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

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière d'Automatique

جامعة محمد خيضر بسكرة
كلية العلوم و التكنولوجيا
قسم:الهندسة الكهربائية
شعبة: الآليـــــة

Support de cours du module :

Micro-contrôleurs

Préparer par : Dr. Hassina MEGHERBI

Pour les étudiants de1er année Master,

Option : - Automatique avancée;

- Génie des systèmes industriels.

Année universitaire : 2014-2015

Table des matières

Chapitre 1 : Généralités sur les Mémoires

1 Terminologie et Fonctionnement Générale des Mémoires:

1.1 Définition :

Une mémoire est un dispositif (circuit intégré, support magnétique, etc.) qui emmagasine les informations et les restitue à la demande.

Les informations peuvent être des instructions d'un programme, des données associées ou des résultats intermédiaires. Chaque mot est identifiée par une adresse.

On ce limite dans ce cours par l'étude des mémoires à semi-conducteurs.

1.2 Structure générale d'une mémoire :

Un bloc mémoire peut être représenté comme sur la figure 1. Pour pouvoir identifier individuellement chaque mot (donnée) on utilise n lignes d'adresse. Une ligne de commande (R/W) indique si la mémoire est accédée en écriture (l'information doit être mémorisée) ou en lecture (l'information doit être restituée). La ligne de validation ou de sélection du bloc (CS) permet de valider le bloc mémoire (l'information mémorisée se retrouve sur la ligne de sortie et permet de inhiber ce

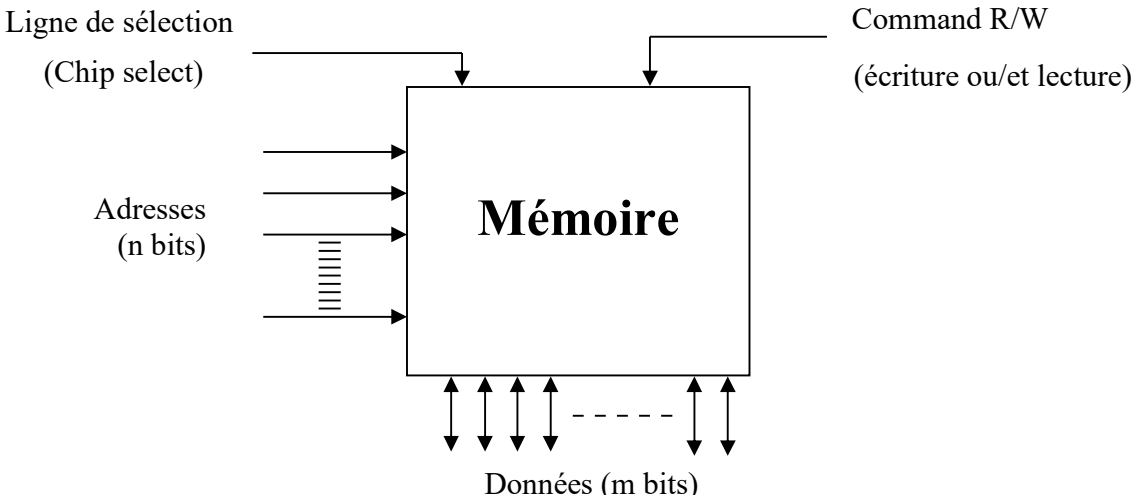


Figure 1**: Structure générale d'une mémoire**

bloc (les lignes des données sont isolées). Les lignes de données véhiculent aussi bien les données à stocker que les données lus.

1.3 Caractéristiques :

Format : représente le nombre de bits de la donnée (m).

Nombre de cases mémoires : Nombre de données pouvant être stockées (N).

$N=2^n$ (où n est le nombre de bits d 'adresse)

Capacité : représente le nombre total de bits peuvent être stockés C=N x m.

Exemple : Calcul de la capacité d'une mémoire.

Soit une mémoire de 8 bits de données et de 16 bits d'adresse.

Format : m=8; **Nombre de cases mémoires :** $N = 2^{16} = 65536$

Capacité : $C = m \times N = 8 \times 65536 = 524288$ bits

Pour des raisons de simplification, on exprime la capacité :

en kilo-octets (Ko) 1 octet♠ = 8 bits

1 Ko = 2^{10} = 1024 octets

en méga-octets (Mo) 1 Mo = 2^{20} octets = 1048576 octets

en giga-octets (Go) 1 Go = 2^{30} octets = 1073741824 octets

en tera-octets (To) 1 To = 2^{40} octets = 1099511627776 octets

Pour l'exemple précédent C = 524288 bits

C = 524288/8 = 65536octets = 65536/1024 = 64 Ko

Temps d'accès : temps d'obtention d'une information contenue dans la mémoire

1.4 Organisation interne de la mémoire:

Une mémoire est constituée d'un ensemble de cellules organisées sous une forme matricielle, figure 2. Chacune de ces cellules peut stockée un seul bit et chaque ligne de la matrice correspond à un mot (une donnée) de m bits.

♠ octet (francais) = byte (englais)

Attention : bit ≠ byte (=8bits)

2 Les Différentes Types de Mémoires

On distingue deux grandes catégories de mémoires: les mémoires volatiles et les mémoires non-volatiles.

2.1 Les Mémoires Volatiles :

Dès que l'alimentation électrique est coupée, on perd le contenu de ce genre de mémoire. Il s'agit donc d'une mémoire temporaire. En revanche, tant que l'alimentation électrique est maintenue, on peut accéder à leur contenu, le lire ou le modifier à volonté. Dans cette catégorie on trouve les mémoires dites "**vives**" ou **RAM** (**R**andom **A**ccess **M**emory, mémoire à accès aléatoire).

Le processus de mémorisation s'effectue:

- soit à l'aide d'une bascule de type D. Il s'agit dans ce cas d'une **mémoire SRAM, ou RAM statique**. Les figures 5 et 6 donnent deux structures de base différentes pour une cellule mémoire de type SRAM.

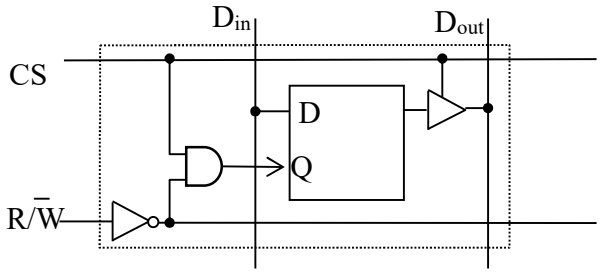


Figure 5: **Structure de base d'une cellule mémoire SRAM avec entrées/sorties séparées**

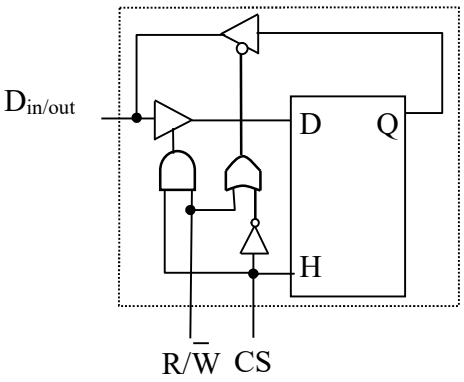


Figure 6: **Structure de base d'une cellule mémoire SRAM avec entrées/sorties non séparées**

- soit à l'aide d'un micro-condensateur (condensateur grille-substrat d'un transistor MOS). La mémoire est dite **DRAM, ou RAM dynamique**.

Le transistor MOS fondamental comporte une série de condensateurs parasites figure 7.a, qui se réduise à la plus importante, figure 7.b, celle du grille-substrat qu'on l'appelle la condensateur de structure.

La DRAM possède une plus grande densité d'intégration, car un cellule mémoire nécessite environ quatre fois moins de transistors que dans une mémoire SRAM. Sa consommation s'en retrouve donc aussi très réduite. En contrepartie, la présence de courants de fuite dans le condensateur à travers l'impédance d'entrée contribue à sa décharge. Ainsi, l'information est perdue si on ne la rafraîchie pas périodiquement. Le rafraîchissement consiste en la lecture de l'information puis le rechargement du condensateur de structure. Ce rafraîchissement indispensable à plusieurs conséquences:

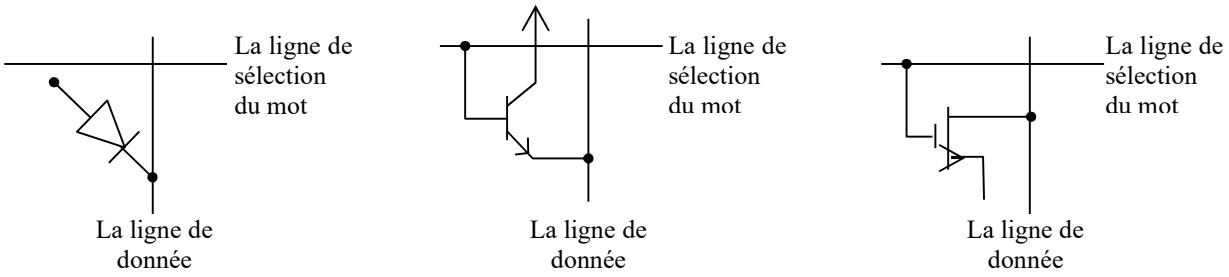
- il complique la gestion des mémoires dynamiques car il faut tenir compte des actions de rafraîchissement qui sont prioritaires.
- la durée de ces actions augmente le temps d'accès aux informations.
- D'autre part, la lecture de l'information est destructive. En effet, elle se fait par décharge de la condensateur du cellule mémoire lorsque celle-ci est chargée. Donc toute lecture doit être suivie d'une réécriture.

En conclusion, les mémoires DRAMs, qui offrent une plus grande densité d'information et un coût plus faible, sont utilisées pour la mémoire centrale, alors que les mémoires SRAMs, plus rapides, sont utilisées lorsque le facteur vitesse est critique, notamment pour des mémoires de petite taille comme les caches et les registres.

2.2 Les Mémoires Non-Volatiles :

Ce type de mémoires conservent leurs contenus lorsqu'elles ne sont plus alimentées. On distingue dans cette catégorie plusieurs types (figure 11) :

2.2.1 Mémoires Mortes (ROM : Read-Only Memory, mémoire à lecture seule)



a) **à base de diode** b) **à base de transistor** c) **à base de transistor**

Figure 9: **Cellule mémoire ROM**

2.2.1.1 Mask ROM

Mask ROM est une mémoire dont le contenu, programmé lors de sa fabrication, ne peut être ni modifié, ni effacé par l'utilisateur.

Le principe utilisé par ce type de mémoire pour stoker l'information diffère d'une technologie à une autre. Mais dans la plus part des mémoires ROM, les lignes de sélection des mots et les lignes de données forment une grille. A chaque intersection de deux lignes de la grille, l'absence ou la présence d'un élément (diode ou transistor, figure 9) différencie entre un 1 ou un 0. La figure 10 illustre une mémoire ROM à diode de capacité 4x3, son contenu des adresse 0 à 3 sont 010,111,101 et 100.

2.2.1.2 FPROM(Fuse Programmable ROM) :

Ces mémoires sont livrées non enregistrées par le fabricant. Une fois programmées, on ne peut plus modifier leur contenu.

2.2.1.3 OTP (One Time PROM) :

C'est une mémoire programmable par l'utilisateur mais une seule fois.

2.2.1.4 EPROM (Erasable PROM) :

Ces mémoires possèdent les avantages de la PROM avec un plus qui est l'effacement des données par l'utilisateur et la possibilité de la reprogrammer.

2.2.1.5 UVPROM (ou UVEPROM) :

C'est une mémoire effaçable aux UV après 10 à 20 minutes d ’exposition, ce type de mémoire est placée dans un boîtier avec fenêtre. L'effacement est total de la mémoire; et la programmation se fait par une tension de 25V.

2.2.1.6 EEPROM (Electricaly EPROM) :

Elle est effaçable et programmable électriquement. L'effacement se fait adresse par adresse. Son coût de fabrication est élevé.

2.2.1.7 FLASH EPROM:

Elle est effaçable électriquement. Et l'effacement est total de la mémoire. Elle est plus rapide (d'où le nom FLASH)et moins cher que l ’EEPROM.

2.2.2 NOVRAM :

NOVRAM est l'association dans le même boîtier :

- d’une EPROM et d’une RAM de même capacité. L'EEPROM permet de réaliser en moins de 10 ms une sauvegarde globale de la RAM. Cela permet une sauvegarde du contenu de la mémoire en cas de coupure d'alimentation électrique.
- d’une SRAM et d’une pile lithium.

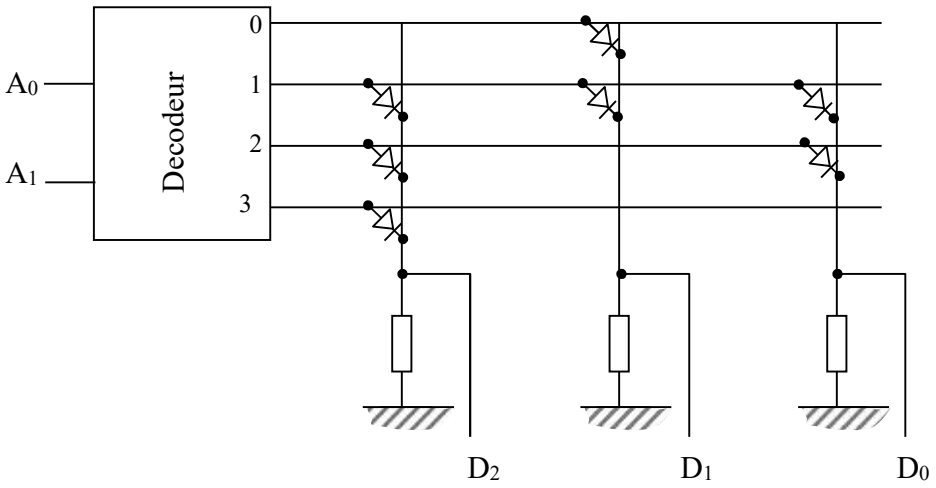


Figure 10: **Une mémoire ROM à diode**

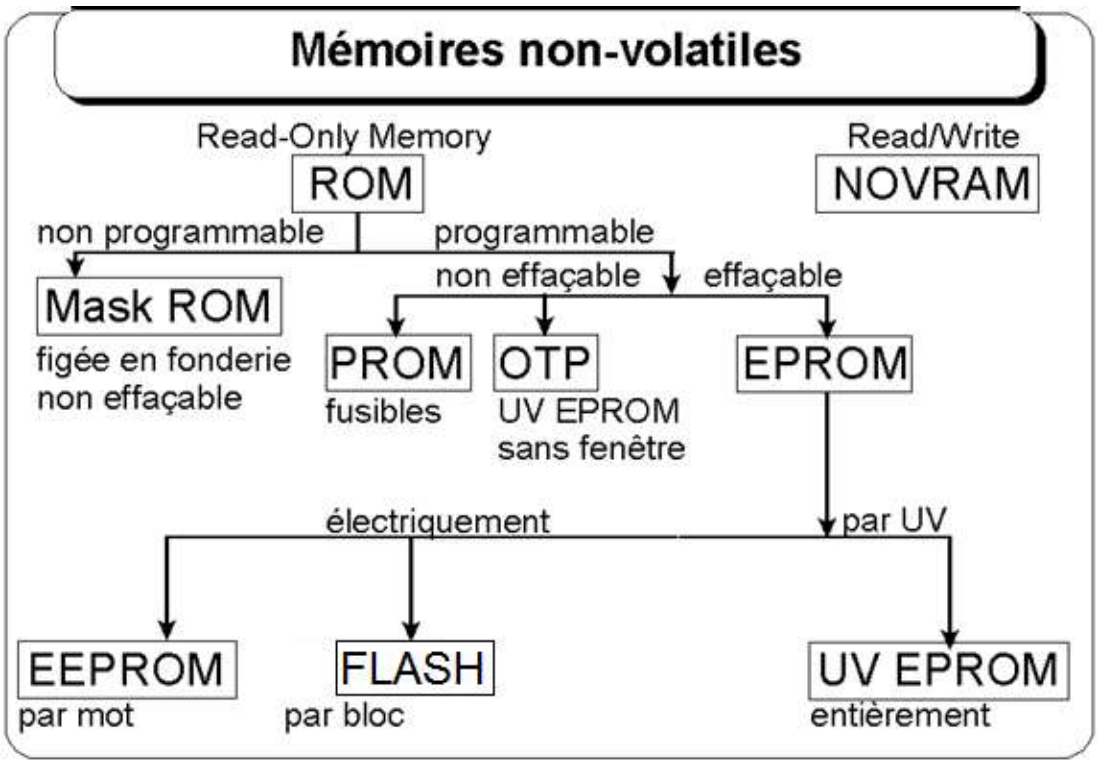


Figure 11: **Les différents types de mémoires non-volatiles**

Chapitre 2 : Présentation Générale d'un Microordinateur

1 Définition d'un micro-ordinateur

En grosso modo un micro-ordinateur est une machine ultra rapide qui traite les données, résout les problèmes et prend des décisions, tout cela sous la gouverne (ou la commande) d'un programme (qui est une séquence d'opérations programmée nécessitant le minimum d'intervention humaine). Du point de vue électronique, un micro-ordinateur est une combinaison d'éléments et de circuits numériques qui se concertent pour exécuter un programme. Ce dernier est un ensemble d'instructions codées et conservées dans la mémoire avec les données associées (les données sur lesquelles ce programme travaille).

