquement. Les données peuvent être lues / écrites à partir de ce registre.

- **TIMSK (Timer/Counter Interrupt Mask)**

C'est un registre commun pour tous les trois timers, est utilisé pour contrôler quelles interruptions sont "valides" en définissant leurs bits dans TIMSK et pour déterminer quelles interruptions sont actuellement en attente.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.10. Timer/counter interrupt mask

- Bit 7 - OCIE2: Timer/Counter2 Output Compare Match Interrupt Enable

Lorsque le bit OCIE2 est défini (un) et que le bit I du registre d'état est défini (un), l'interruption de comparaison de minuterie / compteur 2 est activée.

- Bit 6 - TOIE2: Timer/Counter2 Overflow Interrupt Enable

Lorsque le bit TOIE2 est défini (un) et que le bit I dans le registre d'état est défini (un), l'interruption de temporisation / de dépassement de compteur2 est activée.

- Bit 5 - TICIE1: Timer/Counter1 Input Capture Interrupt Enable

Lorsque le bit TICIE1 est défini (un) et que le bit I dans le registre d'état est défini (un), l'interruption d'événement de capture d'entrée Timer / Counter1 est activée.

- Bit 4 - OCIE1A: Timer/Counter1 Output Compare A Match Interrupt Enable

Lorsque le bit OCIE1A est défini (un) et que le bit I dans le registre d'état est défini (un), l'interruption Timer / Counter1 Compare A Match est activée.

- Bit 3 - OCIE1B: Timer/Counter1 Output CompareB Match Interrupt Enable

Lorsque le bit OCIE1B est défini (un) et que le bit I dans le registre d'état est défini (un), l'interruption Timer / Counter1 Compare B Match est activée.

- Bit 2 - TOIE1: Timer/Counter1 Overflow Interrupt Enable

Lorsque le bit TOIE1 est défini (un) et que le bit I dans le registre d'état est défini (un), l'interruption Timer / Counter1 Overflow est activée.

- Bit 1 - OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable

- Bit 0 - TOIE0: Timer/Counter0 Overflow Interrupt Enable

Lorsque le bit TOIE0 est défini (un) et que le bit I dans le registre d'état est défini (un), l'interruption Timer / Counter0 Overflow est activée.

- **TIFR (Timer/counter Interrupt Flag Register)**

C'est un registre commun pour tous les trois timers, est utilisé pour contrôler les interruptions qui sont "valides" en définissant leurs bits dans TIMSK et pour déterminer quelles interruptions sont actuellement en attente.

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.11. Timer/counter interrupt flag register

- Bit 7 - OCF2: Output Compare Flag 2

Le bit OCF2 est défini (un) lorsqu'une correspondance de comparaison se produit entre le timer / compteur2 et les données dans OCR2, OCF2 est effacé par le matériel lors de l'exécution du vecteur de gestion d'interruption correspondant. Alternativement, OCF2 est effacé en écrivant une logique au Flag. Lorsque le bit I dans SREG, et OCIE2, et les OCF2 sont réglés (un), l'interruption de comparaison Timer / Compteur2 est exécutée.

- Bit 6 - TOV2: Timer/Counter2 Overflow Flag

Le TOV2 est réglé (un) lorsqu'un dépassement se produit dans Timer / Counter2. TOV2 est effacé par le matériel lors de l'exécution du vecteur de gestion des interruptions correspondant. Alternativement, TOV2 est effacé en écrivant une logique au Flag.

- Bit 5 - ICF1: Input Capture Flag 1

Le bit ICF1 est mis (un) pour marquer un événement de capture d'entrée, indiquant que la valeur Timer / Counter1 a été transférée au registre de capture d'entrée - ICR1. ICF1 est effacé par le matériel lors de l'exécution du vecteur de gestion des interruptions correspondant.

- Bit 4 - OCF1A: Output Compare Flag 1A

Le bit OCF1A est défini (un) lorsqu'une comparaison se produit entre le timer / compteur 1 et les données dans le registre de comparaison de sortie OCR1A 1A. OCF1A est effacé par le matériel lors de l'exécution du vecteur de gestion des interruptions correspondant.

- Bit 3 - OCF1B: Output Compare Flag 1B

Le bit OCF1B est défini (un) lorsqu'une correspondance de comparaison se produit entre le Timer/ compteur 1 et les données dans OCR1B - Registre de comparaison de sortie 1B. OCF1B est effacé par le matériel lors de l'exécution du vecteur de gestion des interruptions correspondant.

- Bit 2 - TOV1: Timer/Counter1 Overflow Flag

Le bit TOV1 est défini (un) lorsqu'un dépassement se produit dans Timer / Counter1. TOV1 est effacé par le matériel lors de l'exécution du vecteur de gestion des interruptions correspondant.

- Bit 1 - OCF0: Output Compare Flag 0

- Bit 0 - TOV0: Timer/Counter0 Overflow Flag

Le bit TOV0 est défini (un) lorsqu'un dépassement se produit dans Timer / Counter0. TOV0 est effacé par le matériel lors de l'exécution du vecteur de gestion des interruptions correspondant.

2.3.5 Timer1

Le Timer 16 bits est un périphérique beaucoup plus polyvalent et complexe que les Timers 8 bits typiques. En plus de cela le Timer 1 contient un registre de capture d'entrée de 16 bits et deux registres de comparaison de sortie de 16 bits. Le registre de capture d'entrée est utilisé pour mesurer les largeurs d'impulsion ou les temps de capture. Les registres de comparaison de sortie sont utilisés pour produire des fréquences ou des impulsions sur une broche de sortie du microcontrôleur (OC1A et OC1B).

Depuis Timer1 son registre de 16 bits est divisé en deux octets. Ceux-ci sont appelés TCNT1L (Timer1 low octet) et TCNTIH (Timer1 high octet), Timer1 possède également deux registres de contrôle nommés TCCR1A et TCCRIB. Le bit d'indicateur TOV1 (dépassement de temporisateur) passe à l'état HAUT lorsque le dépassement se produit. Timer1 a également les options de prédicateur de 1: 1, 1: 8, 1:64, 1: 256 et 1: 1024. Voir la Figure 2.16 pour le schéma de principe de Timer1.

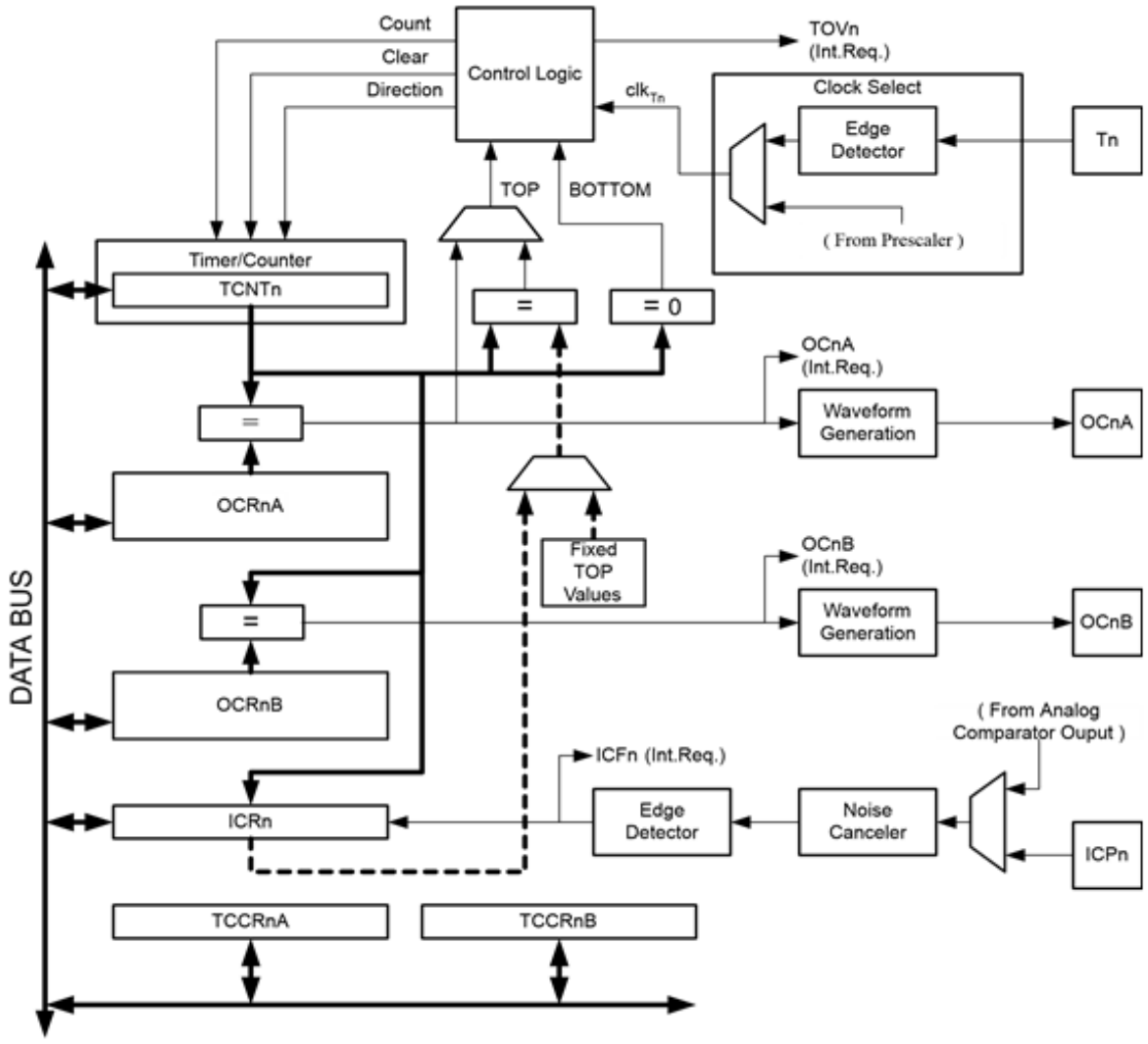


Figure 2.12. Schéma interne du Timer1

Il y a deux registres OCR dans Timer1: OCR1A et OCR1B. Il existe deux indicateurs distincts pour chacune. Les registres OCR, qui agissent indépendamment les uns des autres. Chaque fois que TCNTI est égal à OCRIA, le drapeau OCF1A sera mis à l'horloge suivante. Lorsque TCNT est égal à OCRIB, le flag OCFIB sera mis à l'heure suivante. Comme Timer1 est un temporisateur de 16 bits, les registres OCR sont également des registres de 16 bits et sont constitués de deux

registres de 8 bits. Par exemple, l'OCRIA est composé d'OCRIAH (OCRIA high octet) et OCRIAL (OCRIA low byte).

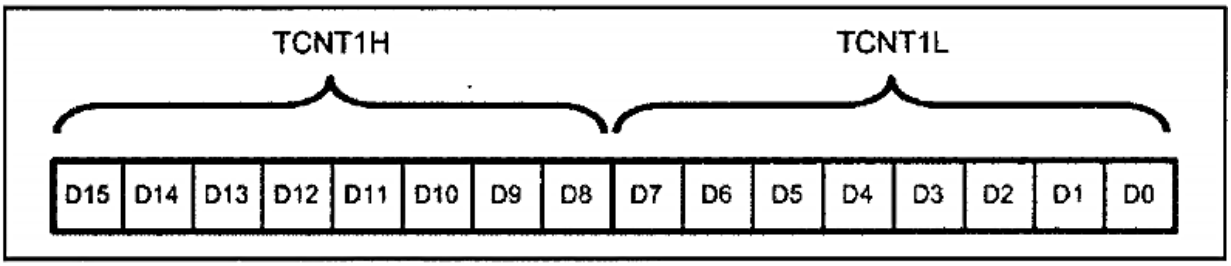


Figure 2.13. Timer1 registres hauts et bas

Le timer 1 est toujours un compteur ascendant binaire dont la vitesse de comptage ou les intervalles de temporisation dépendent du signal d'horloge appliqué à son entrée, tout comme le temporisateur 0 l'était. Comme pour tous les périphériques du microcontrôleur, le timer 1 est contrôlé par un registre de contrôle.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ICNC1	ICES1		WGM 13	WGM 12	CS12	CS11	CS10

Bit	***Function***
ICNC1	Input Capture Noise Canceller (1 = enabled)
ICES1	Input Capture Edge Select (1 = rising edge, 0 = falling edge)
WGM 1x	Output waveform control. See TCCR1A
CS12	Counter Input Select Bits Exactly the same definition as for Timer 0
CS11	
CS10	

Figure 2.14. TCCR1B définition de bit

Les bits de sélection du compteur TCCR1B commandent l'entrée à la temporisation 1 exactement de la même manière que les bits de comptage du temporisateur 0. En effet, les trois bits de sélection fournissent des signaux d'horloge absolument identiques à ceux du temporisateur 0.

Bit	7	6	5	4	3	2	1	0
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TOV0 D0 Timer0 overflow flag bit
0 = Timer0 did not overflow.
1 = Timer0 has overflowed (going from $FF to $00).

OCF0 D1 Timer0 output compare flag bit
0 = compare match did not occur.
1 = compare match occurred.

TOV1 D2 Timer1 overflow flag bit

OCF1B D3 Timer1 output compare B match flag

OCF1A D4 Timer1 output compare A match flag

ICF1 D5 Input Capture flag

TOV2 D6 Timer2 overflow flag

OCF2 D7 Timer2 output compare match flag

Figure 2.15. TIFR {Timer/Counter Interrupt Flag Register)

Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

COM1A1:COM1A0 D7 D6 Compare Output Mode for Channel A (discussed in Section 9-3)

COM1B1:COM1B0 D5 D4 Compare Output Mode for Channel B (discussed in Section 9-3)

FOC1A D3 Force Output Compare for Channel A (discussed in Section 9-3)

FOC1B D2 Force Output Compare for Channel B (discussed in Section 9-3)

WGM11:10 D1 D0 Timer1 mode (discussed in Figure 9-18)

Figure 2.16. TCCR1A (Timer 1 Control) Registre

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ICNC1 D7 Input Capture Noise Canceler
0 = Input Capture is disabled.
1 = Input Capture is enabled.

ICES1 D6 Input Capture Edge Select
0 = Capture on the falling (negative) edge
1 = Capture on the rising (positive) edge

D5 Not used

WGM13:WGM12 D4 D3 Timer1 mode

CS12:CS10

D2	D1	D0	
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (no prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Figure 2.17. TCCR1B (Timer 1 Control) Registre

Il existe également un registre auxiliaire nommé ICR1, qui est utilisé dans des opérations telles que la capture. ICR1 est un registre de 16 bits fait de ICR1H et ICR1L.

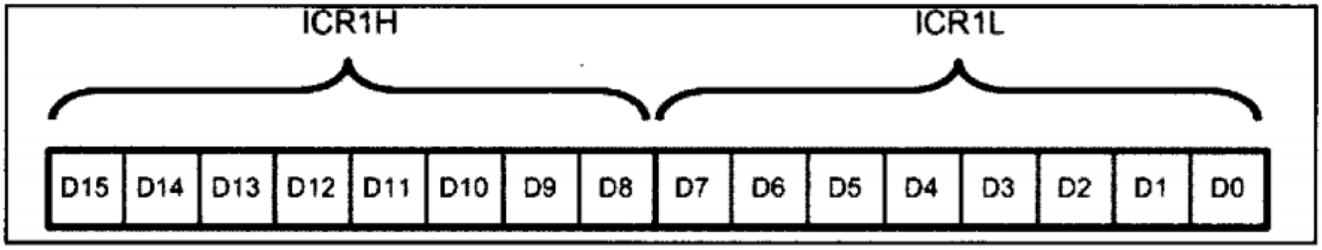


Figure 2.18. Registre de capture d'entre (ICR) pour Timer1

- **TIFR (Timer/counter Interrupt Flag Register)/TIMSK (Timer/Counter Interrupt Mask)**

C'est un registre commun pour tous les trois timers, est utilisé pour contrôler quelles interruptions sont "valides" en définissant leurs bits dans TIMSK et pour déterminer quelles interruptions sont actuellement en attente.

2.3.6 Timer 2

Le temporisateur 2 est généralement (en fonction d'un microcontrôleur spécifique utilisé) un timer / compteur de 8 bits avec une comparaison de sortie et des fonctions PWM similaires au temporisateur 1.

La différence la plus intéressante avec le Timer 2 est qu'il peut utiliser un cristal séparé de l'horloge du système comme source d'horloge. La sélection du cristal externe en tant que source d'horloge pour le temporisateur 2 est effectuée en réglant le bit AS2 dans le registre d'état asynchrone (ASSR).

Le réglage du bit AS2 permet au Timer 2 d'utiliser le cristal externe comme source d'horloge. Cela signifie que la source d'horloge du temporisateur 2 fonctionne de manière asynchrone avec l'horloge du microcontrôleur. Les trois autres bits du registre ASSR sont utilisés par le programmeur pour s'assurer que les données ne sont pas écrites dans les registres du timer2 au même moment où le matériel met à jour les registres du timer2.

Un registre de contrôle unique, TCCR2, contrôle le fonctionnement du timer2.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

Bit	***Description***
FOC2	Force a Compare match in non-PWM mode.
WGM20	Setting this bit enables the Timer 2 PWM function.
COM21	These two bits set the output compare mode function. The bit definitions are identical to the COM1x1 and COM1x0 bits of timer 1.
COM20	
WGM21	Set to enable counter clear on compare match.
CS22	Counter prescaler select bits for Timer 2. See AVR spec. for details.
CS21	
CS20	

Figure 2.19. TCCR2 Définitions de Bit

NOTE : Les registres de Timer2 discuté avec résumé pour plus de détails, retour au registre de Timer1 et Timer0

- **TCNT2 (Timer/Counter Register)**
  La valeur de timer est stockée. Étant donné que TIMER2 est un temporisateur à 8 bits, ce registre a une largeur de 8 bits.

Bit	7	6	5	4	3	2	1	0	
	TCNT2[7:0]								TCNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.20. TCNT (Timer/Counter register)

- **TCCR2** Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.21. TCCR2

- **TIFR (Timer/counter Interrupt Flag Register)/TIMSK (Timer/Counter Interrupt Mask)**

C'est un registre commun pour tous les trois timers, est utilisé pour contrôler quelles interruptions sont "valides" en définissant leurs bits dans TIMSK et pour déterminer quelles interruptions sont actuellement en attente.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.22. TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.23. TIFR

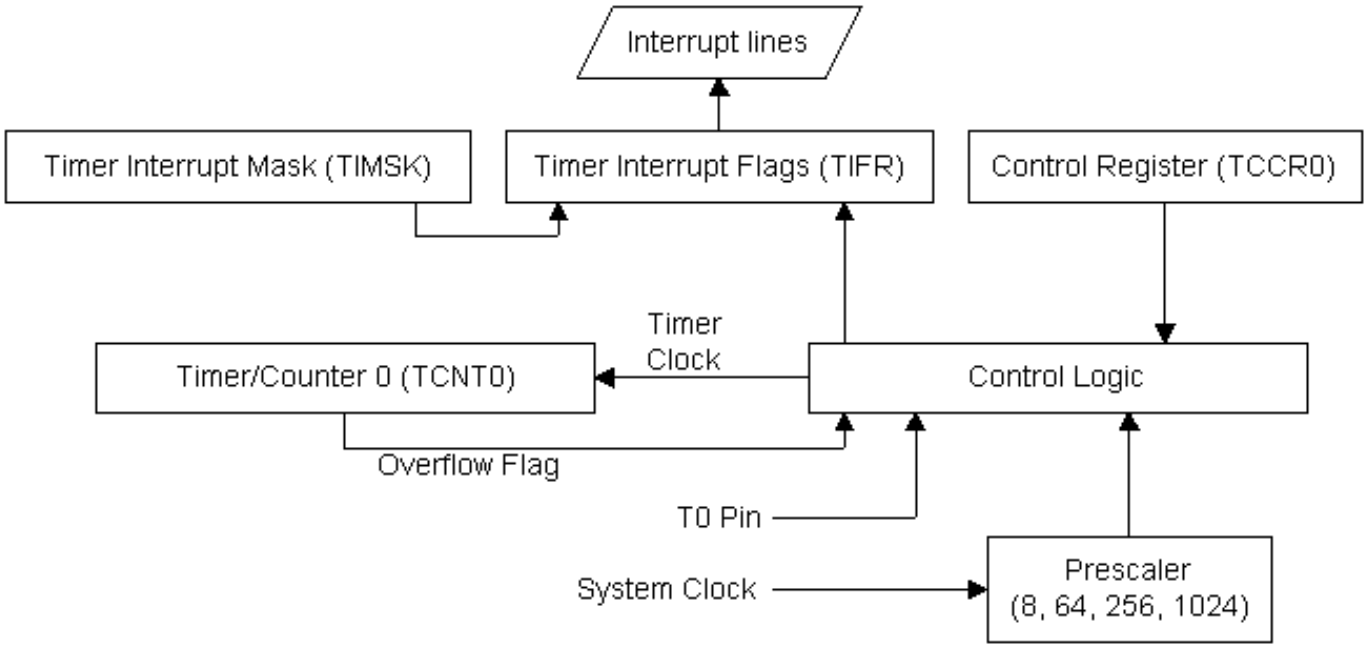


Figure 2.24. Timer 8 bits Diagramme

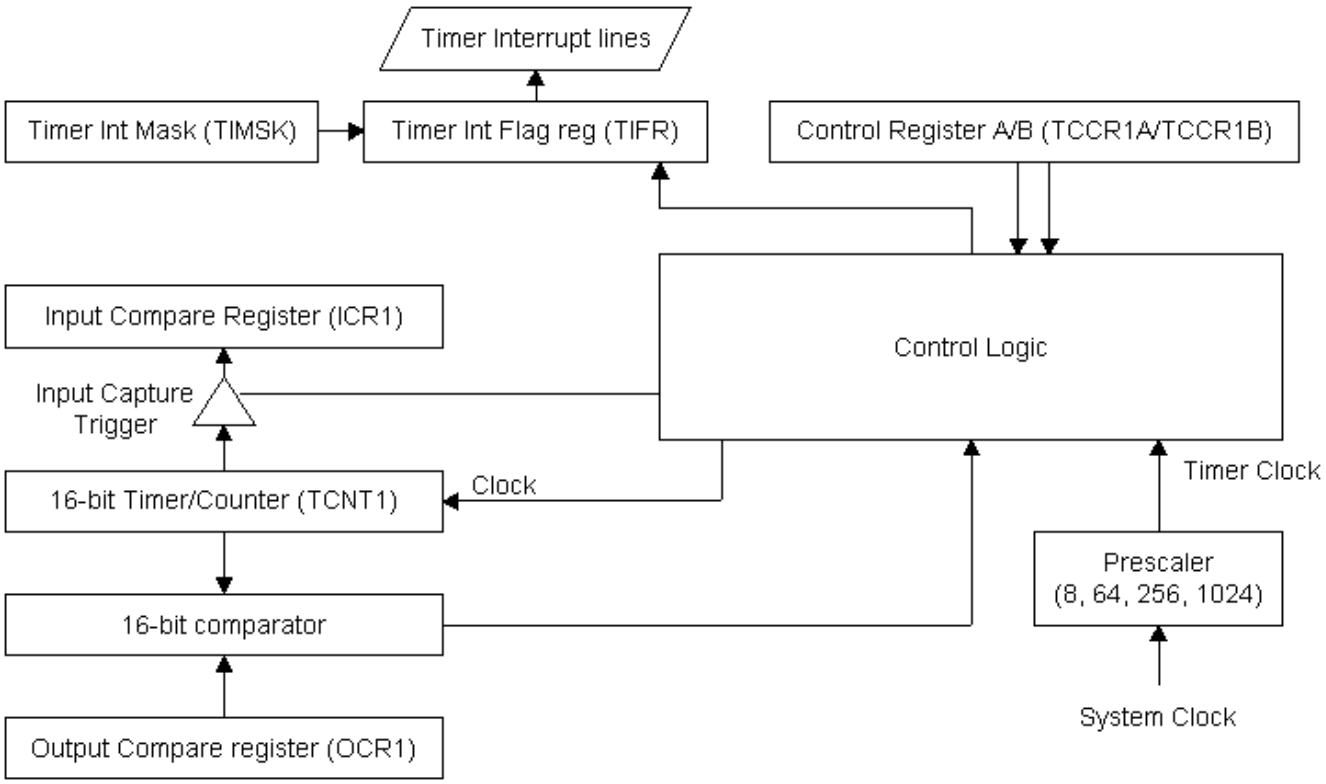


Figure 2.25. Timer 16 Bits Diagramme

2.4 Les Modes de Timer

2.4.1 Mode normal

Le mode normal est le mode de fonctionnement le plus simple.