2 Architecture d'un micro-ordinateur

Tout les micro-ordinateurs contiennent au moins les éléments de base suivants:

Le micro-processeur (μ-p) encore dénommée l'unité centrale de traitement (CPU: Central processing Unit); la mémoire; l'horloge; l'unité d'entrée/sortie.

La figure 1 illustre ces éléments et les interconnexions de base entre eux. Les interconnexions se classifient selon se qu'elles transportent (données, adresses ou commandes) et elles sont appelées **bus.**

2.1 Le MicroProcesseur

Le microprocesseur est un circuit intégré (puce ou chip) chargé d'interpréter et d'exécuter les instructions d'un programme, de lire les données ou de sauvegarder les résultats dans la mémoire et de communiquer avec les unités d'entrées/sorties.

2.2 L'horloge

Une horloge est un système logique, piloté par un oscillateur, qui émet périodiquement une série d'impulsions calibrées permettant la synchronisation du

travail de tous les éléments du microprocesseur. Ces signaux périodiques définissent le **cycle de base** ou **cycle machine**.

2.3 La mémoire

La mémoire a pour rôle de stoker et restituer les instructions codées qui forment le programme à exécuter et les données associées. Elle peut être utilisée pour stoker temporairement les résultats intermédiaires des opérations arithmétiques ou logiques.

2.4 L'unité d'entrée/de sortie

1) **Périphérique d'entrée** est constituée de tous dispositifs (organes) servant à introduire des informations et des données dans la mémoires ou dans le CPU. On peut mentionner comme exemples : le clavier, souris, mémoires de masse et.
2) **Périphérique de sortie** regroupe les dispositifs qui traduisent ou convertissent les données stockées dans la mémoire ou les résultats calculés par le CPU en une forme utilisable à l'extérieur. Parmi ces dispositifs de sorties, citons : Les écrans de visualisation, les imprimantes, les unités de disques et de bande et les convertisseurs numériques-analogiques.
3) **Interface d'entrée/de sortie** : Comme les périphériques et les composants du microordinateur sont des systèmes différents physiquement et électriquement, un circuit d'interface s'avère nécessaire pour assurer la compatibilité entre eux.

2.5 Le bus d'adresse

C'est un bus unidirectionnel qui véhicule les adresses qui correspondent soit à des emplacements en mémoire ou à un organe d'entrée/sortie. Si le bus contient 16 lignes le microprocesseur (µ-p) peut adresser jusqu'à 2^{16}=65536 adresses différentes.

2.6 Le bus de donnée

Le bus de donnée est un bus bidirectionnel sur lequel transitent, dans les deux sens, les données entre la mémoire, le µ-p et les unités entrée/sortie. La largeur du bus de donnée est la longueur du mot véhiculé (qui peut être une donnée ou une instruction). Elle est généralement utilisé pour caractériser un micro-ordinateur.

2.7 Le bus de commande

C'est un groupe de lignes véhiculant soit des signaux servant à la synchronisation des divers actions menées par les éléments du micro-ordinateur ; soit des signaux de commande.

3 Mots machines et leurs formats :

3.1 Définition

Un mot machine est un groupe de bits traitées et manipulées par un µ-ordinateur de largeur égale au largeur de bus de donnée. Il peut être une instruction d'un programme ou une donnée associée.

Un mot machine = Mot de donnée ou mot d'instruction

3.2 Format d'un mot de donnée

Les données se représentent sous diverses formes : binaires signées, non-signées, DBC, à virgule flottante, code ASCII (pour les caractères alphabétiques). Mais le µ-ordinateur ne sait pas faire la différence entre eux. C'est le programmeur qui doit assurer la bonne traduction à travers son programme.

3.3 Format d'un mot d'instruction :

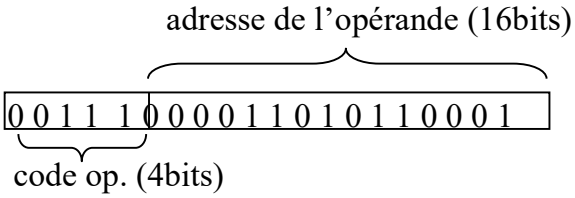
Les mots d'instruction contiennent tout ce qui est nécessaire à un µ-ordinateur pour réaliser ses diverses opérations. Le format et les codes employés varient grandement d'un µ-ordinateur à un autre. Mais dans la plus part des µ-ordinateurs les mots d'instruction précisent deux éléments d'informations de base:

1) Le code opération : qui indique au microprocesseur quelle opération à effectuer
2) La spécification de l'opérande : l'opérande peut être spécifié soit par sa valeur ou son adresse mémoire ou encore un numéro du registre.

Pour une opération nécessitant un seul opérande à retirer de la mémoire on a le format suivant :

Code op.	adresse de l'opérande

Exemple : mot d'instruction à 20bits



Le nombre de bits réservé pour le code opération détermine le nombre d'opération possible qu'un ordinateur peut effectuer, dans cet exemple, on a 2^4=16 opérations possibles différentes.

Dans le cas où l'opération utilise plus d'une seul opérande stockée dans la mémoire, le format du mot d'instruction est :

Code op.	Adresse de l'opérande 1	Adresse de l'opérande 2

Généralement, la longueur du mot dans les µ-ordinateurs ne permet pas d'avoir le code op. et l'adresse de l'opérande dans un seul mot à la fois.

Par exemple, pour un µ-ordinateur à 8bits (largeur de bus de donnée=longueur de mot machine = 8bits) et de largeur de bus d'adresse= 8bits, on va décrire le format du mot d'instruction en fonction du nombre d'opérande, se trouvant dans la mémoire, nécessaire pour faire l'opération en question.

Octet 1	Code op.
Octet 2	Adresse de l'opérande

Un seul opérande à récupérer de la mémoire

Octet 1	Code op.
Octet 2	Adresse de l'opérande1
Octet 3	Adresse de l'opérande2

Deux opérandes à récupérer de la mémoire

Sur certaines microordinateurs les instructions sont toutes de même longueur, sur d'autre cette longueur peut varier avec le code opération.

4 Les processeurs spécialisés

4.1 Micro-contrôleurs

Un microcontrôleur se présente sous la forme d'un circuit intégrés réunissant tous les éléments d'une structure à base de microprocesseur. Généralement, ils contiennent:

- Un microprocesseur (CPU);
- De la mémoire de données (RAM ou EEPROM);
- De la mémoire programme (ROM, OTPROM, UVPROM ou EEPROM);
- Des Interfaces d'entrées/sorties parallèles;
- Des Interfaces séries (synchrone et asynchrone) pour le dialogue d'autre unité;
- Des compteurs programmables (timers) pour générer ou mesurer des signaux avec une grande précision temporelle;
- des CAN/CNA pour le traitement des signaux analogiques.

Ils sont en général utilisés pour contrôler de simples machines (appareils électroménagers, lecteurs de carte à puce...)

Exemple de circuits :

- 80C186XX (80186, 16 bits, Intel)
- 68HC11, 68HC12 (6809, 8 bits, Motorola)
- 68HC16 (68000, 16 bits, Mototrola)
- C167XX (Infineon, ex Siemens)

4.2 Digital Signal Processor (DSP)

Ce sont des processeurs dédiés aux traitements des signaux numériques. Une architecture particulière leur permet un traitement efficace des fonctions complexes telles que FFT, convolution, filtrage numérique ...

Exemples :

- TMS320 (Texas Instrument)
- 2100 et 21000 (Analog Device)
- 56000 (Motorola)

4.3 Processeurs de traitement d'image

4.4 Processeurs spécialisés d'entrées/sorties

Chapitre 3 : Les Microprocesseurs

1 Architecture et constituants d'un microprocesseur

Le microprocesseur se compose essentiellement de l'unit de commande, de l'unité arithmétique et logique (UAL) et d'un ensemble de registres pour stocker les opérandes, les résultats intermédiaires ou les informations de commande, figure 1.

1.1 L'Unité Arithmétique et Logique (UAL)

Comme son nom l'indique, c'est l'unité responsable sur l'exécution des opérations arithmétiques tel que l'addition, la soustraction, l'incrémentation et la décrementation ; et les opérations logiques comme l'opération AND, OR, XOR et le décalage.

1.2 L'unité de commande

L'unité de commande contient une unité chargée du décodage des instructions pour déterminer l'opération à effectuer, une unité pour le calcul des adresses des opérandes (les données à traiter). On y trouve également le séquenceur qui contrôle le fonctionnement des autres composants en leur envoyant des signaux de commande (adresse mémoire valide, adresse E/S valide, lecture/écriture, attente, interruption, etc.). Il peut être câblé ou microprogrammé.

1.2.1 Séquenceur câblé :

Un séquenceur câblé est un circuit séquentiel complexe comprenant un sous-circuit pour chacune des instructions à commander. Ce sous-circuit est activé par le décodeur d'adresse.

1.2.2 Séquenceur micro-programmé :

L'idée de la microprogrammation a été introduite par Maurice Wilkes en 1951. Il est en effet toujours possible de remplacer un circuit logique par une ROM. De même pour reproduire une séquence d'opérations élémentaires il suffit d'un mot par

"tranche" de temps. Cette série de mots constitue un microprogramme. Le code opération de l'instruction à exécuter peut être utilisé pour définir le pointeur sur la première micro-instruction du microprogramme, figure 2. En fonction du code opération le contenu d'un compteur est initialisé, puis celui-ci s'incrémente ensuite à chaque cycle d'horloge.

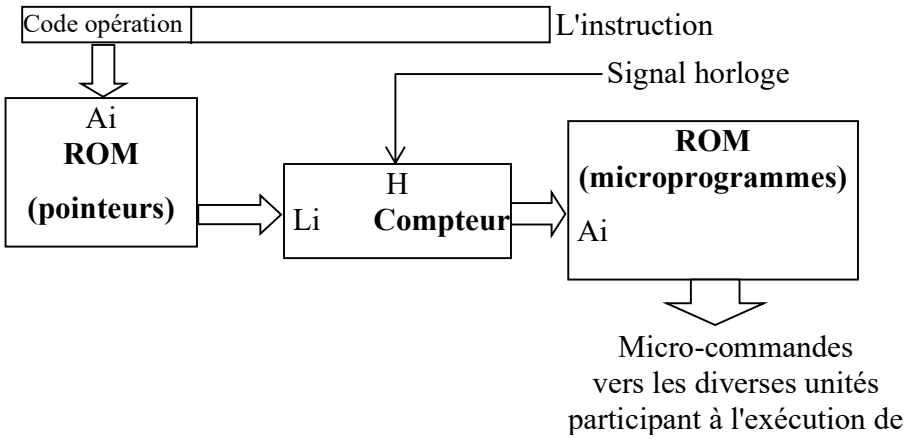


Figure 2 : Principe du séquenceur micro-programmé

1.3 Registres du microprocesseur

Le nombre et le type des registres implantés dans le microprocesseur font partie de son architecture et ont une influence importante sur la programmation et les performances du microprocesseur. On voudrait ici passer en revue les registres fondamentaux, que l'on retrouve sur tous les microprocesseurs ou presque.

1.3.1 Compteur Ordinal (CO)

Ce registre -appelé aussi (Program Counter : PC)- contient l'adresse de la prochaine instruction à exécuter. Après chaque utilisation il est automatiquement incrémenté du nombre de mots correspondant à la longueur de l'instruction traitée : le programme est exécuté en séquence. En cas de rupture de séquence (branchement conditionnel ou non, appel à une routine, etc.) il est chargé avec la nouvelle adresse. Le compteur ordinal, dont la taille dépend de l'espace adressable, n'est généralement pas accessible directement au programmeur.

1.3.2 Registre Instruction (RI)

C'est le registre de destination dans lequel le CPU transfert l'instruction suivante à partir de la mémoire. Sa taille dépend du format des instructions machines. Le décodeur utilise le registre instruction pour identifier l'action (ou le microprogramme) à entreprendre ainsi que les adresses des opérandes, de destination ou de saut. Le programmeur n'a pas accès au registre instruction.

I.3.3 Accumulateur (Acc) :

L'accumulateur est un registre de l'unité arithmétique et logique. Il a de nombreuses fonctions. Il peut contenir un des deux opérandes avant l'exécution et recevoir le résultat après. Cela permet d'enchaîner des opérations. Il peut servir de registre tampon pour les opérations d'entrées/sorties : dans certaines microprocesseurs c'est le seul registre par lequel on peut échanger des données directement avec la mémoire. Sa taille est égale à la longueur des mots en mémoire. Il possède souvent une extension (Q), pour les multiplications, décalages, divisions, etc. Le registre Acc est accessible au programmeur et très sollicité. Certains microprocesseurs possèdent plusieurs accumulateurs, par exemple, le 6800 et le 6502 possèdent deux accumulateurs.

1.3.3 Registres généraux ou banalisés :

Ils permettent de limiter les accès à la mémoire, ce qui accélère l'exécution d'un programme. Ils peuvent conserver des informations utilisées fréquemment, des résultats intermédiaires, etc. Ils sont accessibles au programmeur.

1.3.4 Registres d'indice ou d'index (XR) :

Ils peuvent être utilisés comme des registres généraux mais ils ont une fonction spéciale utilisée pour l'adressage indexé. Dans ce cas l'adresse effective d'un opérande est obtenue en ajoutant le contenu du registre d'index à l'adresse contenue dans l'instruction. Ce type d'adressage et de registre est très utile pour manipuler des tableaux. Le programmeur dispose alors d'instructions permettant l'incrémentation ou la décrémentation du registre d'index. En particulier les registres d'index peuvent être incrémentés ou décrémentés automatiquement après chaque utilisation. Dans

certaines microprocesseurs ces instructions sont applicables à tous les registres généraux, il n'y a alors pas de registre d'index spécifique.

1.3.5 Registre de base :

A de très rares exceptions à l'intérieur d'un programme on ne fait référence qu'à des adresses relatives ou virtuelles. Par contre l'unité centrale a besoin de connaître les adresses physiques où se situent réellement instructions et données. Celles-ci dépendent de l'endroit où a été chargé le programme en mémoire, l'espace physique occupé par un programme pouvant ne pas être contigu. Le rôle des registres de base est de permettre le calcul des adresses effectives. Un registre de base contient une adresse de référence, par exemple l'adresse physique correspondant à l'adresse virtuelle 0. L'adresse physique est obtenue en ajoutant au champ adresse de l'instruction le contenu du registre de base. Le registre de base est encore utilisé quand le nombre de bits du champ adresse ne permet pas d'accéder à toute la mémoire.

1.3.6 Registre d'état (Program Status Word : PSW) :

Une partie des bits de ce registre, aussi appelé registre condition, constitue des indicateurs ou des drapeaux (flags) qui indiquent certains états particuliers. Par exemple à la fin de chaque opération on peut y trouver le signe du résultat (Négatif, Zéro ou Positif), ainsi qu'une éventuelle retenue (Carry) ou un dépassement de capacité (Overflow). Ces bits indicateurs peuvent être testés pour déterminer la suite du déroulement du programme : branchements conditionnels. On trouve également le mode de fonctionnement de l'unité centrale. Deux modes sont possibles le mode utilisateur et le mode système ou superviseur. Dans le mode utilisateur certaines instructions sont interdites : elles provoquent un déroutement vers le système d'exploitation. Un bit peut également indiquer un déroulement pas à pas : demande de trace (T). Le registre peut aussi contenir le niveau de l'interruption en cours de traitement ou un masque des niveaux d'interruptions autorisés.

1.3.7 Registre Pointeur de Pile (PP) :

Une pile est une zone mémoire dans laquelle les informations sont rangées de façon contiguë. Le pointeur de pile (ou bien Stack Pointer : SP) indique le sommet de la pile : la position de la dernière information enregistrée. Dans certaines machines le pointeur de pile indique la position où sera mémorisée la prochaine donnée. Le fonctionnement d'une pile est du type Dernier Entré Premier Sorti (LIFO : Last In First Out). Les deux principales opérations liées à la pile concernent l'ajout d'un élément dans la pile ou le retrait, souvent nommées respectivement PUSH et PULL. Lorsqu'une donnée est enregistrée dans la pile elle est placée à l'adresse qui suit celle du dernier mot stocké. Après l'opération le pointeur de pile est incrémenté. Lorsqu'un mot est retiré de la pile il correspond à la dernière information qui y a été entrée. Après l'opération le pointeur est décrémenté. Une pile est réservée à l'usage de l'unité centrale, en particulier pour sauvegarder les registres et l'adresse de retour en cas d'interruption ou lors de l'appel d'une procédure. Le pointeur de pile est accessible au programmeur, ce qui est souvent source d'erreur. Certaines machines sont dotées de plusieurs pointeurs de piles.

2 Séquencement interne pour l'exécution d'instructions

Quel que soit le format de l'instruction et par conséquent le nombre de champs adresse, son traitement peut être décomposé en trois phases.

2.1 Phase 1 : Recherche de l'instruction à traiter (cycle fetch)

1) Le CO contient l'adresse de l'instruction suivante du programme. Cette valeur est placée sur le bus d'adresses par l'unité de commande qui émet un ordre de lecture à la mémoire.
2) Au bout d'un certain temps (temps d'accès à la mémoire), le contenu de la case mémoire sélectionnée est disponible sur le bus des données à travers le registre tampon de données.
3) L'instruction est stockée dans le registre instruction du microprocesseur.

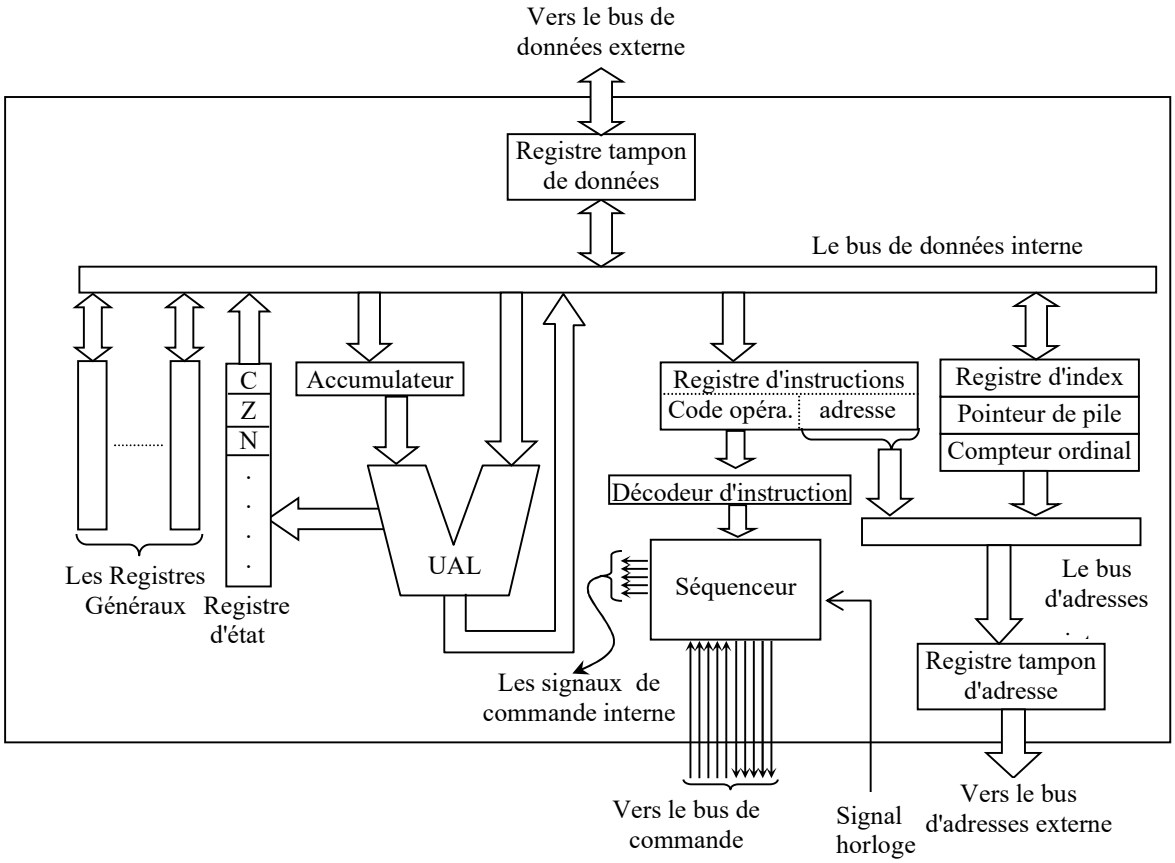


Figure 1. Architecture standard d'un microprocesseur

2.2 Phase 2 : Décodage de l'instruction et recherche de l'opérande

Le registre d'instruction contient maintenant le premier mot de l'instruction qui peut être codée sur plusieurs mots. Ce premier mot contient le code opération qui définit la nature de l'opération à effectuer (addition, rotation,...) et le nombre de mots de l'instruction.

1) L'unité de commande décode le code opération et transforme l'instruction en une suite de commandes élémentaires nécessaires à l'exécution de l'instruction.

2) Si l'instruction nécessite un opérande en provenance de la mémoire ou de l'unité d'entrée, l'unité de commande récupère sa valeur sur le bus de données.
3) L'opérande est stocké dans un registre.

2.3 Phase 3 : Exécution de l'instruction (cycle d'exécution)

1) Le microprogramme ou le sous-circuit réalisant l'instruction est exécuté.
2) Les drapeaux sont positionnés dans le registre d'état.
3) Le résultat est rangé dans la mémoire s'il faut.
4) L'unité de commande incrémente le compteur ordinal, ainsi il contiendra l'adresse de l'instruction suivante. Notons cependant que cette opération est inutile si l'instruction est un branchement. En effet, l'adresse de la prochaine instruction se trouve dans le champ spécification adresse de l'instruction.

3 Le jeu d'instruction:

Il caractérise le microprocesseur. Mais les types fonctionnels sont identiques :

1) Instructions de transfert
2) Instructions arithmétiques
3) Instructions logiques
4) Instructions de décalages, de rotations
5) Instructions de test et de comparaison
6) Instruction de branchement
7) Autres instructions spécifiques au microprocesseur

4 Mode d'adressage

Le mode d'adressage indique comment le microprocesseur accède aux opérandes nécessaires pour exécuter une instruction.

Le mode est défini soit par le code opération lorsque celui-ci impose un type déterminé, soit par un code faisant partie du champ adresse.

Pour faciliter la programmation, il existe de nombreux modes d'adressage. On peut distinguer 8 modes d'adressage fondamentaux, tableau 1.

4.1 Adressage implicite :

Aucun champ adresse n'est nécessaire. Le code opération identifie automatiquement l'opérande, l'instruction ne peut porter que sur un registre particulier (par exemple test sur le signe du résultat d'une opération arithmétique : concerne le registre d'état PSW).

4.2 Adressage immédiat (de niveau 0) :

La valeur de l'opérande est contenue dans le champ adresse si le nombre de bits dans ce champ est suffisant, sinon dans le mot suivant l'instruction.

4.3 Adressage registre :

Le champ adresse contient le numéro du registre opérande.

4.4 Adressage direct absolu (de niveau 1) :

Le champ adresse de l'instruction (ou le mot suivant si le nombre de bits n'est pas suffisant) contient l'adresse effective de l'opérande.

4.5 Adressage indirect :

Le champ adresse (ou le mot suivant) contient l'adresse d'un pointeur : mot en mémoire qui contient l'adresse effective de l'opérande.

4.6 Adressage indexé :

Ce mode d'adressage est très utile lorsqu'on travaille, par exemple, sur des tableaux. Considérons un bloc de n mots consécutifs débutant à l'adresse A. Le k ième mot se trouve à l'adresse A + (k-1). Pour référencer ce mot il est possible d'utiliser un registre d'index.

L'adresse effective est calculée en additionnant le contenu de ce registre d'index à l'adresse qui se trouve dans le champ adresse de l'instruction. Sur certaines machines tous les registres généraux peuvent être utilisés comme registres d'index.

La présence d'un registre d'index s'accompagne généralement de la possibilité d'incrémentation et décrémentation automatiques.

4.7 Adressage basé :

L'adressage basé est comparable à l'adressage indexé mais cette fois l'adresse effective est obtenue en additionnant le contenu du registre de base au contenu du champ adresse de l'instruction. Ce mode d'adressage est utilisé par exemple en cas d'allocation dynamique de la mémoire : la position du programme en mémoire peut changer en fonction de la charge du système et il n'occupe pas toujours un espace contigu. Cette technique permet également de réduire le nombre de bits dans le champ adresse : le registre de base contient la première adresse d'un bloc de 2 k mots et l'adresse (sur k bits) contenue dans l'instruction représente le déplacement à l'intérieur du bloc.

4.8 Adressage relatif :

Le champ adresse contient un déplacement (offset) par rapport à une adresse de référence. L'adresse de référence est normalement celle contenu dans le compteur ordinal. Ainsi, une instruction avec adressage relatif va permettre un adressage en avant ou en arrière dans la mémoire par rapport au contenu du compteur ordinal.

L'adresse effective est donc obtenue en additionnant le contenu du compteur ordinal au contenu du champ adresse de l'instruction. Ce type d'adressage est utilisé par exemple dans des instructions de branchement.

L'avantage principal de ce mode d'adressage est qu'il permet des branchements efficaces, en minimisant le nombre d'octets utilisés.

Remarque:

Le calcul de l'adresse effective de l'opérande peut nécessiter quelques opérations (addition par exemple). L'utilisation de certains modes d'adressage sophistiqués (le 68020 de Motorola dispose par exemple d'une cinquantaine de modes d'adressage) peut donc augmenter le temps de traitement d'une instruction.

Type d'adressage	Format de l'instruction	Calcule de l'adresse effective de l'opérande
Implicite	Code opération	----------
Immédiat	Code opération | opérande	----------
Direct	Code opération | adresse de l'opérande	----------
Registre	Code opération | n° du registre opérande	----------
Indirect	Code opération | pointeur d'adresse	----------
Indexé		Contenu du registre d'index + déplacement
Basé	Code opération | déplacement (offset)	Contenu du registre de base + déplacement
Relatif		Contenu du CO + déplacement

Tableau 1 : Les modes d'adressage.

5 Langage de programmation

Un programme est constitué d'une suite organisée d'instructions. Il peut être écrit dans des langages divers, *symboliques* ou non, et de différents *niveaux*

5.1 Le langage machine (non-symbolique)

Le langage machine est le langage compris par le microprocesseur. Chacune de ses instructions correspond à une opération élémentaire du microprocesseur considéré. Ce langage est difficile à maîtriser puisque chaque instruction est codée par une séquence propre de bits. Afin de faciliter la tâche du programmeur, on a créé différents langages plus ou moins évolués.

5.2 Le langage d'assemblage (langage symbolique bas niveau)

Le langage d'assemblage est le langage symbolique le plus proche du langage machine. Il est composé par des instructions en général assez rudimentaires que l'on appelle des mnémoniques. Ce sont essentiellement des opérations de transfert de données entre les registres et l'extérieur du microprocesseur (mémoire ou périphérique), ou des opérations arithmétiques ou logiques. Chaque microprocesseur peut posséder un langage d'assemblage différent.

Pour qu'un programme écrit en langage d'assemblage devienne exécutable, il faut le traduire dans le langage machine du microprocesseur utilisé pour son exécution en utilisant un logiciel dit assembleur.

A chaque instruction en langage d'assemblage correspond une instruction en langage machine (c-à-d opération élémentaire du microprocesseur). Il en résulte que l'assembleur n'est jamais un logiciel très encombrant

5.3 Le langage évolué (langage symbolique haut niveau)

La difficulté de mise en œuvre du langage d'assemblage et leur forte dépendance avec la machine a nécessité la conception de langages évolués, plus adaptés à l'homme, et aux applications qu'il cherchait à développer. Faisant abstraction de toute architecture de machine, ces langages permettent l'expression d'algorithmes

sous une forme plus facile à apprendre, et à dominer (C, Pascal, Fortran, Java,... etc.). Une fois développé, le programme en langage évolué n'est donc pas compréhensible par le microprocesseur. Il faut le compiler pour le convertir en code machine compréhensible par le microprocesseur. Cette opération est réalisée par un logiciel spécialisé appelé le compilateur.

Une instruction en langage évolué sera traduite par le compilateur en une succession d'instructions du langage machine donc un compilateur est un logiciel de taille plus importante qu'un assembleur.

6 Notion des Technologies RISC et CISC

Actuellement les microprocesseurs se composent en deux grandes familles :

-Microprocesseurs à jeu d'instructions complexe (CISC: Complex Instruction Set Computer)

-Microprocesseurs à jeu d'instructions réduit (RISC: Reduced Instruction Set Computer)

6.1 Les microprocesseurs CISC

Les microprocesseurs à jeu d'instructions complexe (CISC) disposent d'un jeu étendu d'instructions avec de nombreux modes d'adressage. La plupart ne sert que dans des cas relativement rares. Ces instructions complexes nécessitent d'être microcodées et s'exécutent donc en plusieurs cycles. Cette technologie est basée sur un jeu de plus de 400 instructions

6.2 Les microprocesseurs RISC

Les microprocesseurs à jeu d'instructions réduit (RISC) sont constitués d'instructions simples qui permettent de gagner une rapidité d'exécution mais au détriment d'une programmation plus complexe. En effet, celui-ci n'offre que 128 instructions - dites de base- qui peuvent être câblés (donc sans microcode) ce qui permet une exécution des instructions en un seul cycle horloge. De plus, cette technologie a permis la création de nouveaux procédés comme la multiplication des unités spécialisées.

Un processeur RISC peut atteindre une vitesse d'exécution jusqu'à 70% plus rapide qu'un CISC de même fréquence.
Depuis le P5 (pentium), les microprocesseurs utilisent des technologies empruntées de la famille RISC.

6.3 La technologie RISC contre celle de CISC

Le choix entre les deux technologies dépendra des applications visées. En effet, si on diminue le nombre d'instructions, on crée des instructions complexes (CISC) qui nécessitent plus de cycles pour être décodées et si on diminue le nombre de cycles par instruction, on crée des instructions simples (RISC) mais on augmente alors le nombre d'instructions nécessaires pour réaliser le même traitement.