Dans ce mode, le contenu du Timer / compteur s'incrémente avec chaque horloge. Il compte jusqu'à ce qu'il atteigne son maximum de 0xFF (0xFFFF). Lorsqu'il passe de 0xFF (0xFFFF) à 0x00 $(0x0000)_1$, il positionne un bit de flag haut appelé TOV (Timer Overflow). Ce flag de minuterie peut être surveillé.

Donc en resume **TCNTn** est incrémenté à chaque front actif de $\mathbf{clk_{Tn}}$. **TCNTn** évolue de **0** à **FF** (ou **FFFF**). Au front suivant, **TCNTn** déborde et le drapeau TOVn est mis à 1, générant une interruption si elle a été autorisée. **TCNTn** repart de 0 et le cycle recommence.
Si le drapeau **TOVn** est utilisé (sans interruption), il doit être remis à 0 par le programme utilisateur.

0xFF (0x00) Pour le 8 bits registre, 0x0000(0xFFFF) Pour le 16 Bits registre.

- Incréments de minuteries.
- enrouler autour de TOP = 0xFF.
- Commence à nouveau à 0.
- Flag d'interruption TOV0 défini lorsque TCNT0 est réinitialisé à 0.
- Utile pour générer des interruptions tous les N unités de temps.
- Utile pour générer une interruption en N unités de temps.
- Définissez TCNT0 sur une valeur initiale (255 – N).

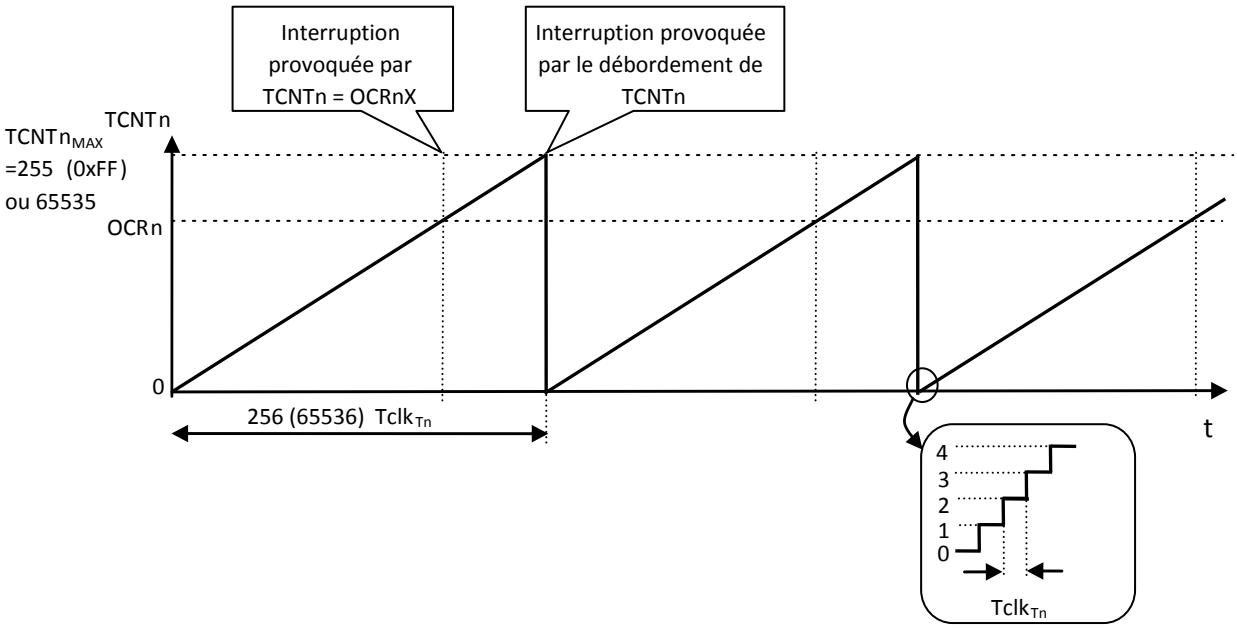


Figure 2.26. Mode de fonctionnement Normal du timer (comptage)

Une nouvelle valeur peut être écrite n'importe quand dans le registre **TCNTn**.

Le Timer combiné avec l'unité de comparaison **OCRn** peut être employé pour produire une interruption au bout d'un temps donné.

Durée comptage (DC) :

TCNTn 8 bits, F quartz = 8 MHz

Sans prédivision pour l'horloge du timer : $Tclk_{TN} = Tclk_{I/O}$: DC = 256 / 8 MHz = 32 µs → F = 30 KHz Avec prédivision par 8 pour l'horloge du timer : $Tclk_{TN} = 8\ Tclk_{I/O}$: DC = 8*256 / 8 MHz = 256 µs → F = 3,9 KHz.

2.4.2 Mode de capture d'entrée

La mesure d'une période avec Minuterie 0 implique le démarrage de la timer au début de l'événement, l'arrêt de la timer à la fin de l'événement, et enfin la lecture de l'heure de l'événement à partir du registre du compteur de minuterie. Le même travail avec le minuteur 1 est géré différemment car le timer 1 est toujours en cours d'exécution. Pour mesurer un

événement, l'heure sur Timer 1 est capturée ou maintenue au début de l'événement, l'heure est également capturée à la fin de l'événement, et les deux sont soustraits pour trouver le temps nécessaire pour que l'événement se produise.

Ces tâches sont gérées par le registre de capture d'entrée (ICR1).

ICR1 est un registre de 16 bits (composé de ICR1H et ICR1L) qui capturera la lecture réelle de la timer 1 lorsque le microcontrôleur reçoit un certain signal. Le signal qui provoque une capture peut être un front montant ou descendant appliquer à la broche de capture d'entrée.

2.4.3 Mode de capture de sortie

Est utilisé par le microcontrôleur pour produire des signaux de sortie. Les sorties peuvent être des ondes carrées ou asymétriques, et elles peuvent varier en fréquence ou en symétrie.

Le programme charge un registre de comparaison de sortie. La valeur dans le registre de comparaison de sortie est comparée à la valeur dans le registre de compteur / compteur, et une interruption se produit lorsque les deux valeurs correspondent. Cette interruption agit comme un réveil pour qu'un processeur exécute une fonction par rapport au signal qu'il produit, exactement quand cela est nécessaire.

2.4.4 Mode modulateur de largeur d'impulsion(PWM)

Le mode (PWM) est l'un des nombreux moyens de fournir une conversion numérique-analogique. PWM est le schéma dans lequel le cycle de fonctionnement d'une sortie d'onde carrée du microcontrôleur est varié pour fournir une sortie CC variable en filtrant la forme d'onde de sortie réelle pour obtenir le courant continu moyen. La Figure 2.30 illustre ce principe.

Comme la montre à la Figure 2.30, la variation du rapport cyclique ou de la proportion du cycle élevé fait varier la tension continue moyenne de la forme d'onde. La forme d'onde est

ensuite filtrée et utilisée pour contrôler les dispositifs analogiques, créant un convertisseur numérique-analogique (DAC).

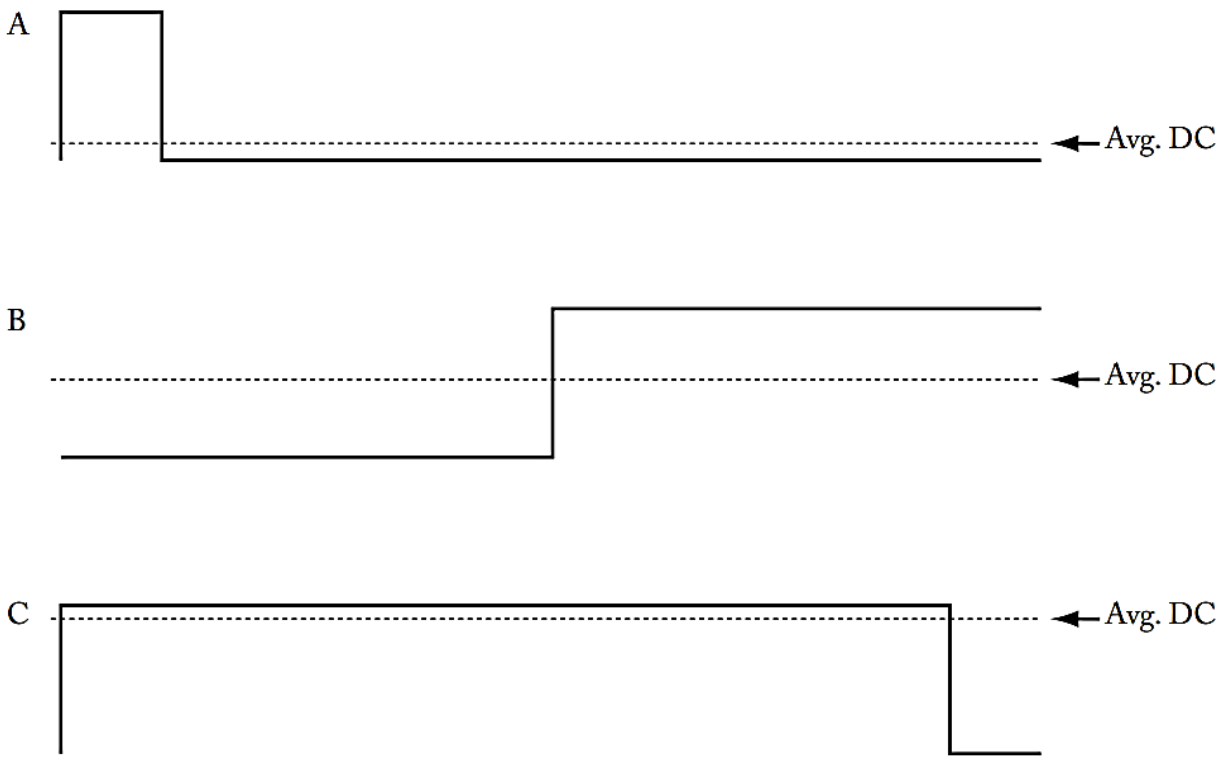


Figure 2.27. PWM forme d'onde

2.4.5 Effacer sur le mode de comparaison de correspondance (CTC)

En mode CTC, le timer est incrémentée avec une horloge. Mais elle compte jusqu'à ce que le contenu du registre TCNT devienne égal au contenu de l'OCR (comparer la correspondance) le registre **OCRn** est employé pour définir le modulo du compteur, Le compteur est incrémenté et lorsque sa valeur est égale à **OCRn**, le compteur est remis à 0 et le drapeau **OCFn0** (**O**utput **C**ompare **F**lag) est mis à 1 pour déclencher une éventuelle interruption ; alors, la minuterie sera effacée et le Flag OCF sera mis à l'heure suivante.

L'indicateur **OCF** est situé dans le registre TIFR.

L'**OCRn** définit la valeur supérieure pour le compteur donc son modulo.