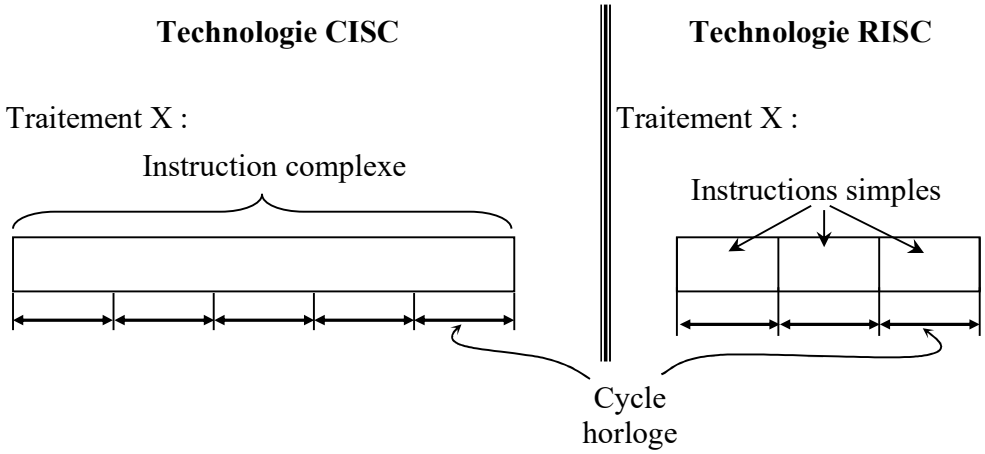


Figure 3. Comparaison entre la technologie CISC et RISC pour un même traitement.

Une comparaison objective entre les deux technologies est présentée dans le tableau 2.

Technologie RISC	Technologie CISC
Instructions simples ne prenant qu'un seul cycle.	Instructions complexes prenant plusieurs cycles.
Instructions au format fixe	Instructions au format variable
Décodeur d'instruction simple (câblé)	Décodeur d'instruction complexe(microcodé)
Beaucoup de registres	Peu de registres
Seules les instructions LOAD et STORE ont accès à la mémoire	Toutes les instructions sont susceptibles d'accéder à la mémoire
Peu de modes d'adressage	Beaucoup de modes d'adressage
Compilateur complexe	Compilateur simple

Tableau 2 Comparaison entre les technologies CISC et RISC

Chapitre 4 : Généralités sur les Microcontrôleurs

1 Définition d'un microcontrôleur :

Un microcontrôleur est un circuit intégré regroupant tous les éléments d'une structure à base de microprocesseur c-à-d d'un ordinateur :

- Un microprocesseur(CPU) ;
- Mémoire de données ;
- Mémoire de programmes
- Une horloge ;
- Des périphériques qui vont dépondre, en générale du type de microcontrôleur choisi :
  - Temporisateurs/Compteurs,
  - Convertisseur analogique numérique (CAN),
  - Chien de garde ('Watch dog'),
  - Des interfaces d'entrée/sortie série (synchrone ou asynchrone),
  - Un contrôleur d'interruptions,
  - Un contrôleur de bus spéciaux.
  - ….. etc.

Il est généralement moins puissant qu'un ordinateur en terme de rapidité et de taille mémoire, il se contente le plus souvent d'un bus de données de 8 ou 16 bits. Ceci en fait un composant très bon marché parfaitement adapté pour piloter les applications embarquées dans de nombreux domaines d'application.

2 Les avantages du microcontrôleur :

- Encombrement de matériel réduit,
- Circuits imprimé peu complexe,
- Faible consommation à cause de l'intégration en technologie MOS, CMOS, ou HCMOS,

- Coût réduit de conception et montage
- Environnement de programmation et de simulation évolués.

3 Les inconvénients du microcontrôleur :

- Ne convient pas nécessairement à tous les problèmes car il n'offre pas une grande puissance de calcul.
- On ne peut pas toujours utiliser tous les périphériques simultanément : pour réduire les coûts, certaines broches sont multiplexées.
- Il faut disposer d'un outil du développement spécifique, compilateur, ou simulateur.

1 Les familles des microcontrôleurs

Plusieurs fabricants se partage le marché des microcontrôleurs, citons INTEL, MOTOROLA, ATMEL, ZILOG, HITACHI, DALLAS SEMI, PHILIPS et enfin MICROCHIP. Chaque fabricant ne propose pas un seul microcontrôleur, mais des familles de microcontrôleurs. On peut citer à titre d'exemple :

La famille MCS51 (8x31,8x51) de INTEL, la famille AVR (ATtiny 28, ATMEGA161, AT90S8535) de ATMEL, la famille 68HCxxx de MOTOROLA (68HC11, 68HC811), la famille PIC 16Cxx (16C84,16F84) de MICROCHIP.

Au sein de même famille les microcontrôleurs possèdent le même microprocesseur et donc le même langage, seul les périphériques changent. Ainsi, la connaissance de la structure matérielle et logicielle d'un produit d'une famille permet une adaptation rapide à tout microcontrôleur de même famille.

Les microcontrôleurs du marché se distinguent principalement par la structure de leurs µprocesseurs (4,8,16 ou 32 bits, CISC ou RISC), la taille des espaces mémoires, la nature et le nombre des périphériques. Certains µcontrôleurs seront spécialisés dans la gestion des entrées/sorties, d'autres auront la possibilité de gérer des grandeurs analogiques ou posséderont des ports I^2C ou de CAN.

Sur le plan logiciel, outre le fait que le jeu d'instructions des µCs 8bits est généralement restreint (autant plus s'il est à architecture RISC), on dispose pour

certains microcontrôleurs des environnements de développement intégrés contenant un assembleur, un cross compilateur C et même parfois un simulateur.

4 Le choix du microcontrôleur :

Lorsqu'on décide de développer un nouveau produit à base de µC, puisque l'offre est très vaste, plusieurs paramètres vont orienter notre choix vers un produit plutôt qu'un autre :

1) Le prix : il y a de grand écart de prix entre les produits, liés par exemple à la taille et au type de mémoire, ainsi qu'à la nature et le nombre de périphériques.
2) Les périphériques : on peut se demander si toutes les fonctions décrites dans le cahier des charges seront réalisées par le µC ou s'il faut ajouter des périphériques externes.
3) La taille des espaces mémoire : L'espace mémoire programme sera-t-il suffisant pour l'application ?
4) La consommation électrique : déterminant pour de produits destinés à fonctionner sur batterie.
5) Les outils de développement de programmes : peut-on développer en langage évolué ? Existe-t-il un environnement de développement intégré (éditeur, assembleur, compilateur, simulateur/debugger) ?
6) Expérience/savoir faire.

5 Applications des microcontrôleurs :

1) Informatique (souri, modem,…),
2) Vidéo (appareil photos numérique, caméra numérique, …),
3) Contrôle des processus industriels (régulation, pilotage, supervision, …),
4) Appareil de mesure (affichage, calcul statistique, mémorisation, …),
5) Automobile (ABS, injection, GPS, airbag, …),
6) Multimédia (téléviseur, carte audio, carte vidéo, MP3, magnétoscope, …)
7) Téléphone (fax, téléphone portable, modem, …),

8) Électroménager (Lave-vaisselle, lave-linge, four micro-onde, …),

6 Architecture d'un microcontrôleur :

On peut envisager le schéma bloc général d'un microcontrôleur de la figure 1, en sachant que certaines fonctions peuvent être absentes d'un type à l'autre.

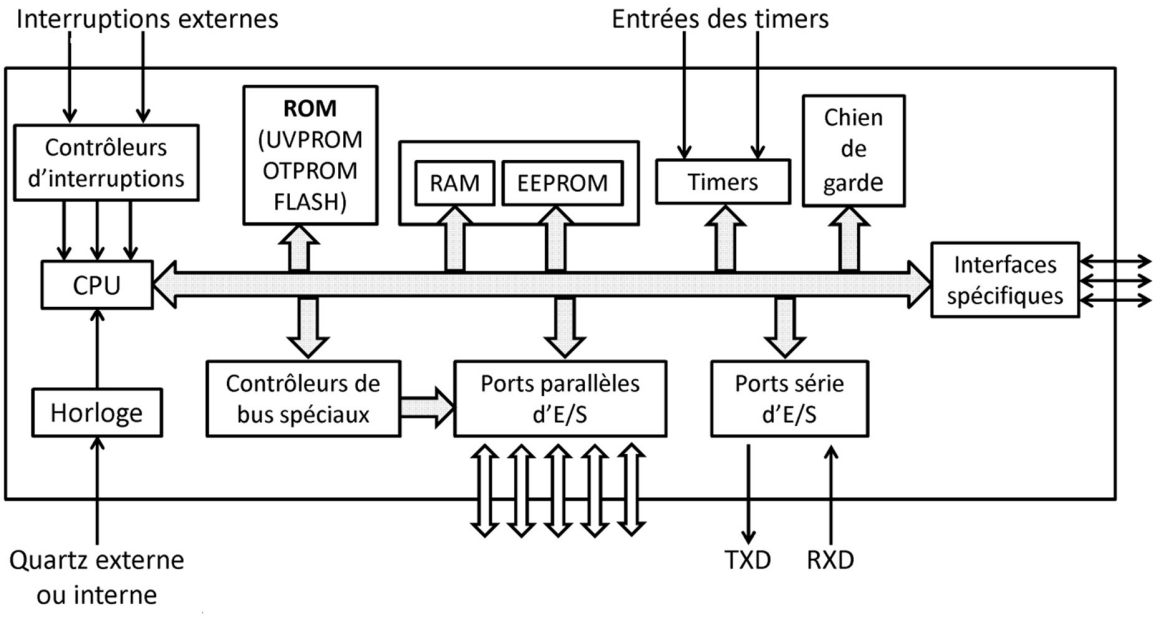


Figure 1. Architecture générale d'un microcontrôleur.

1) **La mémoire programme 'ROM'** est de plus en plus souvent reprogrammable par différentes techniques : ultra-violet (UVPROM), une seule fois (OTP), par block mémoire (FLASH) et même si le µcontrôleur est soudé sur le circuit imprimé en utilise la fonctionnalité de la programmation in-situ (In-System Programming ou ISP). Des lignes d'E/S sont spécifiquement dédiées à cette fonction.
2) **Les mémoires de données 'RAM et EEPROM'** permettent de mémoriser temporairement les données générées par le microprocesseur pendant les différentes phases du traitement numérique (résultats des opérations, états de capteurs, … etc.).

3) **L'horloge** est souvent un quartz entouré de deux condensateurs, mais il peut aussi être réalisé en interne par un circuit RC que l'on peut calibrer lors de la première utilisation du microcontrôleur.
4) **Le chien de garde 'watchdog'** est un temporisateur particulier qui génère, dans un délai préfixé ou programmé, une impulsion sur le reset du microcontrôleur. Son utilité est de permettre le redémarrage du microcontrôleur en cas de problème (exemple, la présence d'une boucle sans fin accidentelle).
5) **Les Timers** sont des temporisateurs ou compteurs programmables. Ils permettent la génération des impulsions calibrés ou des signaux PWM, et la mesure du temps ou d'évènements. Ils peuvent générer des interruptions et s'accompagner de pré-diviseur (prescaler).
6) **Les interfaces spécifiques :** Certain microcontrôleurs possèdent des entrées comparateur, ou encore des convertisseurs A/D et D/A très utiles dans la gestion des processus automatisés.
7) **Les ports série d'entrées/sorties** permettent l'échange de données entre le microcontrôleur et un périphérique bit par bit sur un fil unique, d'une manière séquentielle et suivant un ordre précis. Il y a deux types de liaison série : synchrone et asynchrone.
8) **Les contrôleurs de bus spéciaux de communication** sont parfois présents par exemple : UART, I2C, SSP, CAN, FlexRay, USB, Ethernet, ...etc. Ils rendent le microcontrôleur plus onéreux mais nécessaires si l'application est basé sur un de ces bus.
9) **Le contrôleur d'interruptions** est indispensable dans un microcontrôleur. Exceptionnels sont les cas où il ne faut pas envisager au moins un cas d'urgence ou tenir compte d'événement asynchrones à l'exécution du programme. L'interruption est un programme qui permet l'interruption de l'exécution d'un programme (mais pas d'une instruction), pour réaliser une tâche prioritaire généralement de courte durée.
10) **Les ports parallèles d'entrées/sorties** permettent l'échange de données entre le microcontrôleur et un périphérique généralement en 8 bits. Ils

permettent de recueillir des informations en entrée (états de capteurs) ou d'envoyer des signaux binaires en sortie pour commander des actionneurs et piloter des modules de l'environnement extérieur.

Les broches de ces ports (Les lignes d'E/S ou encore le Pins) sont configurable individuellement en entrée ou en sortie.

La configuration ainsi que l'état logique de ces pins est effectuée par des opérations d'écriture et de lecture dans différents registres associés à chaque port. On trouve généralement :

- Un registre de direction pour une configuration en entrée ou en sortie,
- Un registre de donnée recopiant les états logique de chaque pin de port configuré en entrée,
- Un registre de donnée contrôlant le niveau logique de chaque pin de port configuré en sortie.

7 Modes de fonctionnement électrique du microcontrôleur :

Un microcontrôleur possède plusieurs modes ou états de fonctionnement. On distingue principalement le mode normal, le 'sleep mode' le 'idel mode' et le 'power down mode'.

Les spécificités de chaque mode doivent être examinées dans la fiche technique du microcontrôleur, mais le principe est de réduire la consommation tout en préservant les données acquises.

Chapitre 5 : Micro-Contrôleur ATtiny 2313

1 Description du Microcontrôleur ATtiny 2313

ATtiny 2313 est un microcontrôleur conçu par la firme ATMEL. C'est un microcontrôleur 8-bits CMOS à basse puissance et de haute performance basé sur l'architecture RISC renforcée d'AVR, figure 1. Il possède les caractéristiques générales suivantes:

- 120 instructions puissante- La plus part s'exécute en un seul cycle de base (sauf les instructions de saut ou d'accès mémoire).
- Architecture Harvard est utilisée
- 32x8 (General Purpose working Registers) registres de travail à usage général
- Une vitesse d'exécution de 20 MIPS (Million Instruction Per Second) à une fréquence de 20MHz.

Il se présente dans de différents boîtiers, figure 2:

- 20-pin PDIP(Plastic Dual In-Line Package),
- 20-pin SOIC (Small Outline Integrated Circuit),
- 20-pad QFN/MLF(Quad Flat No Leads/ Micro Lead Frame).

Son architecture est donnée sur la figure 3, illustrant les composantes suivantes :

Mémoires:

- 2K Octets (In-System Self Programmable Flash) mémoires Flash auto-programmable
- 128 Octets (In-System Programmable EEPROM) mémoire programmable dans le système
- 128 Octets SRAM Interne
- Verrou de programmation pour la sécurité des programmes dans Flash et les données dans L'EEPROM

Les périphériques:

- Timer/compteur 8-bit avec un diviseur de fréquence et mode de comparaison séparés
- Timer/compteur 16-bit avec un diviseur de fréquence et mode de comparaison et d'acquisition séparés
- 4 canaux PWM
- Comparateur analogique sur puce
- Un timer chien de garde programmable avec oscillateur interne
- USI-Universal Serial Interface
- USART en duplex Integral
- 3 ports parallèles d'entrée/sortie (port A :3bit, port B : 8bits, et port D : 7bits) fournissant 18 Lignes d'E/S programmable.

Les caractéristiques spéciales de ce microcontrôleur sont :

- debugWIRE On-Chip Debugging permet l'accès en lecture ou écriture de toutes les mémoires et un contrôle complet des étapes d'exécution. Cela inclus l'exécution par étape (single step), run-to-cursor, setp-out et software break instructions. Elle permet aux programmeurs de tester leur travail.
- programmation in-situ (In-System Programmable ou ISP) est une fonctionnalité qui permet au microcontrôleur d'être programmer ou reprogrammer alors qu'il est placé en système via un Port SPI (Serial Peripherical Interface)
- Sources d'interruptions externes et internes
- Low-power Idle, Power-down, and Standby Modes
- Circuit de détection de chute de tension programmable (Programmable Brown-out Detection Circuit)
- Oscillateur interne calibré

Les tensions de travail sont: – 1.8 - 5.5V (ATtiny2313V)
– 2.7 - 5.5V (ATtiny2313)

La vitesse de fonctionnement est :

- ATtiny2313V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
- ATtiny2313: 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V

Sa consommation électrique typique est :

- Mode Actif 1 MHz, 1.8V: 230 µA
  32 kHz, 1.8V: 20 µA (including oscillator)
- – Mode de ‘Power-down’ < 0.1 µA at 1.8V

Le µc ATtiny 2313 peut être supporté par un bon nombre de programme et d'outils de développent de systèmes comme : Compilateur C, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, et des Kits d'Evaluation.

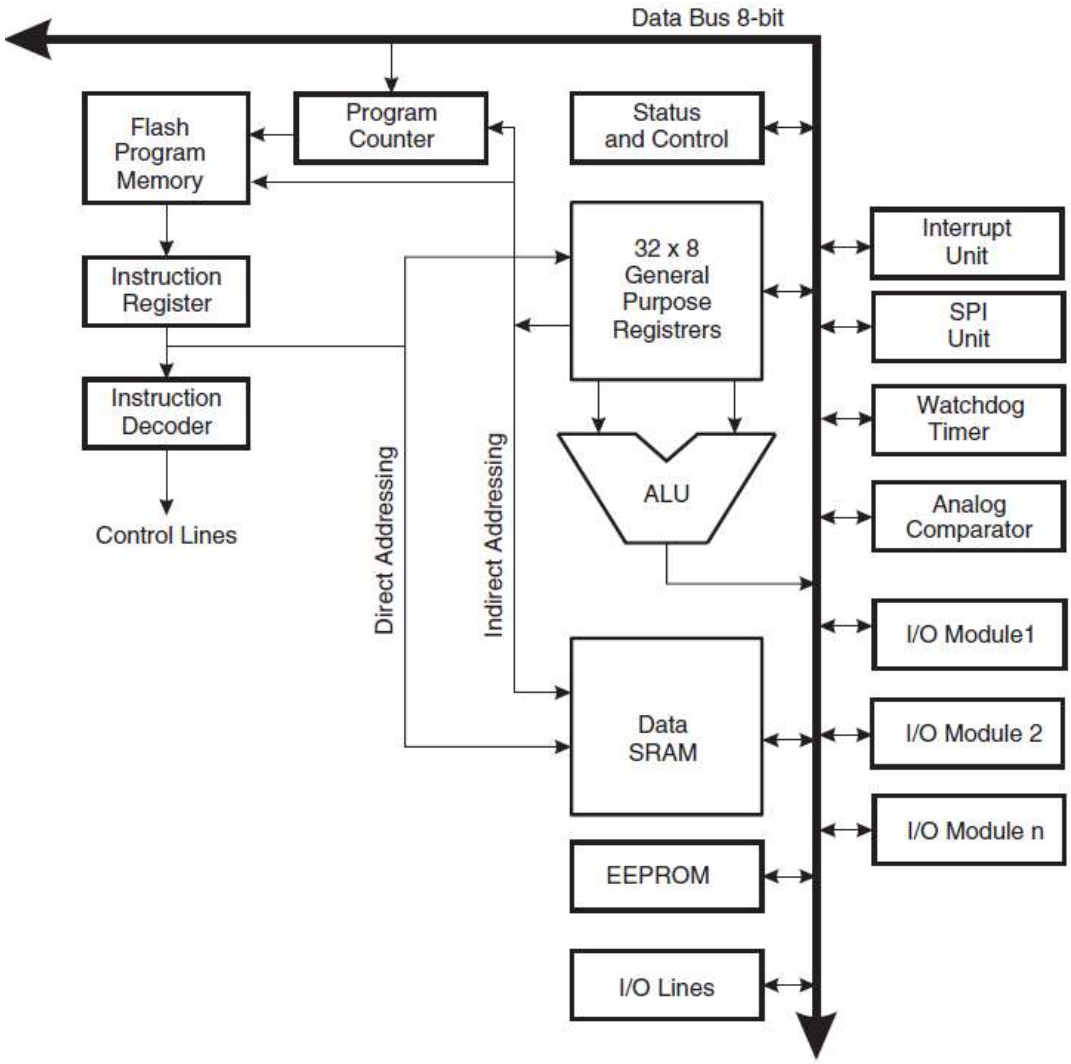


Figure 1. L’architecture RISC AVR.

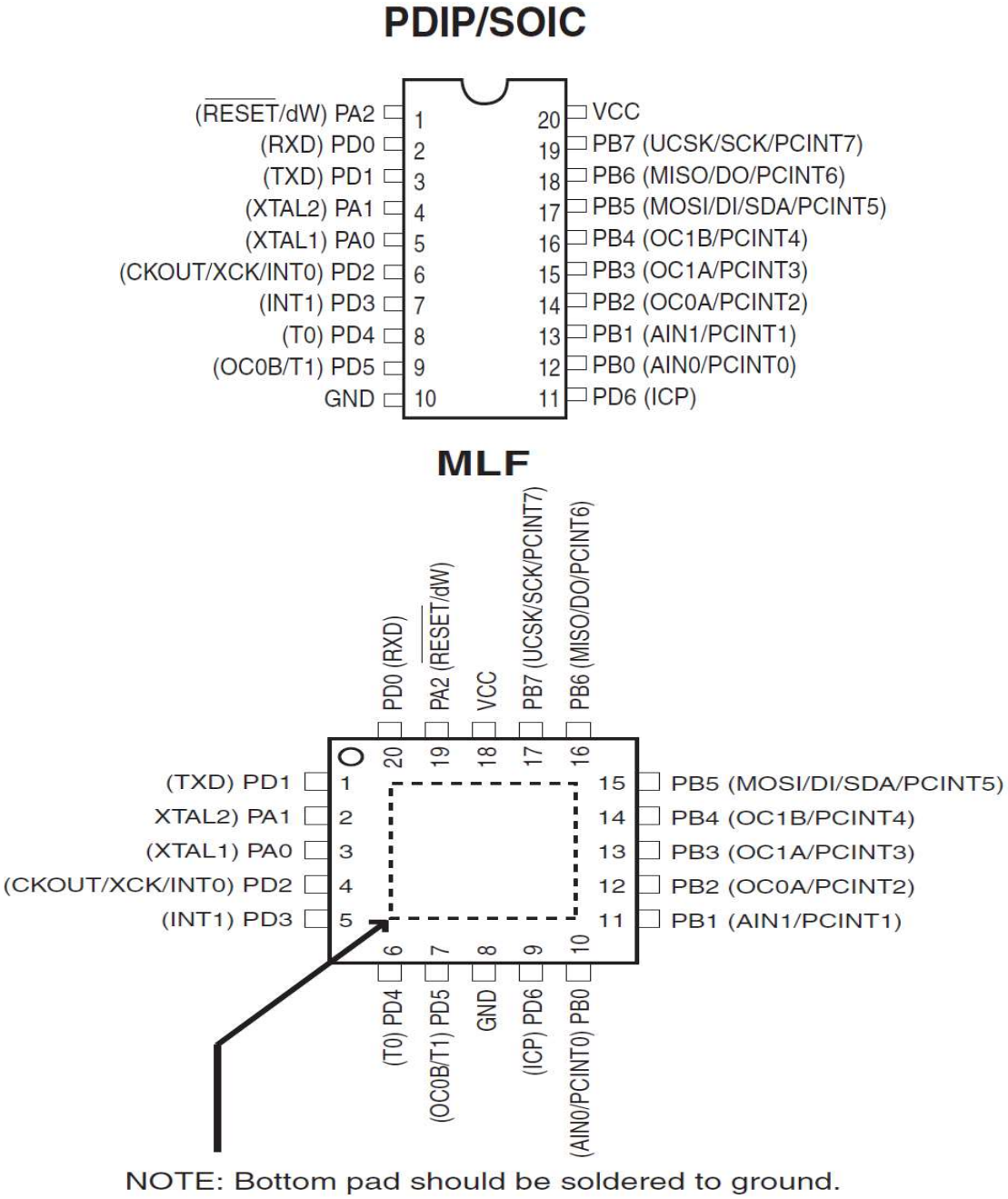


Figure 2. Les différents types de boîtiers du microcontrôleur ATtiny 2313

2 Description des broches du µC ATtiny 2313 :

Alimentation :

VCC : suivant les versions quand elles existent : Soit une alimentation variant de 4.5V à 5.5V, soit une alimentation variant de 2.7V à 5.5V. Et même 1.8V pour certain (suffixe V ou aucun suffixe).

GND : la masse : 0Volt.

$\overline{\boldsymbol{RESET}}$ **:** La broche de reset est une entré qui réinitialise le µC. Elle est activée par un niveau logique bas qui doit avoir une durée opportune. Habituellement, le temps de $\overline{RESET}$ tourne autour de 50 ns. Des temps plus courts n'assurent pas la génération du $\overline{RESET}$

Comparateur analogique :

AIN0, AIN1 : Permet de comparer deux tensions et peut fonctionner simultanément avec un Timer/compteur et déclencher une interruption.

Timer/compteur

OC0A , OC0B, OC1A, OC1B, OCP, T0, T1 : Permet de contrôler, de comparer ou de compter des temps.

Horloge

XTAL1 XTAL2 : Broche utilisée par le quartz ou le résonateur. Les circuits ATMEL sont fournis avec les fusibles d'horloge réglés sur un oscillateur RC interne de fréquence 1MHz. Si l'utilisateur veut utiliser ces broches (quand elles sont a usage multiples) pour connecter un quartz ils doit configurer les fuses-bits. Ces broches ne sont plut utilisables pour d'autres fonctions, sauf changement des fusibles.

Transmission Série USART

RXD, TXD, XCK, USCK, DO, DI, SCL : Dans le cas d'une utilisation en port série, Les broches RXD et TXD ne sont plut utilisables pour d'autre fonctions. XCK est une clock pour les transmissions synchrones. DI (Data Input) entrée de donnée série, DO (Data Output) Sortie de donnée série. USCK : Three_wire mode Universal Serial Interface Clock. SCL : Two-wire mode Serial Clock for USI Two-wire mode.

SERIAL PERIPHERICAL INTERFACE (SPI)

SCL(USCK), SCK : Permet de programmer les µC ou d'établir des liaisons rapides entre différents périphériques.

TWO-WIRE INTERFACE(I2C)

SCL(USCK), SDA : permet d'établir des liaisons courtes sur 2 fils entre différents µC. Equivalent au protocole I2C®.

Interruptions

PCINT0…7 : interruptions extérieures permet de déclencher un sous-programme quand la broche change d'état.

DebugWire On-Chip Debugging

Dw :Reset ou break pour le débugage.

Les ports d'entrée/sortie :

Port A (PA2…PA0), port B (PB7…PB0), port D (PD6…PD0) se sont des ports de E/S bidirectionnel. Toutes les pattes du port ont des résistances internes de rappelle « pull-up ». Le buffer de sortie est en mesure de fournir jusqu'à 20 mA de courant, suffisant pour piloter un afficheur à LED. Les pattes sont en haute impédances quand une condition de reset devient active, ou bien lorsque l'horloge n'est pas active.

3 ALU-Arithmetic Logic Unit (Unité Arithmétique et Logique):

L'ALU fonctionne en directe avec les 32 registres à usage générales. Dans un seul cycle de base, des opérations arithmétiques entre ces registres ou entre un registre et une valeur immédiate sont exécutés. Les opérations de l'ALU sont classées dans trois catégories : arithmétique, logique et fonctions sur bit. Quelques implémentations de cette architecture offrent des multiplicateurs puissants supportant des multiplications signées/non-signées et le format fractionnel.

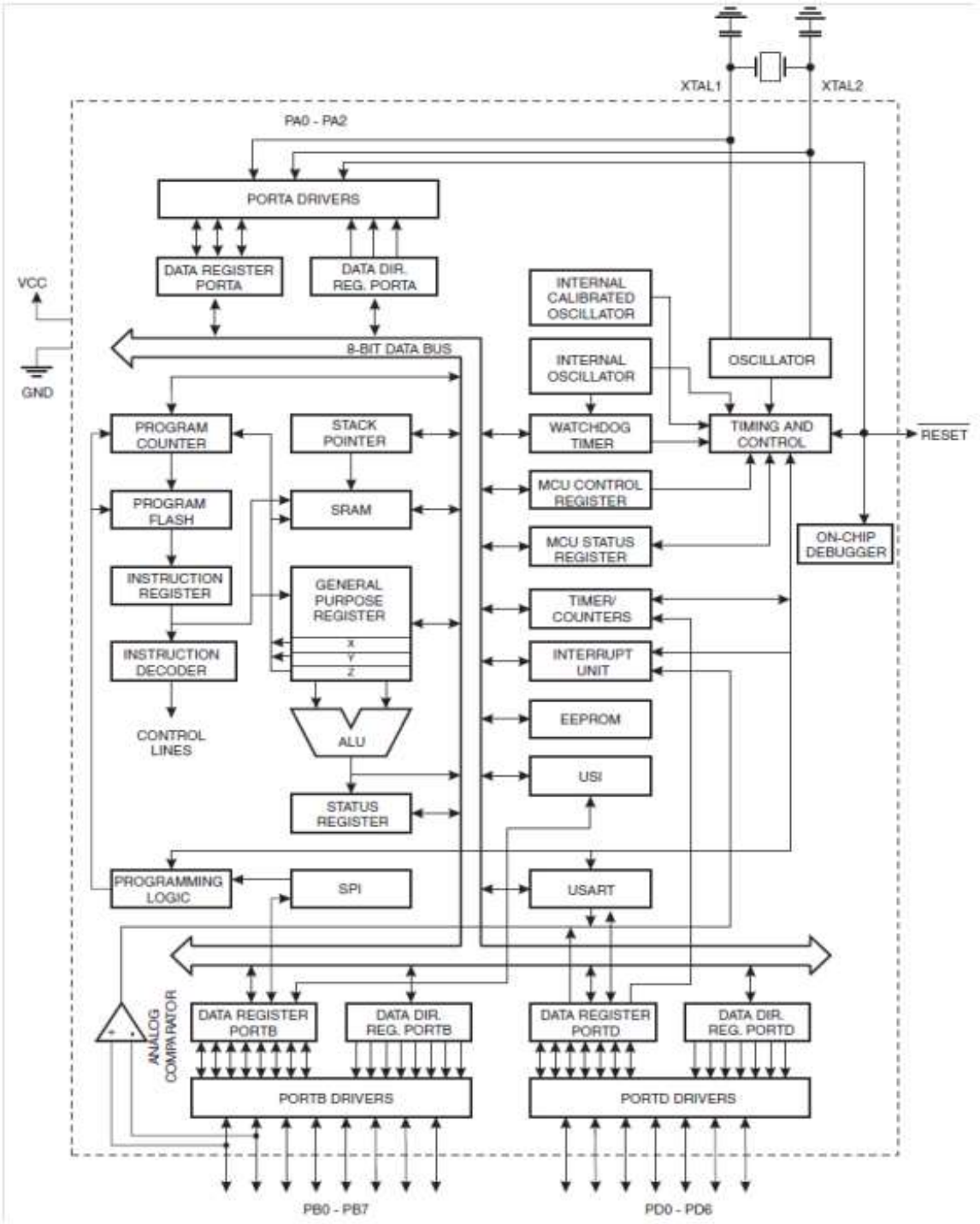


Figure 3. Les schéma bloc du microcontrôleur ATtiny 2313

4 Registre d'état (SREG):

Il est définie par:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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	

Bit 7- I : Global Interrupt Enable

Les interruptions sont permises si ce bit est mis à1. Si une interruption est en cours ce bit est automatiquement mis à zéro. L'instruction RETI le remet à 1 pour permettre une nouvelle interruption. Le bit I peut être mis à 1 ou à 0 par les instructions SEI et CLI respectivement.

Bit 6 – T : Bit Copy Storage

Ce bit est utilisé par les instruction BLD (Bit LoaD) et BST (Bit STore) comme bit de source et de destination, respectivement pour le bit opérande.

Bit 5 – H : Half Carry Flag

Le bit H indique la retenue partielle dans quelques opérations arithmétiques. Il est très utile dans les opérations BCD arithmétique.