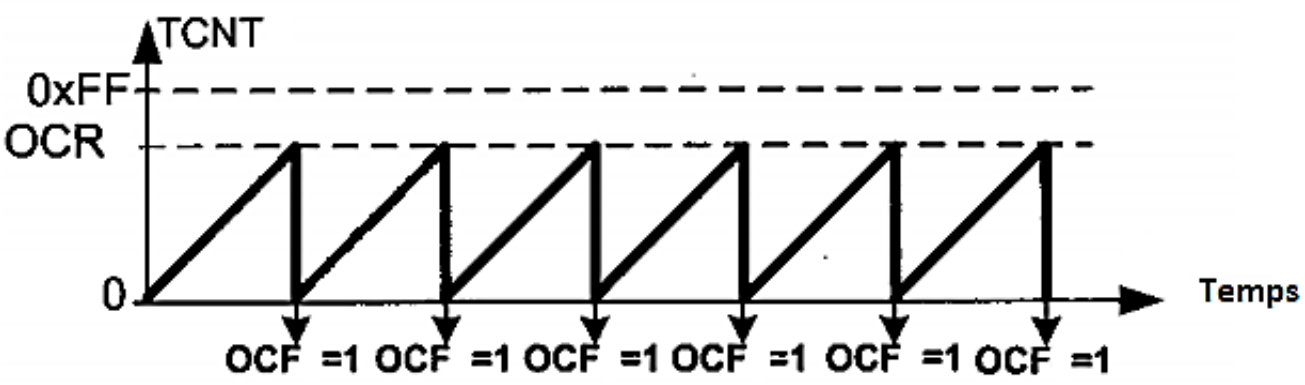


Figure 2.28. Timer/compteur CTC mode

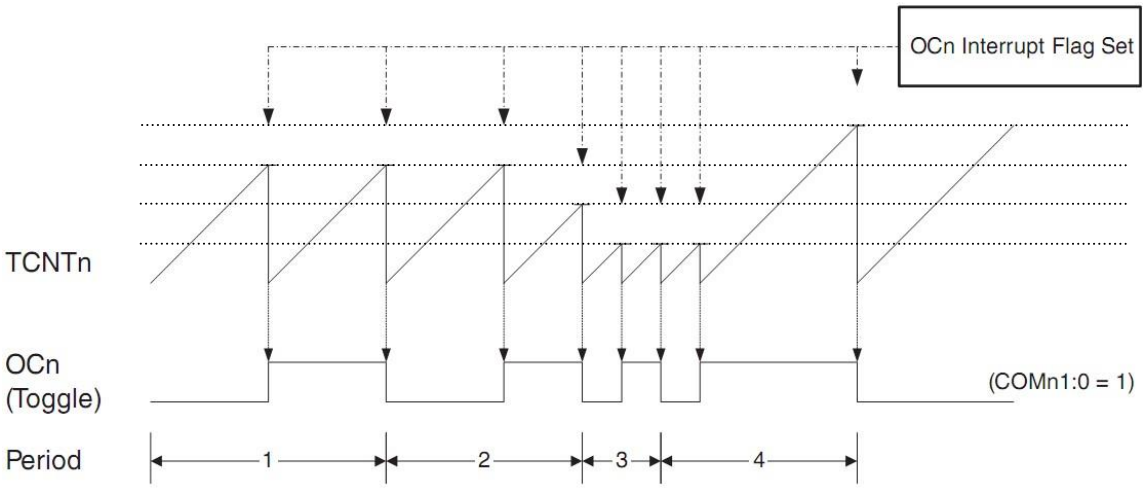


Figure 2.29. Mode de fonctionnement CTC du timer

Le drapeau **OCFn** est mis à 0 lors de l'exécution du programme de traitement de l'interruption.

La durée entre 2 interruptions est égale à : TclkTn * (OCRn + 1). *Le +1 n'est pas justifié dans la doc*.

Chapitre III : Réalisation pratique de la fréquence mètre

3.1 Introduction

Dans ce chapitre nous parlera au fréquencemètre et comment de fonctionne, et ensemble les précédent donné pour faire cette appareil, qui dépendre sur la programmation C AVR avec le logiciel codevision et l'ATmega 328P.

3.2 Principe de fonctionnement du fréquencemètre numérique

Le fréquencemètre peut être représenté de la façon suivante, en réalité ce fréquencemètre est constitue de trois modules :

1. La base de temps qui fixe la durée du comptage (oscillateur + diviseur de fréquence)
2. Le circuit de comptage compte le nombre d'impulsions du signal (mise en forme + compteur)
3. L'affichage constitue du décodeur (qui n'apparait pas dans le schéma) + l'afficheur

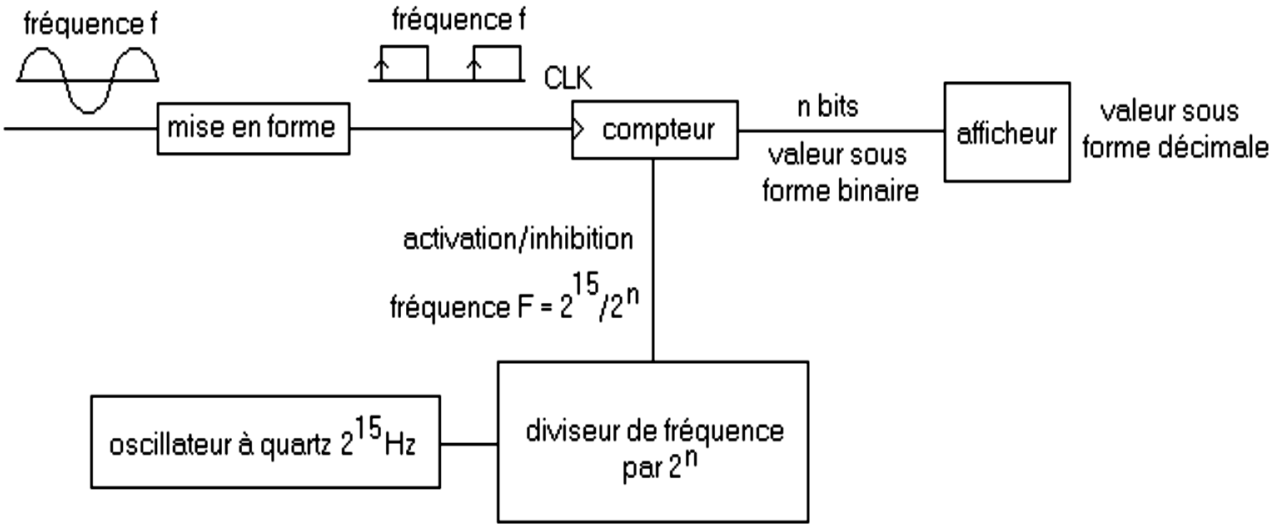


Figure 3.1 . Schéma de principe d'un fréquencemètre numérique

Un fréquencemètre est un instrument de mesure destiné à afficher la fréquence d'un signal périodique, l'appareil est principalement un compteur d'occurrences d'une transition caractéristique du signal entrant.

1. Utilise des impulsions de même fréquence pour faire la mesure.
2. Un oscillateur aussi stable que possible appelé base de temps fournit la référence à laquelle comparer les fréquences.
3. La mesure peut se faire :
   - soit en comptant les impulsions issues de l'entrée dans un temps donné (correspondant à un nombre déterminé de périodes de la base de temps). On obtient alors directement la fréquence.
   - soit en comptant le nombre de périodes de la base de temps dans l'intervalle entre un nombre déterminé d'impulsions issues du signal d'entrée. On obtient un multiple de la période du signal à mesurer, à partir duquel il faut calculer la fréquence.
   - soit, indirectement, en mélangeant un signal dérivé des transitions caractéristiques à un autre, de fréquence proche, constitué à partir de la base de temps, et en mesurant ensuite, par l'un ou l'autre des moyens précédents, la fréquence des battements qui s'ensuivent.

- Fréquence : est le nombre de fois qu'un phénomène périodique se reproduit par unité de mesure

- La mise en forme est chargée de transformer un signal périodique en un signal de même période en forme de créneaux (attention, cette opération ne marche pas pour tous les signaux… on peut citer par exemple les signaux qui sont toujours de même signe). Nous verrons qu'une mise en forme peut conduire à une fréquence erronée quand le signal passe plus de deux fois par zéro au cours d'une période (signal cardiaque, …).

- Le compteur voit sa sortie en binaire incrémentée de 1 à chaque front montant reçu sur son entrée d'horloge (sortie de la mise en forme). - L'ensemble oscillateur à quartz/diviseur de fréquence permet d'activer ou d'inhiber le comptage pendant une durée connue précisément (grâce à l'oscillateur à quartz très stable dans le temps !). Les créneaux en sortie de cet étage ont une fréquence F =1/T= 215/2n où n est le numéro de la sortie du diviseur.

- L'afficheur comporte un décodeur qui permet, à partir du résultat binaire de la sortie du compteur, de faire apparaître sur des afficheurs à LED la valeur décimale correspondante. Nous allons voir que suivant la gamme de fréquence étudiée (vis à vis de l'horloge interne), le système nous donne soit la fréquence (on parle alors de fréquencemètre), soit la période (on parle alors de période mètre)

3.3 La partie réalisation :

La réalisation de ce projet fait intervenir essentiellement 2 circuits :

1. Une carte Arduino Uno
2. Un afficheur LCD 2x16 caractères

Dans la carte arduino on exploite les circuits Timers, le timer 2 qui joue le rôle de base de temps et le timer1 qui implémente le circuit de comptage des impulsions

Le circuit LCD implémente la partie affichage de la figure 3.1

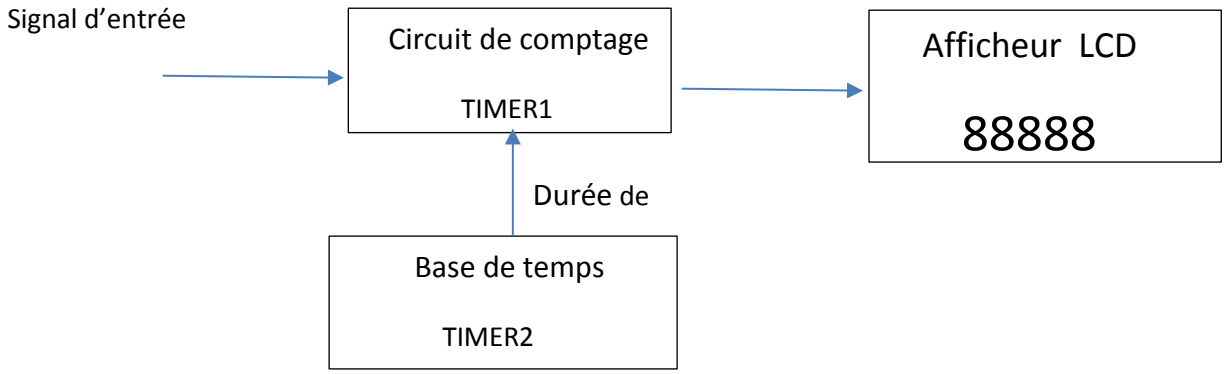


Figure 3.2. Schéma bloc du circuit de la réalisation

3.3.1 Le circuit LCD :

L'écran à cristaux liquides, aussi connu sous l'appellation ACL ou LCD, désigne le composant principal des moniteurs plats.

Sont des modules compacts intelligents et nécessitent peu de composants externes pour in

bon fonctionnement, ils consomment relativement peu (de 1 à 5 mA), il existe plusieurs des types come 2*16 / 4*16 / 4*24 (X lignes * Y caractères).

Ils utilisent le code ASCII comme un codage do fonctionnement (code 20h à 7Dh ou 7 Eh) : chiffres et lettres caractères de ponctuations, opérateurs arithmétiques, etc.

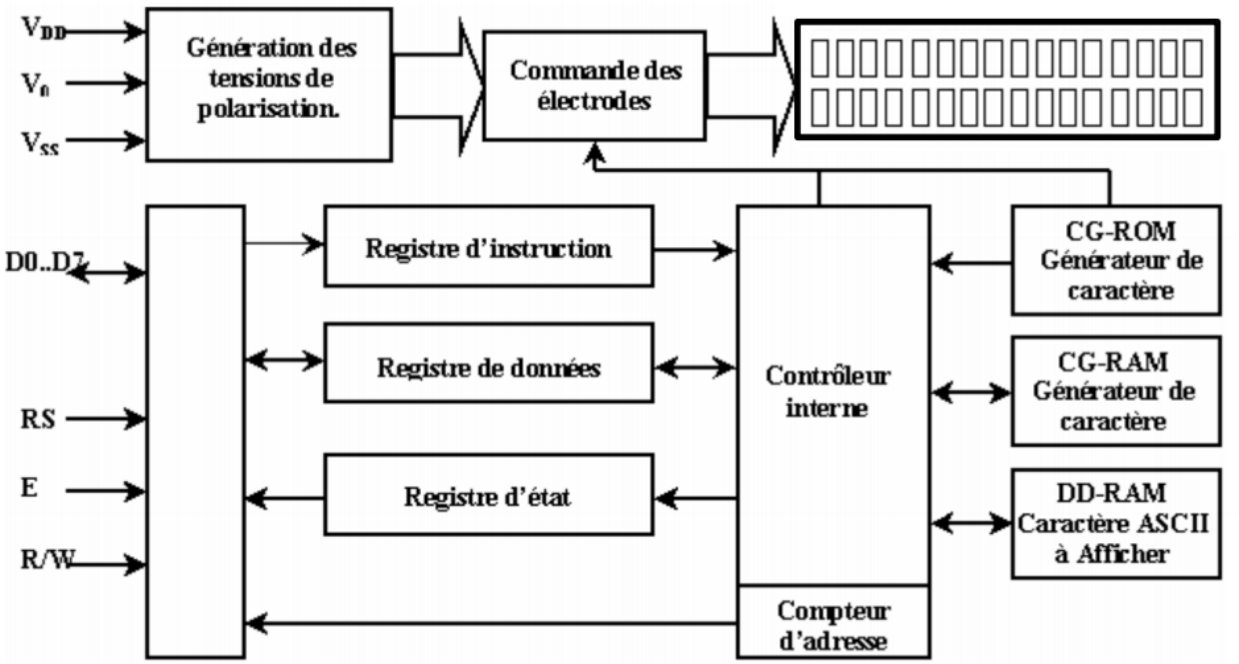


Figure 3.3. La structure interne d'un afficheur LCD

a le brochage et le pin mode

L'écran LCD peut fonctionner dans deux modes différents, à savoir le mode 4 bits et le mode 8 bits. En mode 4 bits, nous envoyons le quartet de données par quartet, premier quartet supérieur puis quartet inférieur.