Bit 4 – S : Sign bit**, $S = N \oplus V$**

Le bit de signe S est toujours le ou exclusive du drapeau N et celui dépassement du complément à 2 V.

Bit 3 – V : Two's Complement Overflow Flag – drapeau de dépassement du complement à 2

Bit 2 – N : Negative Flag

Le drapeau négatif N indique que le résultat de l'opération arithmétique ou logique est négatif.

Bit 1 – Z: Zero Flag

Le drapeau zéro Z indique que le résultat de l'opération arithmétique ou logique est nul

Bit 0 – C: Carry Flag

Le drapeau de la retenu C indique la retenu du résultat de l'opération arithmétique ou logique.

Notez bien :

- Le registre d'état change après chaque opération de l'ALU, voir la liste du jeu d'instructions.
- Lors de l'exécution d'une routine d'interruption le contenu du registre d'état ne se sauvegarde pas et ne se restaure pas automatiquement, il faut les faire par programme.

5 Registres généraux (General Purpose Working Registers)

Les 32x8-bit registres généraux forment la file de registres à accès rapide avec un temps d'accès égale à un seul cycle machine.

Six des 32 registres (R26-R31) peuvent être utilisés comme trois registres d'adressage indirect de 16-bit utilisés dans l'adressage de l'espace de données. Un de ces registres pointeurs d'adresse peut être utilisé pour pointer sur des tableaux dans la mémoire des programmes Flash. On les appels les registres X, Y, et Z.

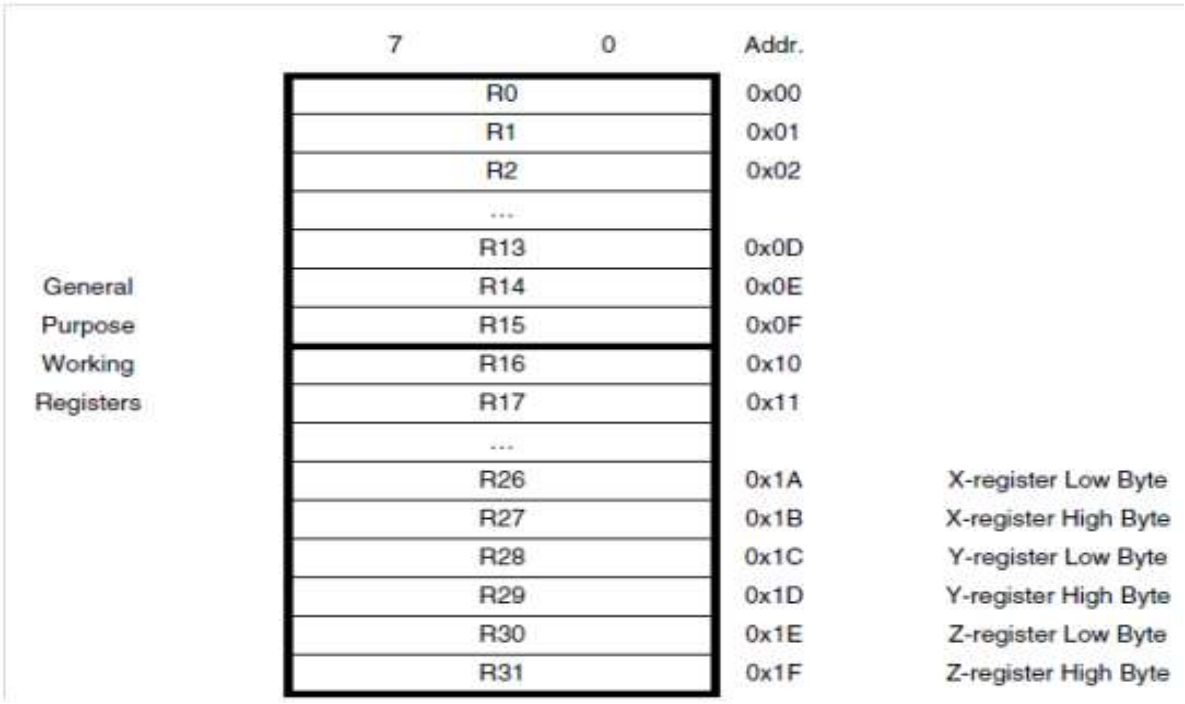


Figure 4: Les registres généraux

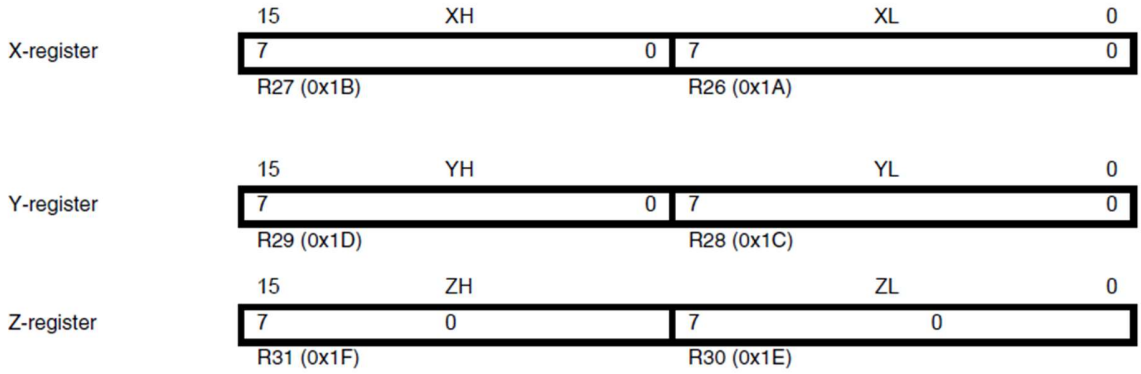


Figure 5: Les Registres X, Y et Z

6 Exécution des instructions:

L'exécution de chaque instruction passe par deux cycles : le cycle Fetch où l'instruction est obtenue de la mémoire des programmes et le cycle d'exécution où cette instruction est exécutée.

Dans le µc ATtiny 2313, les deux types de cycles s'effectuent simultanément, figure 5. Pendant qu'une instruction est entrain d'être exécuter, les octets composants la prochaine instruction sont en cours de recherche en mémoire de programme. Ce mode de fonctionnement, dit **pipeline** (chaîne de montage) à un seul niveau, permet aux instructions d'être exécuter en un seul cycle de base.

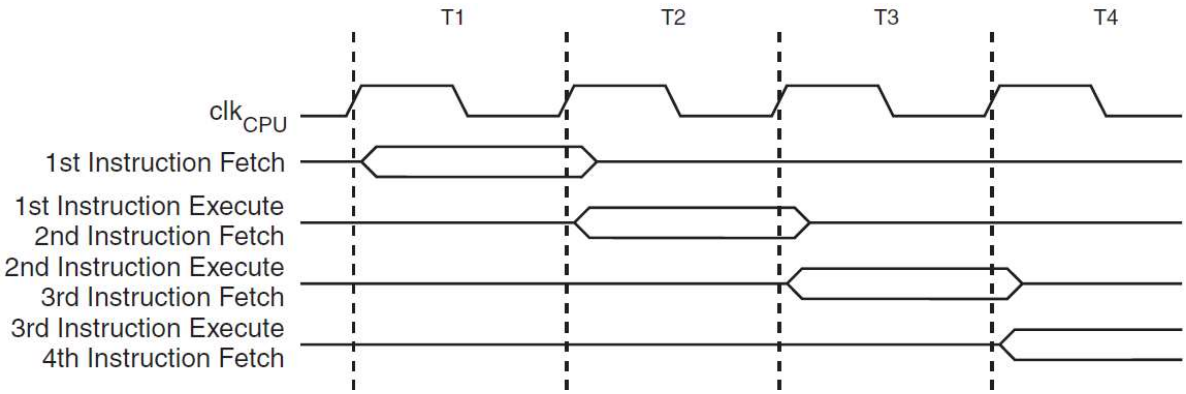


Figure 6: Digramme temporelle des cycles d'exécution d'une instruction

7 Les ports d'entrées/sorties (Input/Output port):

Un port est un point d'accès aux interfaces d'entrées/sorties.

Il y a trois ports dans le µc ATtiny 2313 : Port A (PA2..PA0) à 3bits, Port B (PA7..PA0) à 8bits, Port D (PD6..PD0) à 7 bits, ce sont des ports bidirectionnel

Pour la gestion de la communication avec les périphériques le µc utilise trois types de registres associés à chaque port :

- Registre de direction, noté DDRx (Data Direction Register), indique la direction de la donnée, ou mieux, la direction que peut prendre chaque bit de la donnée du port.
- Registre de données, noté PORTx, contient la donnée de sortie à envoyer.
- Registre des PINs d'entrée (Input PIN Adresse Register), noté PINx, sert à acquérir la donnée à entrer.

Où x indique le nom du port / x={A,B,D}.

Les bits de la données d'un port sont appelés les pins. Chaque pin est décrite par : DDRxn, PORTxn, et PINxn; où x indique le nom du port / x={A,B,D} et n indique le numéro du pin.

7.1 Configuration des pins:

La configuration se fait par le registre de direction DDRx. Le bit DDRxn du registre DDRx sélectionne la direction du PIN n:

- Si DDRxn =1 Alors PINxn est configuré en sortie;
- Si DDRxn=0 Alors PINxn est configuré en entrée

On donne le schéma simplifié correspondant à chaque pin d'entrée-sortie, illustré pour le pin 2 du port B (PB2):

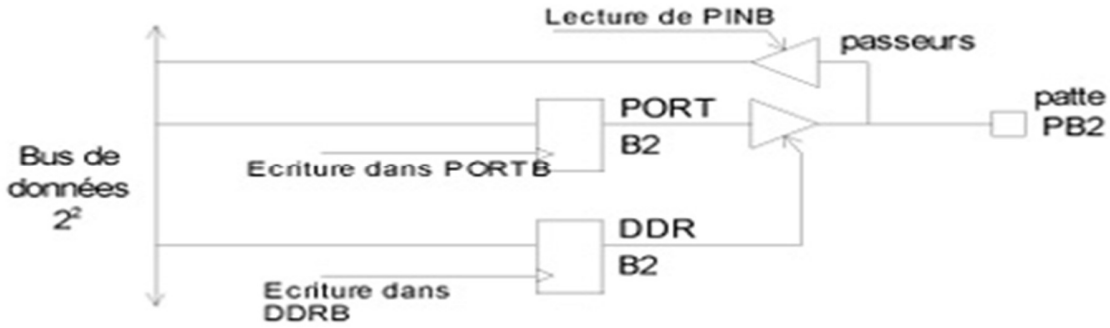


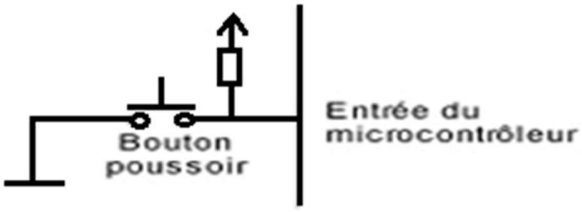
Figure 7: Schéma simplifié d'un pin d'entrée-sortie

On y trouve deux passeurs. L'un permet à tout instant de lire l'état du pin en question. L'autre permet d'imposer une valeur logique au pin, lorsqu'il est configuré en sortie.

On y trouve aussi deux bascules. Chacune fait partie d'un registre 8 bit. L'une de ces bascules est le registre DDR. L'autre est le registre PORT.

7.2 Utilisation des interrupteurs et boutons poussoirs

Le schéma généralement utilisé pour lire la valeur d'un interrupteur ou d'un bouton-poussoir est le suivant :



Lorsque le bouton-poussoir est pressé, l'entrée du µcontrôleur reçoit la valeur ' 0'. Lorsqu'il est ouvert, une résistance est nécessaire pour qu'une valeur « 1 » soit transmise à l'entrée. On l'appelle **résistance de rappel**, ou **pull-up resistor**.

Sans cette résistance, l'entrée serait en l'air (=libre =non connectée=non reliée). Or les entrée en circuit fabriqués en technologie C-MOS ont une impédance très élevée, leur état est donc indéterminé lorsqu'elles ne sont pas reliées. On observe facilement dans la pratique qu'une entrée en l'air change de valeur à chaque instant, sous l'influence des perturbations électromagnétiques ambiantes.

Avec ce schéma, la valeur lue sur l'entrée sera 0 lorsque le bouton est pressé (donc actif) et 1 lorsque le bouton est relâché.

Le fabricant de la famille des µcontrôleurs AVR à intégré une résistance de rappel sur le circuit intégré pour imposer la valeur logique '1' à l'entrée lorsque celle-ci est libre. Ainsi, un bouton poussoir ou une interrupteur peut être connecté sans besoin de résistance externe.

La fonctionnalité correspondant à l'état des bits des registre DDR et PORT est donnée sur le tableau suivant :

DDRXn	PORTXn	Configuration du pin_n du port X
0	0	Entrée, Haute impédance
0	1	Entrée, avec résistance de rappel
1	0	Sortie, état 0
1	1	Sortie, état 1

Exemple:

On veut mettre les pins 0 et 1 du port B au niveau haut, les pins 2 et 3 du port B au niveau bas, et configurer les pins 4 à 7 en entrée.

Si le pin 5 est au niveau haut mettre une variable x à 33 sinon à 0

Le programme en code C est le suivant :

```
#include<tiny2313.h>
Int x;
Void main (void)
{
        DDRB=0x……..;
        While(1)
        {       PORTB=0x……..;
                If (PINB5==1)   x=33;
                Else x=0;
        }
}
```

8 Les modes de basse consommation (de sauvegarde de puissance)

Mode ***Idle*** arête le CPU pendant que le SRAM, Timer/compteurs, port SPI, et le système d'interruption continuent à fonctionner.

Le mode ***power-down*** sauvegarde le contenu des registres mais stop l'oscillateur, inhibant de la sorte le fonctionnement du reste des éléments jusqu'à l'arrivé de la prochaine interruption externe ou une RESET matérielle.

Dans le mode ***Standby***, l'oscillateur Crystal/résonateur est fonctionnel alors que les autres éléments ne le sont pas. Cela permet un démarrage très rapide avec une consommation de puissance minimal

9 Mémoire de programme

Le µc ATtiny 2313 contient une mémoire In-System programmable Flash (ISP Flash) de 2 K octets pour stocker les programmes. Comme les instructions de l'AVR sont de largeur 16 ou 32 bits, la mémoire Flash est organisée en 1Kx16.

Ce type de mémoire peut être reprogrammé dans la puce à travers un SPI « Serial peripheral interface » ou par un programmeur de mémoire non-volatile conventionnel.

10 Mémoire de donnée

10.1 La mémoire de donnée interne SARM

Elle peut être décomposée en deux parties significatives : L'une contient les données et l'autre est destinée aux registres.

L'espace adressable par le µc pour les données est illustré dans la figure 7.

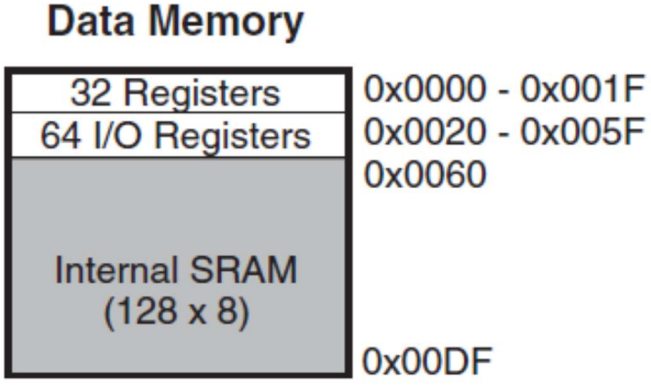


Figure 8. Cartographie de la mémoire de données

Espace de 0000-001F est réservé pour les 32 registres à usage général. L'espace qui suit est réservé aux 64 registres standards d'I/O décrits dans le tableau « registre summary ». Les 128 derniers emplacements mémoires sont réservés pour la mémoire SRAM interne.