Un quartet est un groupe de quatre bits, donc les quatre bits inférieurs (D0-D3) d'un octet forment le quartet inférieur tandis que les quatre bits supérieurs (D4-D7) d'un octet forment le quartet supérieur. Cela nous permet d'envoyer des données 8 bits. Alors qu'en mode 8 bits, nous pouvons envoyer les données 8 bits directement d'un seul coup, car nous utilisons toutes

les 8 lignes de données. Le mode 8 bits est plus rapide et sans défaut que le mode 4 bits. Mais l'inconvénient majeur est qu'il nécessite 8 lignes de données connectées au microcontrôleur. Cela nous fera manquer de broches E / S sur notre MCU, le mode 4 bits est donc largement utilisé. Aucune broche de contrôle n'est utilisée pour définir ces modes. C'est juste la façon de programmer ce changement.

N pin	Symbol	Fonction
1	VSS	GND
2	VDD	+5V
3	V0	Ajustement de contraste
4	RS	0 = Instruction ; 1 = Donnée
5	R/W	0 = Ecriture ; 1 = Lecture
6	E	Enable
7	DB0	Bus de donnée, bit 1
8	DB1	Bus de donnée, bit 2
9	DB2	Bus de donnée, bit 3
10	DB3	Bus de donnée, bit 4
11	DB4	Bus de donnée, bit 5
12	DB5	Bus de donnée, bit 6
13	DB6	Bus de donnée, bit 7
14	DB7	Bus de donnée, bit 8
15	A	+5Volt du rétroéclairage
16	K	Mass du rétroéclairage

Tableau 3.1. Brochage d'un afficheur LCD

L'afficheur LCD utilise 6 à 10 broches de données ((D0 à D7) ou (D4 à D7) + RS + E) et deux d'alimentations (+5V et masse). La plupart des écrans possèdent aussi une entrée analogique pour régler le contraste des caractères.

3.3.2 Le Microcontrôleur 328P

Est un composant électronique qui rassemble tous les éléments d'un "mini-ordinateur" et qui se présente sous la forme d'un circuit intégré. Permet de réaliser des systèmes et montages électroniques programmés. Cela veut dire que l'on pourra, avec le même montage, réaliser des fonctions très différentes qui dépendront du programme qui aura été programmé

dans le microprocesseur.

Nous utilisons le ARDUINO come un base circuit complémentaire pour le microcontrôleur 328P, Le microcontrôleur ATMega328P est un microcontrôleur 8bits de la famille AVR dont la programmation peut être réalisée en langage C.

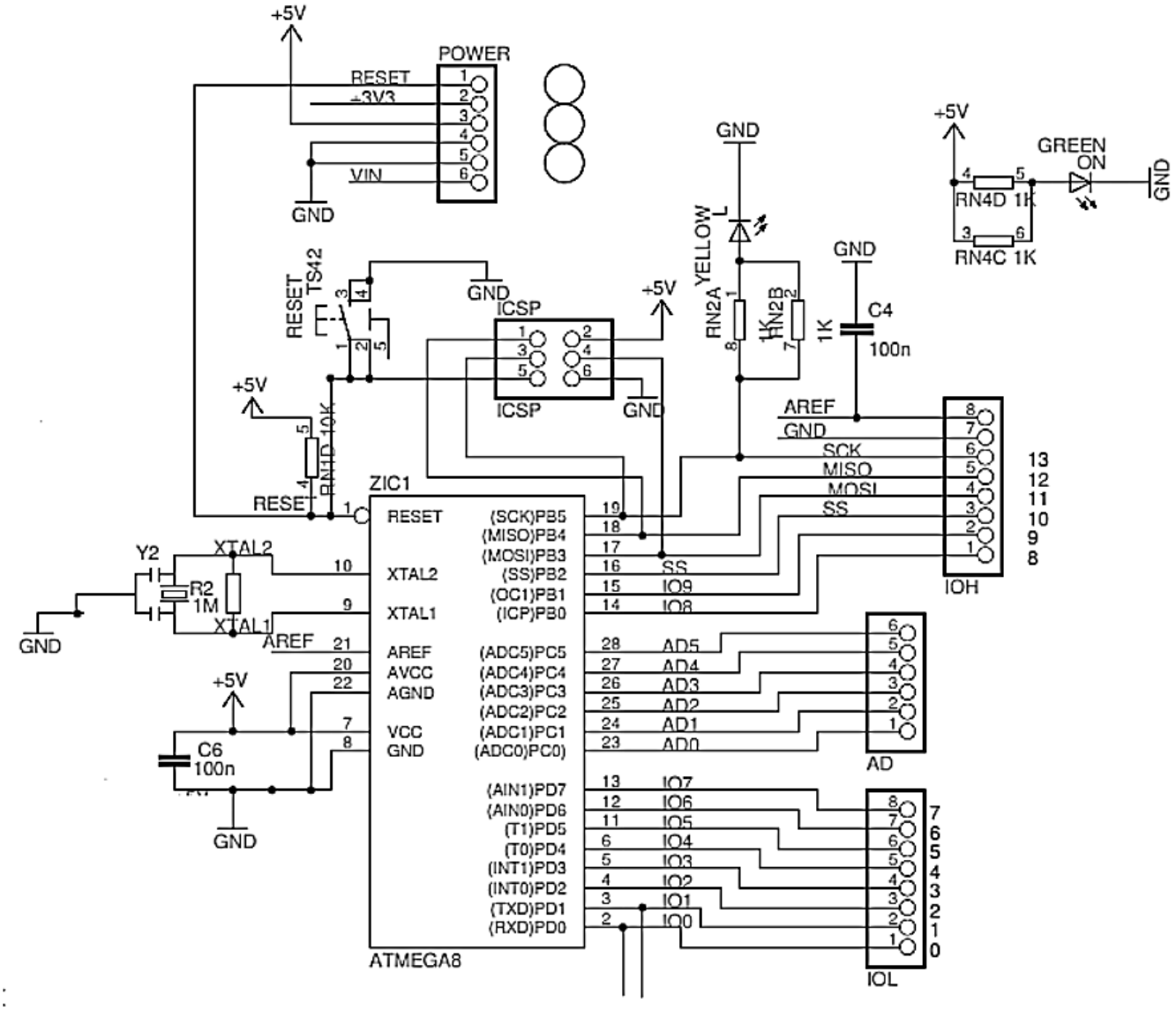


Figure 3.4. Schéma simplifié de la carte Arduino UNO

Les signaux d'entrée-sortie du microcontrôleur sont reliés à des connecteurs avec ARDUINO selon le schéma ci-dessous.

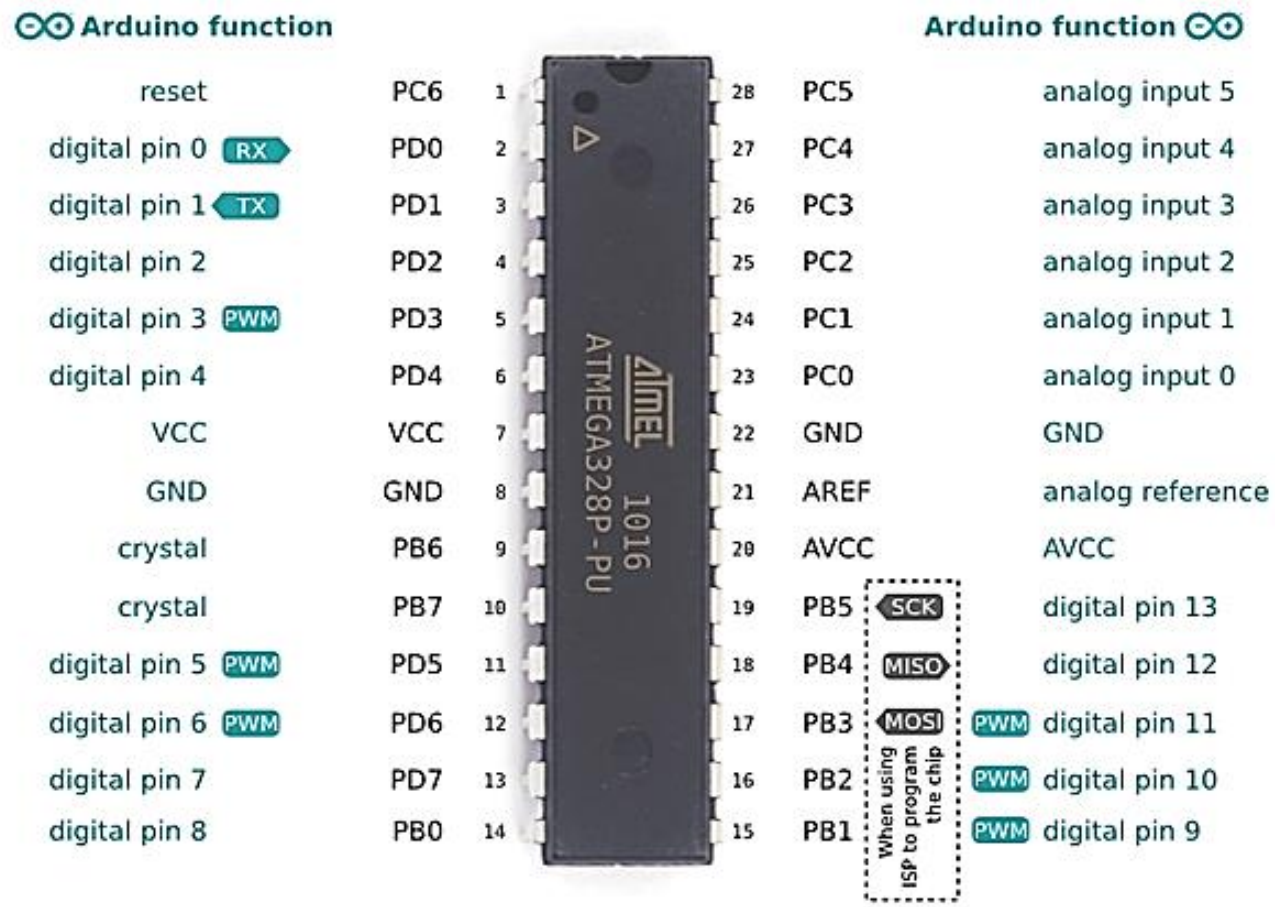


Figure 3.5. Cartographie des broches d'ATmega 328P

Le Potentiomètre

C'est un type de résistance variable à trois bornes, dont une est reliée à un curseur se déplaçant sur une piste résistante terminée par les deux autres bornes. Ce système permet de recueillir, entre la borne reliée au curseur et une des deux autres bornes, une tension qui dépend de la position du curseur et de la tension à laquelle est soumise la résistance.

Dans ce projet utiliser pour ajuster le contraste de LCD.