10.2 La mémoire EEPROM

Le µC ATtiny 2313 possède une mémoire de donnée type EEPROM de capacité 128 Octets. Elle est organisée dans un espace de données séparé ou un seul octet peut être lu ou écrit à chaque fois. L'EEPROM a une endurance de 100,000cycle d'écriture/effacement.

La mémoire EEPROM diffère de la mémoire SRAM dans la conservation de l'information après la coupure de l'alimentation, et en plus on peut écrire et lire dans l'EEPROM lors de l'exécution du programme.

Il y a trois registres associés à l'accès de l'EEPROM :

- EEAR (EEPROM Address Register) : registre de l'adresse contient l'adresse de la donnée à lire ou à écrire.
- EEDR (EEPROM Data Register) : registre de donnée contient la donnée à écrire en cas d'écriture ou la donnée lue en cas de lecture.
- EECR (EEPROM Control Registre) : registre de contrôle pour contrôler l'accès à la mémoire EEPROM.

Ces registres sont accessibles dans l'espace d'I/O.

Lors de la lecture dans L'EEPROM le CPU s'arrête pendant 4 cycles horloge avant que l'instruction suivante ne s'exécute. Lors de l'écriture, le CPU s'arrête pendant 2 cycles horloge avant que l'instruction suivante ne s'exécute.

10.2.1 Description du registre de contrôle de l'EEPROM (EECR)

Bit	7	6	5	4	3	2	1	0	
	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

Figure 9. Registre de contrôle de l'EEPROM.

.

Bit 5-4 : EEPM1, EEPM0 (EEPROM Programmation Mode bits)

Les bits de mode de programmation définis quelle opération de programmation à entreprendre lorsque le bit EEPE est écrit. Ces bits sont ignorés si le bit EEPE est mis à 1.

EEPM1	EEPM0	Temps de programmation	Opération
0	0	3.4 ms	Effacer et écrire en une seule opération
0	1	1.8 ms	Effacer seulement
1	0	1.8 ms	Écrire seulement
1	1	-	Réserver pour une utilisation ultérieur

Bit 3 : EERIE (EEPROM Ready Interrupt Enable)

- Si le bit de validation de l'interruption EE_RDY (EERIE) est mise à 1, cette interruption est validé si le bit I du registre d'état SREG est à 1.
- Sinon (EERIE=0) l'interruption est bloquée.

Sachant que l'interruption EE_RDY génère une interruption constante lorsque l'EEPROM est prête pour la programmation (c-à-d à la fin de chaque opération programmation).

Bit 2 : EEMPE (EEPROM Master Programming Enable)

Le bit maître de l'autorisation de la programmation de l'EEPROM détermine si la mise à 1 de l'EEPE a un effet ou non.

- Si EEMPE=1, la mise à 1 de EEPE pendant 4 cycle horloge va programmer l'EEPROM à l'adresse sélectionnée.
- Sinon (EEMPE=0) la mise à 1 de EEPE n'a aucun effet.

Lorsque EEMPE est mise à 1 par le programme, le matériel (hardware) efface le bit et le met à 0 après 4 cycles horloge.

Bit 1 : EEPE (EEPROM Programming Enable)

Lorsque le bit d'autorisation de programmation est écrit, l'EEPROM va être programmée selon les bits EEPMn. La mise à 1 du bit EEMPE doit être effectuée avant la mise à 1 du bit EEPE, sinon aucune programmation n'aura lieu. Lorsque le temps de programmation (d'écriture) est écoulé le bit EEPE est mis à 0 par le hardware. Et lorsque le bit EEPE est mis à 1, le CPU est arrêté pour 2 cycles horloge avant que la prochaine instruction ne s'exécute.

Bit 0 : EERE (EEPROM Read Enable)

Le bit d'autorisation de lecture l'EEPROM est une sorte de demande de lecture. Lorsque l'adresse est placée dans le registre EEAR, le bit EERE doit être mis à 1 pour que la lecture aura lieu.

L'opération de lecture de l'EEPROM s'effectue en un seul cycle horloge, et elle ne peut pas s'effectuer si une opération d'écriture est en cours et on ne peut pas changer le contenu du registre EEAR. Donc on doit vérifier la valeur du bit EEPE qui doit être =0 avant que la prochaine instruction ne s'exécute.

10.2.2 Procédure de lecture/écriture dans l'EEPROM

Bien que l'utilisation de l'EEPROM à l'aide des registres reste simple, une procédure particulière est effectuée avant toute demande d'écriture dans cette mémoire, dans le but de sécuriser au maximum le bon fonctionnement de celle-ci.

10.2.2.1 Procédure d'écriture

- Test du bit EEPE = 0afin de savoir si une procédure est en cours.
- Ecrire l'adresse où l'on souhaite écrire la donnée dans les registres EEAR.
- Ecrire la donnée dans le registre EEDR.
- Mettre à 1 le bit EEMPE.
- Mettre à1 le bit EEPE dans un délai inférieur à quatre cycles d'horloge sous peine d'annuler l'écriture.

10.2.2.2 Procédure de lecture

- Test du bit EEPE = 0 afin de savoir si une procédure est en cours.

- Ecrire l'adresse où l'on souhaite accéder à la donnée dans les registres EEAR.
- Mettre à 1 le bit EERE.
- Lecture de la donnée dans le registre EEDR.

10.2.2.3 Exemples de procédure en C pour la lecture et l'écriture dans l'EEPROM

```
Void EE_READ (Var ,Adr)      //Lire une donnée dans l'EEPROM d'adresse Adr et la mettre dans une variable Var
{
        While (EECR & 0x02==0x02) ;//vérifier si EEPROM est prête
        EEAR =Adr ;   //mettre l'adresse dans le registre EEAR
        EECR |= 0x01 ;          //mettre à 1 le bit EERE
        Var=EEDR ;   //Lire la donnée en la mettant dans la variable Var
}

Void EE_Write(Adr, Val)        //Ecrire une donnée val dans l'EEPROM à l'adresse Adr
{
        while (EECR & 0x02==0x02) ;
        EEAR=Adr;
        EEDR=Val;
        EECR|=0x04;
        EECR|=0x02;
}
```

Chapitre 6 : Timers dans le Microcontrôleur ATtiny2313

1 Généralités sur le Timer/counter

Le timer/counter (temporisateur/compteur) ou tous court un 'timer' est un périphérique matériel important dans le microcontrôleur **fonctionnant indépendamment de l'exécution du programme**. Il peut fonctionner soit comme :

- un temporisateur (timer) : mesure le temps entre deux actions ou évènements et génère des impulsions. Un signal d'horloge interne (clk_I/O) ou dérivé de ce signal est utilisé pour synchroniser le timer.
-
- un compteur (counter) : compte le nombre d'évènements sur une broche (pin). Un signal externe provenant d'un pin d'un port est utilisé pour le synchroniser.

2 Les timer/counter dans le µC ATtiny 2313

Le µ-controleur ATtiny 2313 possède deux timers avec des résolutions différentes : **Timer/conter0 de 8 bits** et **Timer/conter1 de 16 bits**. La résolution du timer indique sa capacité de comptage, par exemple le timer 8 bits peut compter jusqu'à 2^8=256 avant de passer à 0.

3 Circuit de sélection du signal horloge

Le circuit logique qui permet la sélection du signal horloge pour le timer/counter0 (clk_{T0}) et celui du timer/counter1 (clk_{T1}) est présenté sur la figure 1.

La source du signal horloge est sélectionnée par les trois bits CS00, CS01, CS02 du registre de control du timer/counter0 (TCCR0B) selon le tableau 1. La même logique de sélection est valable pour le timer/counter1 mais en utilisant les bits CS10, CS11, CS12 du registre de control du timer/counter1.

Tableau 1

CS02	CS01	CS00	Description
0	0	0	Arrêter le timer/counter0
0	0	1	clk_{T0} =CK (Pas de division)
0	1	0	clk_{T0} =CK/8 (Division par 8)
0	1	1	clk_{T0} =CK /64 (Division par 64)
1	0	0	clk_{T0} =CK /256 (Division par 256)
1	0	1	clk_{T0} =CK /1024 (Division par 1024)
1	1	0	clk_{T0}=T0 (Le timer0 est synchroniser sur front descendant)
1	1	1	clk_{T0}=T0 (Le timer0 est synchroniser sur front montant)

Timer/counter0 à 8 bits:

Ce timer est caractérisé par :

- Deux unités indépendantes de sortie de comparaison (Output Compare Units) ;
- Deux registres tampons de sortie de comparaison (Double Buffered Output Compare Registers) ;
- Remise à zéro du timer sur Compare Match ;
- Phase Correct Pulse Width Modulator (PWM);
- Période de PWM variable ;
- Générateur de fréquence ;
- Trois sources d'interruptions indépendantes (TOV0, OCF0A, and OCF0B).

Le schéma block simplifié du timer/counter0 est illustré sur la figure 2. Sur la figure 3, on décrit les registres du timer/counter0.

4 Les registres du timer/counter0

4.1 Le registre TCNT0

Valeur de TCNT0 est accessible en lecture ou écriture en permanence. mais si le mode comparaison est actif, le contrôle peut être passé et entraîner une erreur dans votre programme, le compteur ne se bloquera pas.

4.2 Les registres OCR0A et OCR0B

Ces registres sont les registres de sortie de comparaison (Output Compare Registers). Leurs contenues sont constamment comparées avec celui du TCNT0. Le résultat de comparaison est utilisé par le générateur de forme d'onde (Waveform Generator to generate) pour générer un signal PWM ou une sortie à fréquence variable sur les pins OC0A and OC0B (Output Compare pins).

4.3 Les registres TCCR0A, TCCR0B

TCCR0A et TCCR0B (Timer/Counter0 Control Registers) sont les registres de contrôle du timer/counter0.

- Les bits CS02, CS01 et CS00 (les bits 2-0 du registre de control TCCR0B) permettent la sélection de la source du signal horloge clkT0, voir tableau 1.
- Les bits WGM02, WGM01 et WGM00 (Le bit 3 du registre de control TCCR0B et les bits 1-0 du registre de control TCCR0A, respectivement) sont les bits qui permettent de déterminer le mode d'opération du timer (c-à-d, la séquence de comptage, la source de la valeur TOP du timer et le type de génération de forme d'onde), tableau 2.

Les modes d'opération que peut supporter le timer/counter0 sont :

- *Le mode normal ;*
- *Le mode CTC (Clear Timer on Compare Match) ;*
- *Le mode Phase correct PWM (pulse width modulation- modulation de largeur d'impultion)*
- *Le mode fast PWM*

- Les bits COM0A1-0 (les bits 7-6 du registre TCCR0A) et les bits COM0B1-0 (les bits 5-4 du registre TCCR0A) sont les bits qui contrôlent le comportement du pin de sortie de comparaison (OCA) et (OCB), respectivement, tableau 3 -4.
- Les bits FOC0A et FOC0B (les bits 7-6 du registre de contrôle TCCR0B) sont actifs seulement si les bits WGM02-0 spécifient un mode non-PWM (mode normale ou mode CTC). Dans les modes PWM, ils doivent être mis à zéro. La mise à 1 de ces bits entraine le forçage d'une comparaison immédiate dans l'unité de génération de forme d'ondes. Cela veut dire que la comparaison se fait normalement selon les valeurs des bits COM0A1-0 et COM0B1-0 mais sans la génération d'aucune d'interruption et sans remise à zéro du TCNT sur compare match en mode CTC. Ces bits sont est toujours lu comme zéro.

4.4 Le registre TIFR

Le registre **TIFR** (Timer Interrupt Flag Register) indique l'état des interruptions internes du Timer0 et du Comparateur.

- Le bit OCF0A Output Compare Flag (respectivement OCF0B) est mis à 1 quand une égalité se produise entre le contenu du registre TCNT0 et celui du registre OCR0A (respectivement OCR0B).
- Le bit TOV0 Timer/Counter0 Overflow Flag est mis à 1 quand le Timer/Counter0 dépasse une certaine valeur. Cette valeur peut être la valeur MAX ou BOTTOM ou bien la valeur stockée dans le registre OCR0x, cela dépend des bits WGM02-1, tableau 2.

Ces bits sont remis à 0 par le matériel lors de l'exécution de l'interruption. Le programmeur peut aussi les mettre à 0.

4.5 Le registre TIMSK

Le registre TIMSK (Timer Interrupt Mask) contient trois bits de masquage d'interruptions associés au timer/counter0 : **OCIE0A, OCIE0B et TOIE0**.

- Lorsque le bit OCIE0A-Output Compare Interrupt Enable- (respectivement OCIE0B) et le bit I du registre d'état sont mis à 1 l'interruption de comparaison A (respectivement B) du timer/counter0 est validée.
- Lorsque le bit TOIE0 –Timer/counter0 Overflow Interrupt Enable - et le bit I du registre d'état sont mis à 1 l'interruption de débordement du timer/counter0 est validée.

4.6 Unité de sortie de comparaison

Le schéma synoptique de l'unité de sortie de comparaison est donné sur la figure 4.
Le générateur de forme d'onde emploi les bits COM0A1-0 (respectivement COM0B1-0) pour définir l'état du registre tampon OC0A (respectivement OC0B).
La direction des pins PB2/OC0A et PD5/OC0B est toujours contrôlée par le registre de direction des données (DDRx). Pour sélectionner la sortie de comparaison OC0x provenant du générateur de forme d'onde il faut que l'un des bits COM01-0 soit mis à 1. Avant que la valeur de OC0x soit disponible sur le pin, il faut configurer ce pin en sortie en mettant un 1 dans le bit correspondant du registre de direction de données.
Les registres OCR0x sont à double tampon ('double buffered') seulement en modes PWM. Dans les modes non-PWM la caractéristique du 'double buffering' est désactiver. Le 'double buffering' synchronise la mise à jour du registre OCR0x ce qui prévient d'obtenir des impulsions PWM de longueur impaire et non-symétrique. Ce fonctionnement est très simple : lorsque le 'double buffering' est validé le CPU a accès au registre tampon de OCR0x et dans le cas contraire le CPU a accès directement au registre OCR0x.

5 Le fonctionnement du Timer/Compteur0

Le compteur TCNT0 est le cœur du timer. Son signal d'horloge clkT0 peut être l'horloge interne, via le pré diviseur (prescalar), une horloge externe sur la broche T0 (PD4) ou inactif quand aucune source d'horloge n'est choisie.

La valeur de ce registre est incrémentée et comparée en permanence à celle du registre OCR0A (respectivement OCR0B). Le résultat de comparaison est utilisé par le générateur de forme d'onde pour produire une sortie PWM ou une sortie à fréquence variable sur le pin OC0A (respectivement OC0B).

5.1 Les modes d'opération du timer/counter0

Le mode d'opération est le comportement du timer0 et les pins de sortie de comparaison. Il est défini par les bits de mode de génération de forme d'onde WGM02-0 et les bits de mode de sortie de comparaison COM0x1-0. Les bits WGM02-0 affectent la séquence de comptage contrairement aux COM0x1-0. L'inversion ou non de la sortie PWM générée est contrôlé par les bits COM0x1-0. Pour les non-PWM modes ces bits contrôlent si la sortie doit être mise à 1, mise à 0, ou inverser en cas d'égalité (compare match).

Les modes d'opération du timer/counter0 sont :

- Le mode normal ;
- Le mode CTC (Clear Timer on Compare Match) ;
- Le mode Phase correct PWM (pulse width modulation- modulation de largeur d'impultion)
- Le mode fast PWM

6 Programmation du timer/counter0 en mode normal

Le mode normal est le mode le plus simple (WGM01=0 et WGM00=0) où le registre TCNT0 s'incrémente à chaque coup d'horloge jusqu'à sa valeur maximale (Fixed TOP Value=0xFF), puis redémarre à partir du BOTTOM=0x00. Le drapeau de dépassement (TOV0) est mis à 1 dans le même cycle horloge de redémarrage du registre TCNT0. Il peut être lu par le programme (pour générer une interruption) et

être remis à 0. La remise à 0 de ce bit se fait en écrivant un 1. A tous moment, une nouvelle valeur peut être écrite dans le registre TCNT0, figure 5. Dans le mode normal, l'unité de sortie de comparaison peut être utilisée pour générer des interruptions à des moments données. La génération des signaux en ce mode est à éviter car elle occupe beaucoup du temps du CPU. Car on ne peut pas forcer le timer à zéro en cas d'égalité entre les registres TCNT0 et OCR0x.