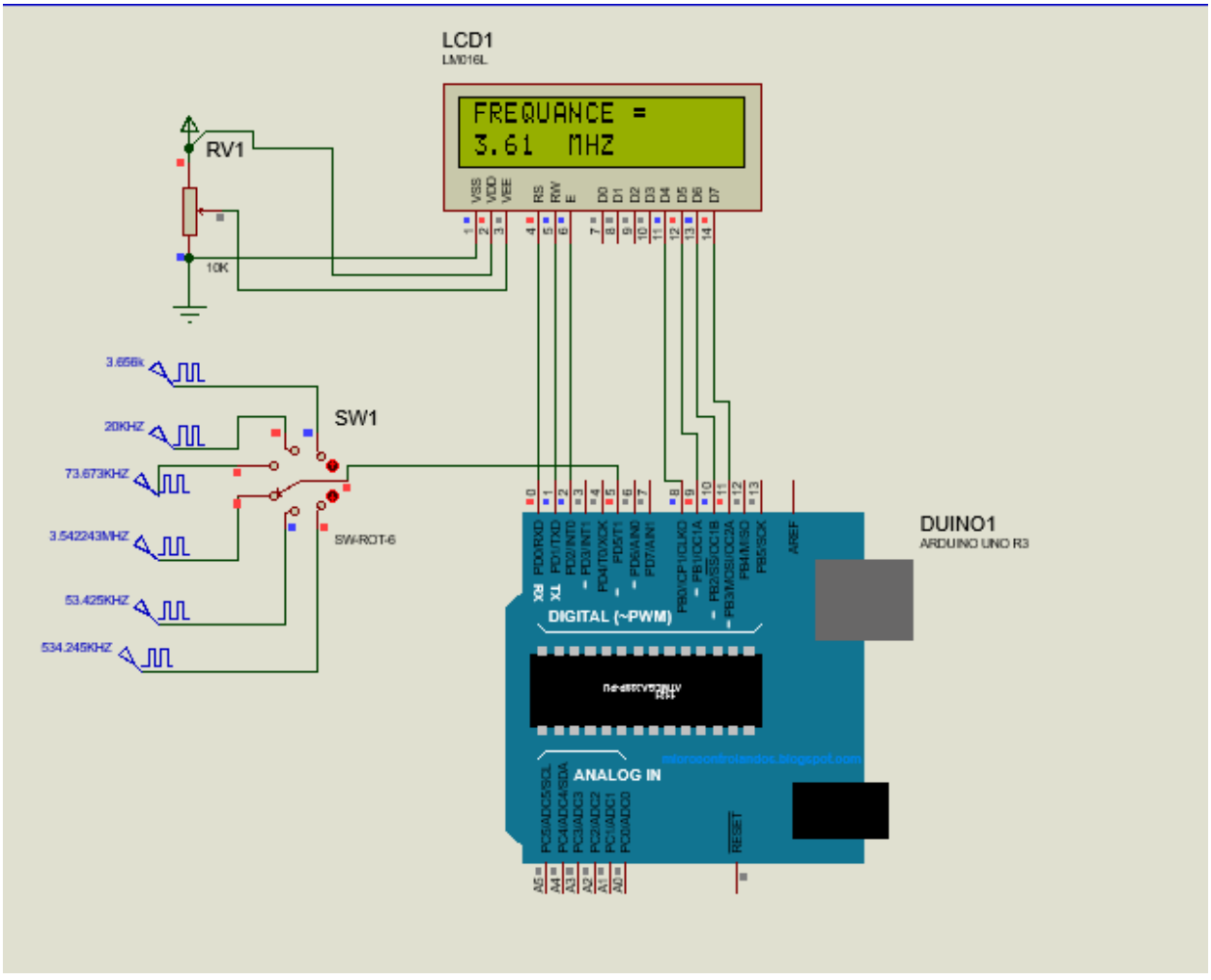


Figure 3.6. Simule le Project avec les appareils utilisés

3.4 La Partie Software

3.4.1 CodevisionAVR

CodeVisionAVR est un compilateur croisé C, un environnement de développement intégré et un générateur de programme automatique conçu pour la famille de microcontrôleurs atmel avr. il est conçu pour fonctionner sous Windows.

- Environnement de développement intégré facile à utiliser et compilateur compatible ANSI C.
- Éditeur avec indentation automatique, mise en évidence de la syntaxe pour l'assembleur C et AVR, paramètres de fonction et structure / union.
- Assistance d'interruption.
- Outre son propre IDE, CodeVisionAVR peut également être utilisé comme extension intégrée à Atmel Studio 7.

3.4.2 Les organigrammes et le programme

a L'organigramme

Est une représentation graphique normalisée de l'enchaînement des opérations et des décisions effectuées par un programme d'ordinateur.

Un organigramme sert à indiquer la répartition des responsables, d'ensembles de tâches entre les postes, et les relations de commandement qui existent entre eux.

Il ne donne que peu d'informations en ce qui concerne la répartition des tâches.

Il permet de représenter les relations de commandement (ainsi que les statuts, cadre, assimilé cadres, ...) les rapports de subordination.

Les postes de direction et d'exécution sont représentés par des rectangles et les postes d'états-majors par des ovales.

- ***Organigramme du programme général***

Premier étape déclaration des variables et les constants qui on a utilisé pour sauvegarde les nombres d'impulsion qui génère par timer 1 et le nombres d'interruptions qui génère par le timer2 (mode CTC) et pour utilise pour la partie mathématique qui calcule la valeur de fréquence finale pour affiche sur LCD .

Count =**count_hihg** + **count_low :** le variable count représente le nombre d'impulsion qui généra par le timer 1.

Count_high : c'est la partie de poids fort de conteur (16 bits poids fort).

Count_low : c'est la partie de poids faible de conteur (16 bits poids faible).

Et on a utilisé conteur de 32 BIT pour augmente l'avaleur de fréquences que on va mesurer.

Avec ce comptage sur 32 bits, on a intérêt à augmenter la durée de comptage pour faire des mesures en basse fréquence. On arrête le comptage lorsque le nombre d'interruptions atteint une valeur **ninter**, ce qui fait une durée de comptage **ΔT**=**ninter***Ti. Un durée de 1 seconde (environ) est obtenue avec **ninter=60**. Avec

Freq : représente la valeur mesure de fréquence

On initialise le LCD premièrement pour bien performance d'affichage.

En suit on configure les pins les sorties et les entries (l'entrée de signal qui va calcule leur fréquence)

L'étape suivant on mette le variable count a zéro pour confirme que le calcule d'impulsions de timer 1 se démarre a 0 pour bien précision de mesure.

S.Programme start_count si la plus un portant partie de fréquencemètre dans sa partie On active les timers 1 et 2 et nous avons faire une activation des interruptions de chaque timer En suite sa et d'après l'information qui on a obtenu (les nombres d'impulsion et interruption) On calcule la valeur de fréquence et pour affichera sur LCD .

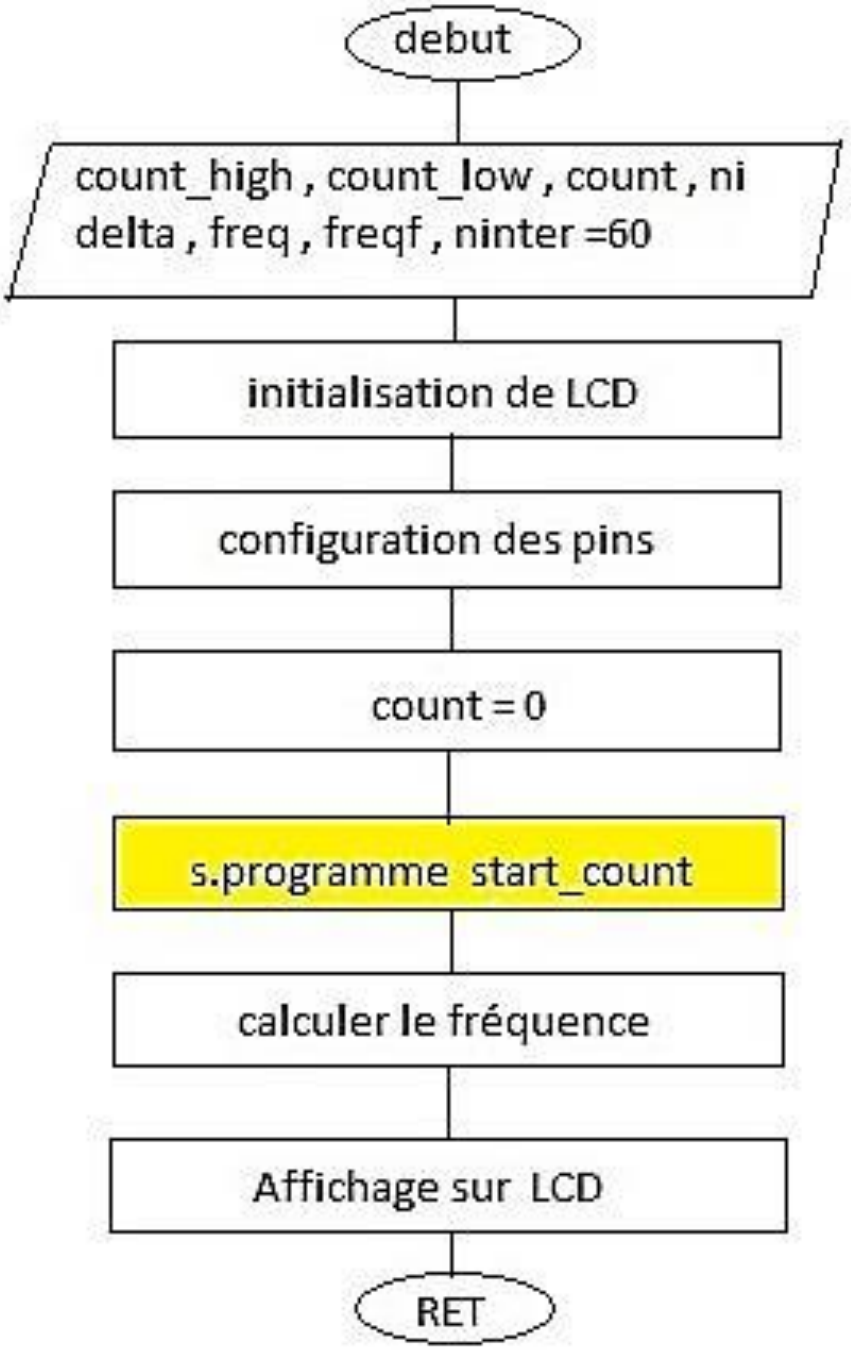


Figure 3.7. Organigramme du programme général

b Le programme

la programmation d'un fréquencemètre avec ATmega 328P, capable de mesurer des fréquences allant de quelques Hz jusqu'à *4 MHz*. Le max chiffre qui être un peu des errer.

La mesure de fréquence se fait en comptant les fronts montants sur l'entrée pendant une durée déterminée, par exemple 1 seconde. La manière la plus précise de faire ce comptage est d'utiliser un Timer 16 bits en mode *capture*.

ATmega328P L'entrée utilisée est T1, qui possède le timer à 16 bits reliée à la borne D5 de la carte. Il faut aussi programmer des interruptions périodiques pour fixer précisément la durée du comptage. On utilise pour cela le Timer2 à 8 bits. Le Timer2 est configuré en mode CTC, qui consiste à remettre à zéro le registre du compteur 8 bits (TCNT2) lorsqu'il atteint la valeur du registre OCR2A. On choisit OCR2A=0xFF, ce qui signifie que la période d'interruption est égale à 256 tops d'horloge. L'horloge du compteur est l'horloge de la carte (16 MHz), à laquelle une division est appliquée (*prescaler*). On choisit le facteur de division le plus élevé (1024), ce qui fait une fréquence d'horloge *16/1024 MHz*.

La période d'interruption est alors :

$T_i = (1024*256)/16 = 16384\ \mu s.$

En déclenchant une interruption lorsque le compteur 16 bits revient à zéro. On définit donc deux variables 16 bits pour le comptage count_high (16 bits de poids fort) et count_low (16 bits de poids faible).

On arrête le comptage lorsque le nombre d'interruptions atteint une valeur nombre interruption, ce qui fait une durée de comptage $\Delta T = Niterr * T_i$. Un durée de 1 seconde est obtenue avec *Niterr=60*.

NOTE : le programme dans l'annexes

Dans S.programme start_count on va configure et déclenche le Timer1 (pour le comptage en mode capture) et le Timer2 pour les interruptions.

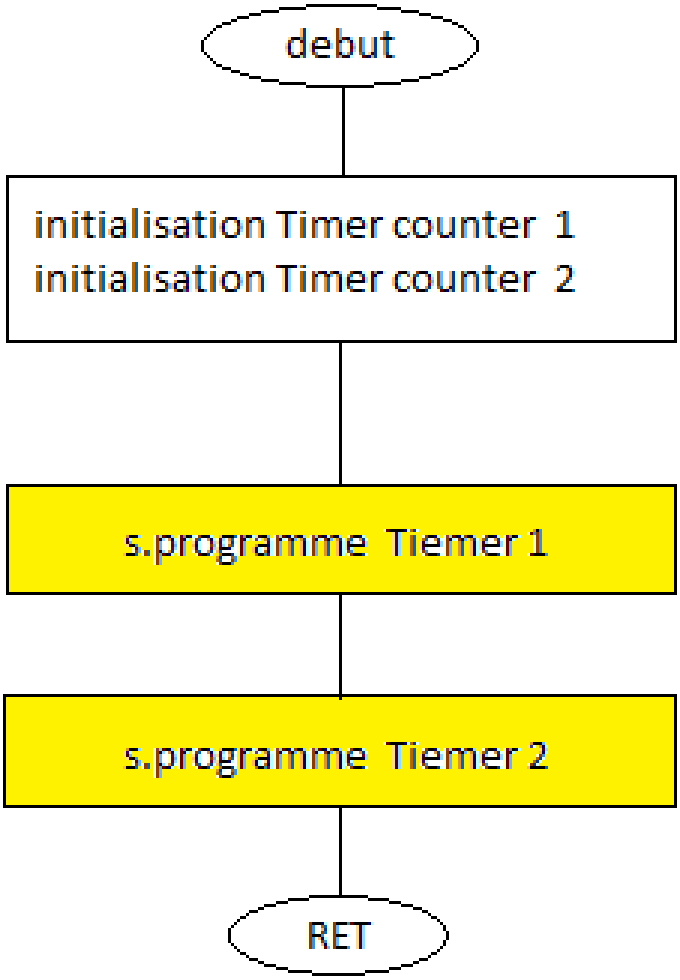


Figure 3.8. Organigramme de Start-Count sous-programme

La fonction d'interruption pour le timer 1 déclenchée lors du débordement du compteur d'impulsions doit incrémenter le 16 bits de poids fort

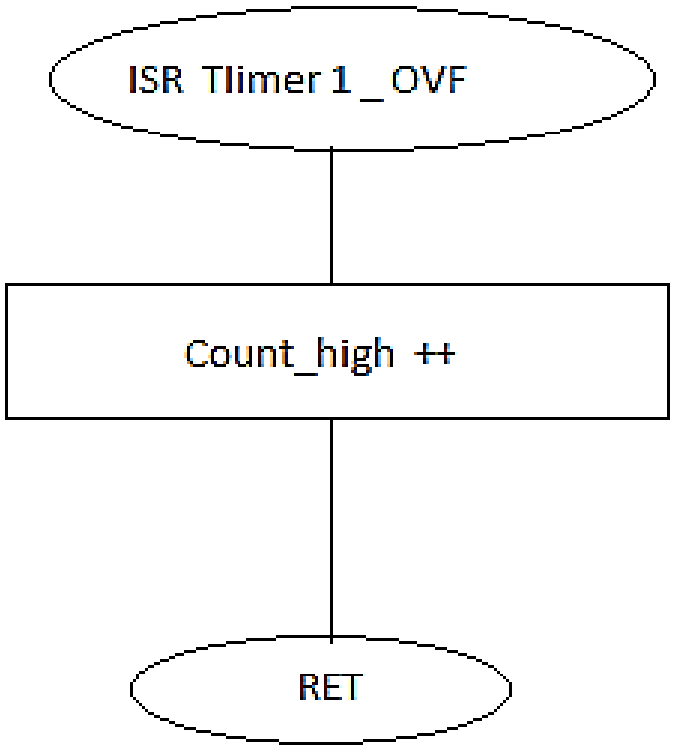


Figure 3.9. Organigramme du sous-programme Timer1

Le Timer2 est configuré en mode CTC (clear timer on compare) la fonction Timer 2 appelée lors de l'interruption déclenchée par le débordement du Timer 2. Lorsque le nombre d'interruptions atteint 60 interruptions on enregistre le nombre d'impulsions comptées dans count puis on remet le compteur d'impulsions à zéro. Se qui permet Aussi determiner une seconde de temps

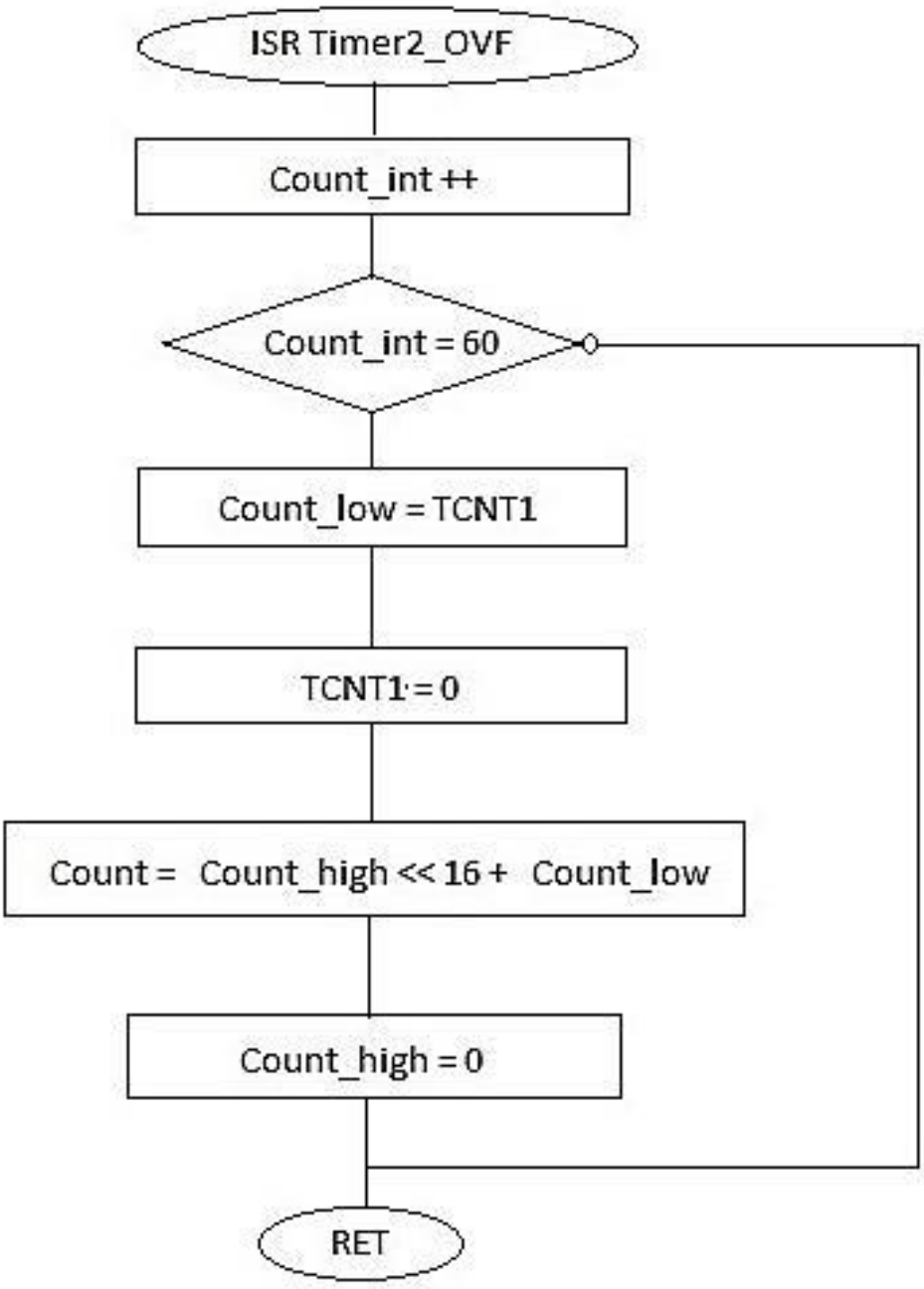


Figure 3.10. Organigramme du sous-programme Timer2

3.4.3 Implémentation

Pour implémenter le programme dans le microcontrôleur nous doit faire le suivant :

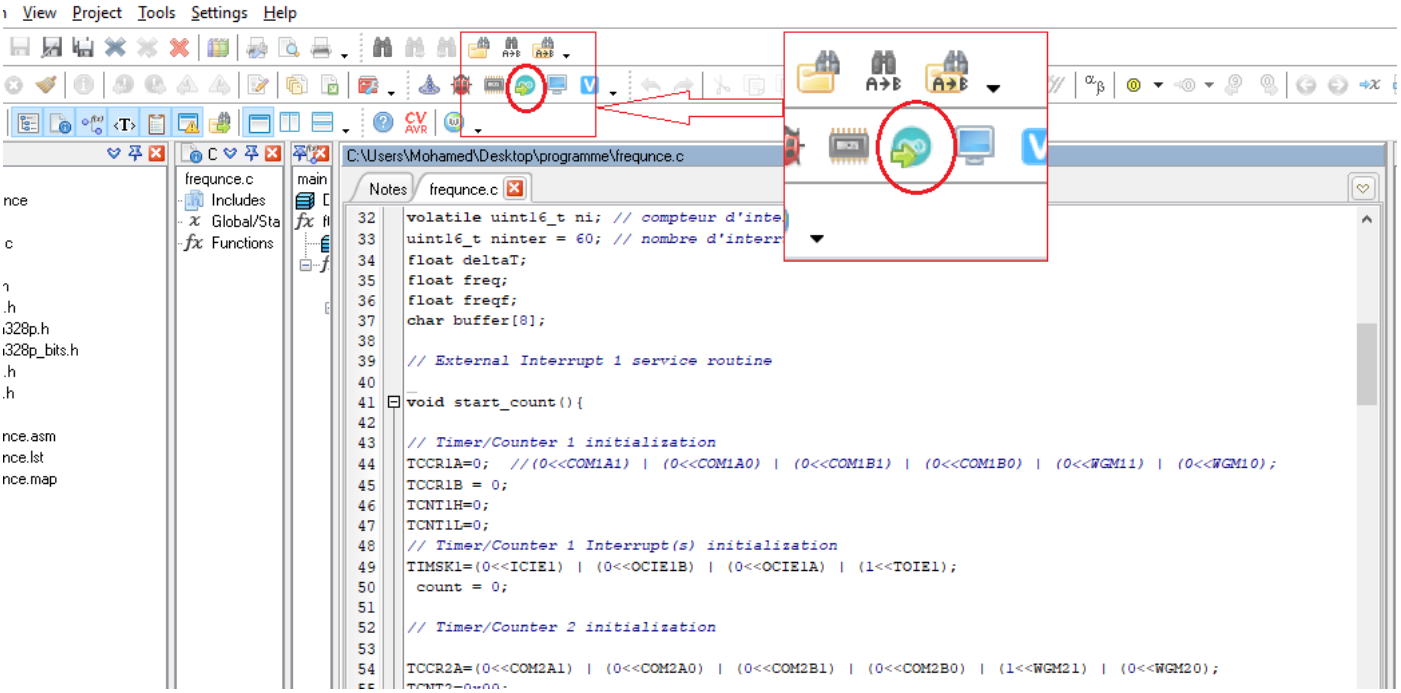


Figure 3.11. 1er étape de l'implémentation

Nous devons cliquer dans le symbole de l'arduino dans l'interface principale, puis il va apparaître une table que nous choisirons arduino uno atmega328P.

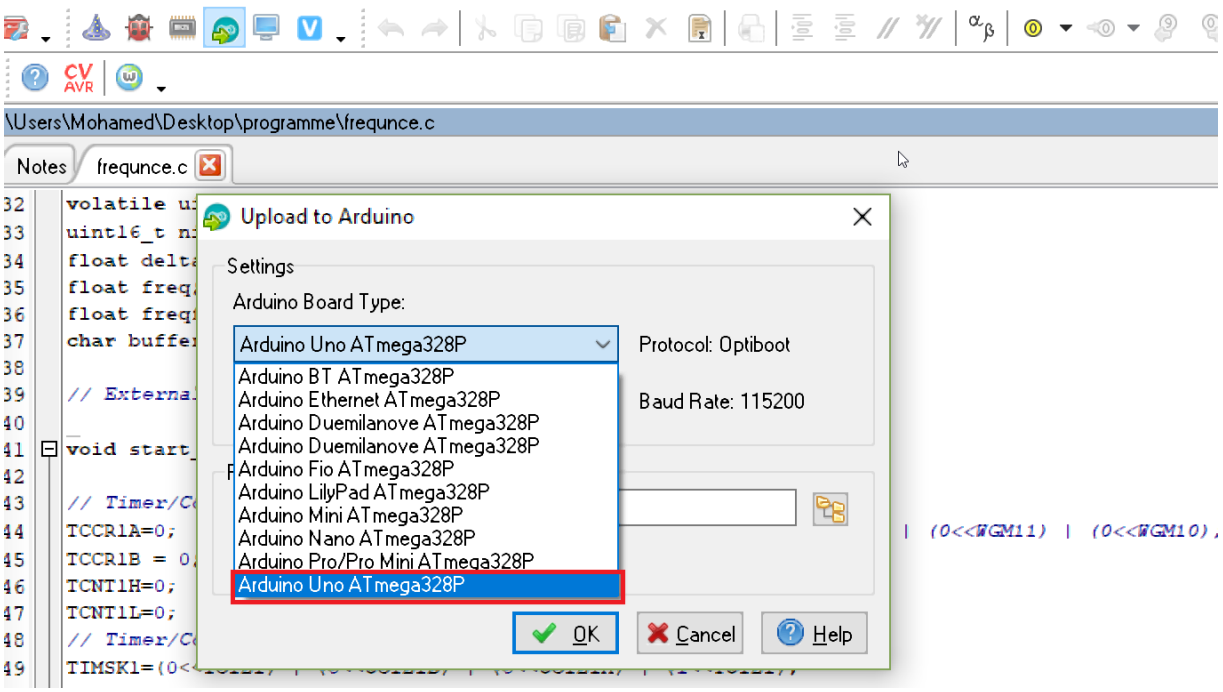


Figure 3.12. 2eme étape de l'implémentation

Après d'appuyé sur OK, le chargement commence.

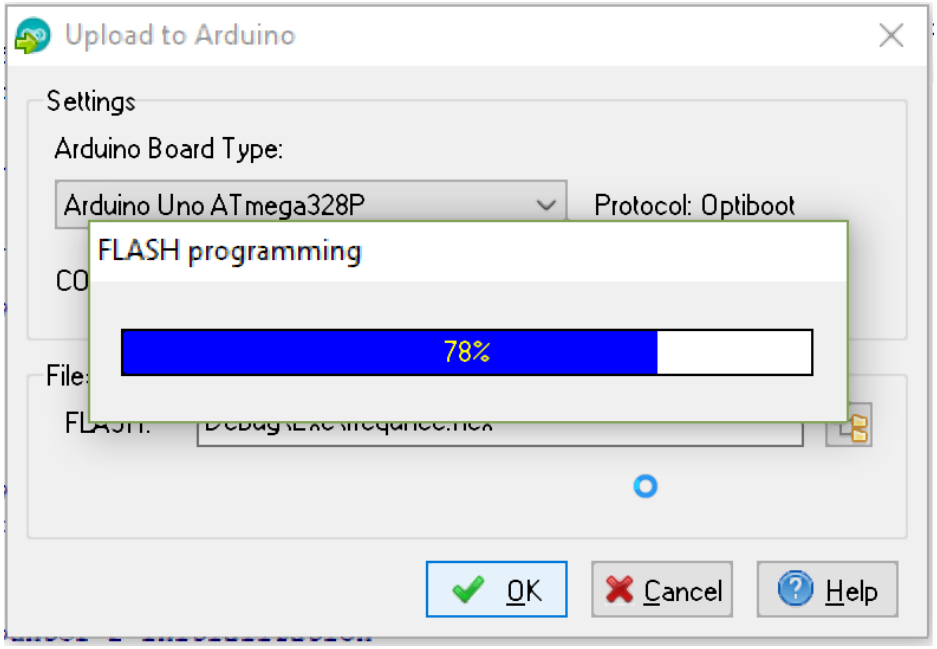


Figure 3.13.3eme étape de l'implémentation

3.5 Les Résultats obtenus :

Fréquence	fréquence a mesure	Erreur relative %
20 HZ	20.190429 HZ	0.00952145
100 HZ	100.553390 HZ	0.0055339
1 KHZ	1.00606286	0.00606286
10.002 KHZ	10.00467967 KHZ	0.000267913
100 KHZ	100.0203396 KHZ	0.000203396
500 KHZ	500.15057373 KHZ	0.000301147
1 MHZ	1.0002319812 MHZ	0.000231981
2.001 MHZ	2.0018727779 MHZ	0.0004361170

Tableau 3.2. Résultat des essais pour 1 seconde

En règle général, on constate que la précision de la mesure augmente avec la fréquence.

L'erreur pour un signal de fréquence 20Hz elle est de l'ordre de 1%, celle-ci descend à 0.02679% pour un signal de fréquence 10Khz.

Ce raisonnement s'avère être logique, puisque plus le nombre d'impulsions à mesurer augmente l'erreur de mesure diminue, car celle-ci est le rapport entre les impulsions loupées et le nombre total des impulsions pendant la fenêtre de base de temps.

Un autre facteur joue aussi un rôle important dans la précision de la mesure, c'est l'exactitude de la durée de la base de temps. Celle-ci est en relation avec la précision de la fréquence de l'oscillateur a quartz du microcontrôleur ainsi que du sous-programme réalisant la mesure de la durée de la base de temps (Timer2).

Fréquence	fréquence a mesure	Erreur relative %
100 HZ	101.570648 HZ	0.01570648
1.001 KHZ	1.00402832 KHZ	0.003025294
100.01 KHZ	100.03662872 KHZ	0.0002662605
2.001 MHZ	2.0019195079 MHZ	0.0004595241

Tableau 3.3. Résultat des essais pour 1/3 seconde

Le tableau 3.3 donne les mesure pour une base de temps de 1/3 de seconde, on constate une détérioration des résultats pour les basses fréquences 100hz et 1Khz.

Et des résultats à peu près identique à la base de temps 1seconde pour les fréquences plus élevées 100Khz et 2 Mhz.

C'est tout à fait normal, si la durée de la base de temps diminue cela augmente l'erreur de la mesure pour les basses fréquences

Conclusion générale

La mesure est un processus de connaissance qui grâce à l'expérience physique nous donne une information quantitative (valeur) du rapport entre la grandeur mesurable et une grandeur de même nature prise comme unité.

Notre fréquencemètre de dimension réduite, simple à réaliser, basé sur un microcontrôleur capable de mesurer la fréquence d'un signal rectangulaire au format TTL comprise entre 20 hertz et 20 Mhz, avec une base de temps à 1 seconde.

Cependant ce fréquencemètre tel qu'il a été implémenté présente un certain nombre de limites :

1) La forme du signal rectangulaire, si on veut englober d'autres formes de signaux il faudra ajouter un circuit de mise en forme à l'entrée du fréquencemètre.
2) La précision de la mesure est loin d'être parfaite, pour l'améliorer il faudra adopter une base de temps à durée variable 10 ms 50ms 100ms 500ms et 1000ms.

Plus on va réduire la durée de la base de temps (temps de comptage), plus la précision de la mesure sera réduite. On a une précision identique quand on dit que l'on peut mesurer une fréquence de 1Khz à 0.1 Hz près, ou une fréquence de 10Khz à 1 Hz près, dans les deux cas on a une précision de 0.01 %.

Selon les résultats obtenus à la fin du chapitre III, on constate que la précision de la mesure de fréquence d'un signal de 20Hz avec une base de temps de 1s n'est absolument sans aucun rapport avec celle de la mesure d'un signal de 10Khz avec la même base de temps.

A partir de là on peut affirmer que pour améliorer la précision de la mesure, il faudra adapter la durée de la base de temps à la fréquence du signal à mesurer.

Remarque : Pour une base de temps fixe par exemple 1000ms la précision augmente avec la fréquence du signal à mesurer. Donc elle diminue sur les basses fréquences, à ce moment-là il faudra augmenter la base de temps.

Le programme pour ATmega 328P

Programme globale de fréquence mètre

```
/*****************************************************
This program was created by the
CodeWizardAVR V3.12 Advanced
Automatic Program Generator
Project : Fréqunce mètre
Date    : 6/3/2018
Chip type           : ATmega328P
Program type        : Application
AVR Core Clock frequency: 16.000000 MHz
Memory model        : Small
External RAM size   : 0
*****************************************************/
*****************************************************/
```

#include Dites au compilateur où rechercher d'autres bits de code que vous utilisez qui ne résident pas dans ce fichier. Ce sont normalement des fichiers ".h" ou des fichiers d'en-tête contenant des macros et des prototypes de fonctions. Avant de compiler un fichier .c, le contenu des #includes est littéralement copié dans le fichier.

```
*****************************************************/
#include <mega328p.h>
#include <alcd.h>
#include <delay.h>
#include <stdlib.h>
#include <stdint.h>
```

**********/

Les prototypes de fonctions donnent le nom, les arguments et le type de retour des fonctions qui seront utilisées ultérieurement dans le programme. Ils peuvent être inclus dans les fichiers d'en-tête ou dans le fichier .c avant que la fonction ne soit définie.

**********/

```
// Declare your global variables here
volatile uint16_t count_high,count_low;
volatile uint32_t count;
volatile uint16_t ni; // compteur d'interruptions
uint16_t ninter = 60; // nombre d'interruptions pour un comptage
float deltaT;
float freq;
float freqf;
char buffer[8];
// External Interrupt 1 service routine
void start_count(){
// Timer/Counter 1 initialization
TCCR1A=0;  //(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B = 0;
TCNT1H=0;
TCNT1L=0;
// Timer/Counter 1 Interrupt(s) initialization
TIMSK1=(0<<ICIE1) | (0<<OCIE1B) | (0<<OCIE1A) | (1<<TOIE1);
```

```
 count = 0;
// Timer/Counter 2 initialization
TCCR2A=(0<<COM2A1) | (0<<COM2A0) | (0<<COM2B1) | (0<<COM2B0) | (1<<WGM21) | (0<<WGM20);
TCNT2=0x00;
OCR2A=0xFF;
// Timer/Counter 2 Interrupt(s) initialization
TIMSK2=(0<<OCIE2B) | (0<<OCIE2A) | (1<<TOIE2);
ni = 0;
  #asm("sei")// activation des interruptions
TCCR2B |= (1 << CS12) | (1 << CS11) | (1 << CS10); // prescaler = 1024
TCCR1B |= (1 << CS12) | (1 << CS11) | (1 << CS10); // external clock on rising edge
    }
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{ count_high++; }
// Timer2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
ni++;
   if (ni==ninter) {
     ni = 0;
     count_low =((uint8_t) TCNT1H) <<8 | TCNT1L;
     TCNT1L = 0;
     TCNT1H = 0;
     count = ((uint32_t)count_high)<<16 | count_low;
    count_high = 0;
```

```
  }}
**************************************************/
```

La fonction principale est littéralement la partie principale du code. Il ne peut y avoir qu'une fonction principale dans le programme compilé final. Le code dans la fonction principale est exécuté séquentiellement, une ligne à la fois.

```
**************************************************/
void main(void)
{
     lcd_init(16);
   lcd_clear();
  while(1){
   DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1) | (1<<DDB0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);
count=0;
start_count();
delay_ms(1000);
deltaT = 1.0*ninter*1024*256/16000000;
freq = count/deltaT;
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("FREQUANCE =");
lcd_gotoxy(0,1);
if(freq>=1000000)          //si fréquence supérieure ou égale à 1Mhz utilisez "Mhz" sur lcd
```

```
{
freqf=(float)freq/1000000;   //divise par 1Mhz
ftoa(freqf,3, buffer);
lcd_puts(buffer);
lcd_putsf(" MHZ");
}
else if (freq>=1000)       //si fréquence supérieure ou égale à 1khz utilisez "Khz" sur lcd
{
freqf=(float)freq/1000;
ftoa(freqf,3, buffer);
lcd_puts(buffer);
lcd_putsf(" KHZ");
}
else                    // si fréquence inférieure à 1khz utilisez "hz" sur lcd
{
ltoa(freq, buffer);
lcd_puts(buffer);
lcd_putsf(" HZ");
}}}
```

Bibliographie

[1] Richard H.Barnett, Sarah Cox ET Larry O'Cull : 'Embedded C programming and the atmel AVR', Delmar Cengage Learning, 2007.

[2] John Morton : 'AVR An Introductory Course', Newnes, 2002.

[3] Juillot Guillaume : 'La programmation des ATMEL AVR', 2003.

[4] ATmel corporation : 'AVR Instruction Set Manual',2016.

[5] HP info Tech : ' CodeVisionAVR V1.23.9b user manual', 2003

[6] Carl Ebeling :'Lecture 6- ATmega 328 Timers and inturrpts',2010

[7] Jean Noël :' Microcontrôleur Atmel ATmega',2005.

[8] http://maxembedded.com/2011/06/avr-timers-timers0/

[9] www.avrbeginners.net

[10] http://www.ladyada.net/learn/avr/index.html

[11] http://web.ics.purdue.edu/~jricha14/

[12] https://fr.wikiversity.org/wiki/Micro_contr%C3%B4leurs_AVR/Le_Timer_2