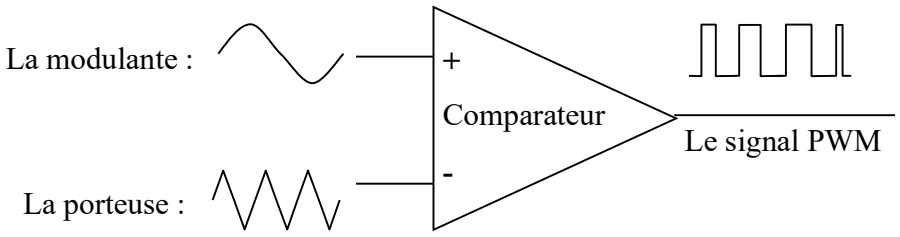
7 Programmation du timer/counter0 en mode CTC (Clear Timer on Compare match)

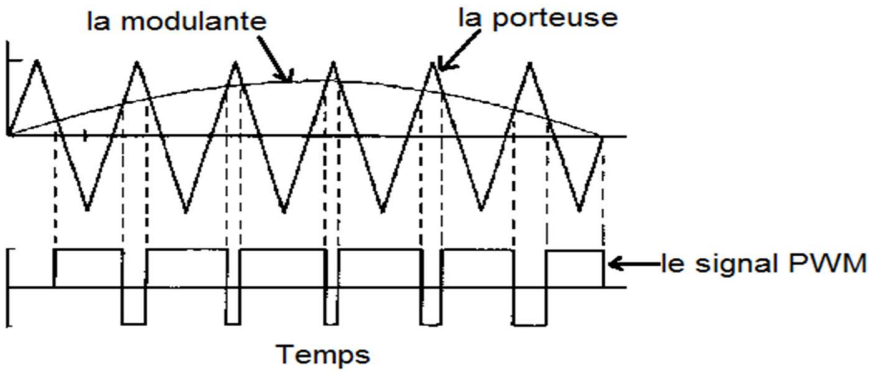
Dans le mode CTC (WGM02-0=2), le registre TCNT0 est mis à zéro lorsque sa valeur est égale à la valeur du registre OCR0x. Ce mode permet un bon contrôle de la fréquence de sortie de comparaison. Il simplifie aussi les opérations de comptage des événements externes. Le schéma bloc du timer/counter0 en mode CTC et son diagramme temporel sont présentés sur la figure 9 et figure 10, respectivement.

Une interruption peut être générer à chaque fois la valeur de TCNT0 atteint la valeur TOP en utilisant le drapeau OCF0x. Si l'interruption est validée, la routine de service peut être utilisée pour changer la valeur TOP. Cependant, le changement de TOP à une valeur proche du BOTTOM lors du comptage du TCNT0 avec un petit facteur de division ou sans division doit être fait avec précaution, car il n y a pas de caractéristique de 'double buffering' dans ce mode. Si une nouvelle valeur écrite dans le registre OCR0x est inférieur à la valeur courante de TCNT0, le compteur va rater le 'Compare Match' et continu son comptage jusqu'à la valeur maximale (0xFF) puis revient à zéro avant que le 'Compare Match' peut se produire.

8 La modulation de largeur d'impulsion (Pulse Width Modulation -PWM)

Elle consiste à comparer la modulante (le signal à moduler) à une porteuse généralement triangulaire. Le signal de sortie vaut 1 si la modulante est plus grande que la porteuse, sinon il vaut 0, voir les deux figures ci-dessous.





La modulation de largeur d'impulsion est une technique très utilisée dans le domaine de télécommunication et le contrôle de puissance (On peut par exemple, contrôler la luminosité des LEDs, mélanger les couleurs en utilisant des LEDs RGB, contrôler la vitesse d'un moteur DC, générer des signaux audio, générer un signal modulé pour par exemple commander une LED infra-rouge d'une télécommande). Elle est principalement utilisée pour contrôler la quantité de l'énergie à fournir au dispositif électrique par fonctionnement tout ou rien. Ainsi, la valeur moyenne de l'énergie reçue par le dispositif électrique dépond du rapport cyclique.

Le **rapport cyclique** pour un signal carré, noté α, est défini comme étant le rapport entre la durée niveau haut (t_H) du signal et sa période (T).

$$\alpha = \frac{t_H}{T}$$

Il varie de 0 à 1, et en pourcentage de 0 % à 100 %.

9 Programmation du timer/counter0 en mode fast PWM

Le mode fast PWM (WGM02:0 = 3 or 7) permet la génération de signaux de type PWM de haute fréquences sur la sortie OC0x.

Dans ce mode, le compteur TCNT0 compte du BOTTOM jusqu'à TOP puis recommence à partir du BOTTOM.

TOP = { Max si WGM02=0 ; OCR0x si WGM02=1

Avant de passer de la valeur TOP au BOTTOM, l'indicateur TOV0 est mise à 1. Si l'interruption associée est validée, sa routine peut être utilisée pour changer la valeur du registre OCR0x.

Au cours du comptage du BOTTOM au TOP la comparaison entre TCNT0 et OCR0x est effectuée. Si une égalité est trouvée, le pin OC0x est programmé par la valeur de COM0x1 :0 pour se mettre à 1 ou à 0. Plus précisément :

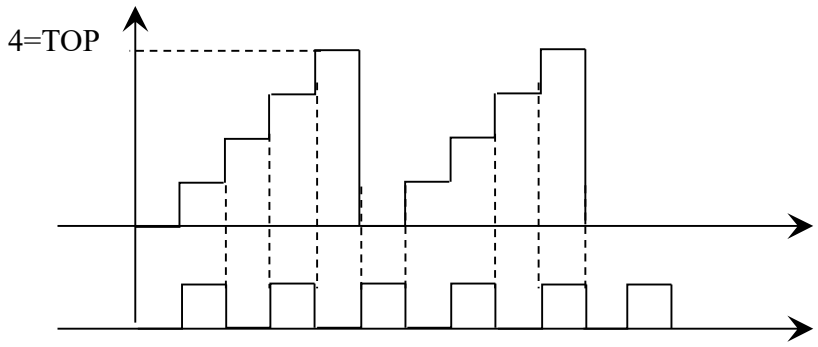
- Si COM0x1 :0=1 et si WGM02=1, la sortie OC0A est inversé en cas d'égalité entre TCNT0 et OCR0A. Cette fonctionnalité n'est pas disponible pour OC0B.
- Si COM0x1 :0=2, le mode de sortie de comparaison est sans inversement (mode non-inverted fast PWM) où la sortie OC0x est :
  - mise à 0(clear) si TCNT0=OCR0x
  - mise à 1(set) si TCNT0=TOP
- Si COM0x1 :0=3, le mode de sortie de comparaison est avec inversement (mode inverted fast PWM) où la sortie OC0x est :
  - mise à 1(set) si TCNT0=OCR0x
  - mise à 0(clear) si TCNT0= TOP

Remarque 1 :

Il faut noter que les valeurs extrêmes du registre OCR0x représentent des cas spéciaux dans la génération du signal PWM.

- Si OCR0x =BOTTOM, le signal à la sortie OC0x est formé d'un pic étroit dans chaque (MAX+1) cycle horloge du timer0.

- Si OCR0x = MAX, le signal à la sortie OC0x sera constant de niveau haut dans le mode non-inverted fast PWM et constant de niveau bas dans le mode inverted fast PWM.



9.1 Fast PWM Mode avec TOP= MAX=0xFF (WGM02=0)

Le schéma bloc du timer/counter0 en Fast PWM mode et son diagramme temporel sont présentés sur la figure 11 et figure 12, respectivement.

La fréquence du signal généré sur OCR0x est obtenu par :

$$f_{OCR0xPWM} = \frac{f_{T0}}{256} = \frac{\frac{f_{CLK}}{N}}{256}$$

$$f_{OCR0xPWM} = \frac{f_{CLK}}{N.256}$$

N représente le facteur de division (1, 8, 64, 256 ou 1024).

Le rapport cyclique est déterminé par la valeur à mettre dans le registre OCR0x. Cette valeur varie de 0 à 255 et peut être obtenue a partir de la formule suivante :

Rapport cyclique=$\frac{OCR0x}{256} \cdot 100$ mode Non-Inversé

Rapport cyclique =$\frac{256-OCR0x}{256} \cdot 100$ mode Inversé

Exemple : en mode Non inversé

Rapport cyclique (%)	OCR0x	
	En décimale	En hexadécimale
30	76	0x4C
64	163	0xA3
80	2044	0xCC

Remarque 2 :

- Le Timer0 peut générer des signaux PWM sur deux pins différents. Ces signaux possèdent la même fréquence mais diffèrent dans le rapport cyclique.
- La génération d'un signal PWM dans ce mode se fait en deux étapes :
  - Programmation du Timer0 pour générer un signal de fréquence spécifiée.
  - Détermination de la valeur fixe à mettre dans le registre de comparaison OCR0x qui permet de définir le rapport cyclique.

9.2 Fast PWM Mode avec TOP=OCR0x (WGM02=1)

Ce mode est utilisé pour avoir plus de contrôle sur la fréquence du signal PWM généré que le mode précédent.

Dans ce mode, le compteur TCNT0 compte du BOTTOM jusqu'à TOP=**OCR0A** puis recommence à partir du BOTTOM. Selon le cas spécial (OCR0x=TOP) qu'on a cité dans la remarque 1, on a

Si COM0A1 :0=2, le signal généré sur OC0A est constant de niveau haut (ce n'est pas signal PWM).

Si COM0A1 :0=3, le signal généré sur OC0A est constant de niveau bas (ce n'est pas signal PWM).

La seule combinaison utilisable est :

COM0A1 :0=1 où le signal généré sur OC0A s'inverse chaque cycle horloge du timer0.

Dans ce cas la fréquence du signal généré sur OC0A est :

$$f_{OC0A} = \frac{f_{clk\_IO}}{2.N.(1 + OCR0A)}$$

Et le rapport cyclique est toujours fixé à 50%

Cela s'explique par le fait que la valeur du registre OCR0A ne peut pas être utilisée comme la valeur maximale de comptage et la valeur de comparaison à la fois.

En fait, c'est le canal B qui est utilisé pour générer un signal PWM avec une fréquence et un rapport cyclique contrôlable.

La fréquence du signal est donnée par :

$$f_{OC0B} = \frac{f_{clk\_IO}}{N.(1 + OCR0A)}$$

Et le rapport cyclique est donné par :

Rapport cyclique$=\frac{OCR0B}{OCR0A} \cdot 100$ mode Non-Inversé

Rapport cyclique $=\frac{(OCR0A+1)-OCR0B}{OCR0A} \cdot 100$ mode Inversé

La génération d'un signal PWM dans ce mode se fait en 2 étapes :

- Programmation du Timer0 pour générer un signal de fréquence spécifiée en même temps que la détermination de la valeur fixe à mettre dans le registre de comparaison OCR0A.
- Détermination de la valeur fixe à mettre dans le registre de comparaison OCR0B qui permet de définir le rapport cyclique.

10 Programmation du timer/counter0 en mode Phase Correct PWM:

Dans le mode phase correct PMW (WGM02:0 = 1 or 5), le compteur commence par compter de BOTTOM à TOP puis décompte de TOP jusqu'à BOTTOM. Notez bien que le compteur est = à TOP juste pendant un seul cycle horloge du timer0.

$$TOP = \begin{cases} MAX & \text{si WGM02 :0=1} \\ OCR0x & \text{si WGM02 :0=5} \end{cases}$$

Quand le compteur atteint BOTTOM, l'indicateur TOV0 est mise à 1.

Au cours du comptage et du décomptage la comparaison entre TCNT0 et OCR0x est effectuée.

Si une égalité est trouvée, le pin OC0x est programmé par la valeur de COM0x1 :0 pour se mettre à 1 ou à 0. Plus précisément :

1. Si COM0A1 :0=1 et si WGM02=1, la sortie OC0A est inversé en cas d'égalité entre TCNT0 et OCR0A. Cette fonctionnalité n'est pas disponible pour OC0B.
2. Si COM0x1 :0=2, le mode de sortie de comparaison est sans inversement (mode non-inverted fast PWM) où la sortie OC0x est :
   - mise à 0(clear) si TCNT0=OCR0x lors du comptage
   - mise à 1(set) si TCNT0=OCR0x lors du décomptage
3. Si COM0x1 :0=3, le mode de sortie de comparaison est avec inversement (mode inverted fast PWM) où la sortie OC0x est :
   - mise à 1(set) si TCNT0=OCR0x lors du comptage
   - mise à 0(clear) si TCNT0=OCR0x lors du décomptage

Remarque 3 :

1) Il faut noter que les valeurs extrêmes du registre OCR0x représentent des cas spéciaux dans la génération du signal PWM.
   - Si OCR0x = BOTTOM, le signal à la sortie OC0x est constamment au niveau bas.
   - Si OCR0x = MAX, le signal à la sortie OC0x sera constant de niveau haut dans le mode non-inverted PWM et constant de niveau bas dans le mode inverted PWM.

2) On peut avoir des transitions sans qu'une égalité ne se soit produite. Il ya deux cas :
   - Lorsque OCR0x=MAX et si la valeur de OCR0x change au moment où TCNT0= MAX, figure 13. La sortie OC0x est programmée selon le cas d'une égalité lors de décomptage.

- Lorsque le timer0 commence le comptage à partir d'une valeur supérieur à la valeur de registre OCR0x. le signal à la sortie OC0x est programmée selon ce qui a due ce passé.

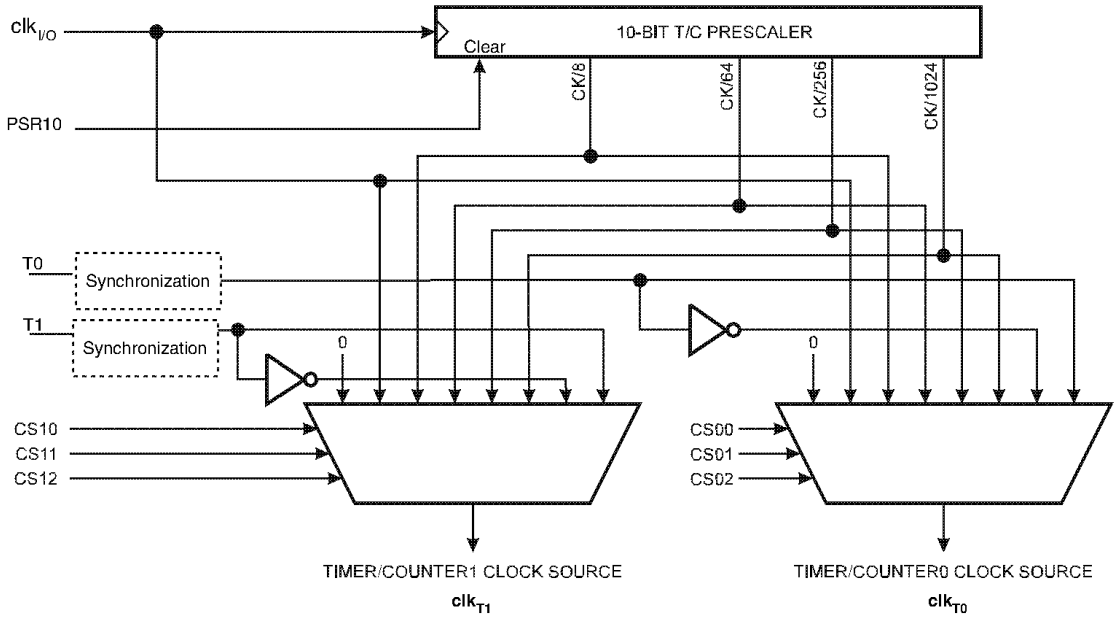


Figure 1. Le circuit de sélection du signal horloge pour les deux timers

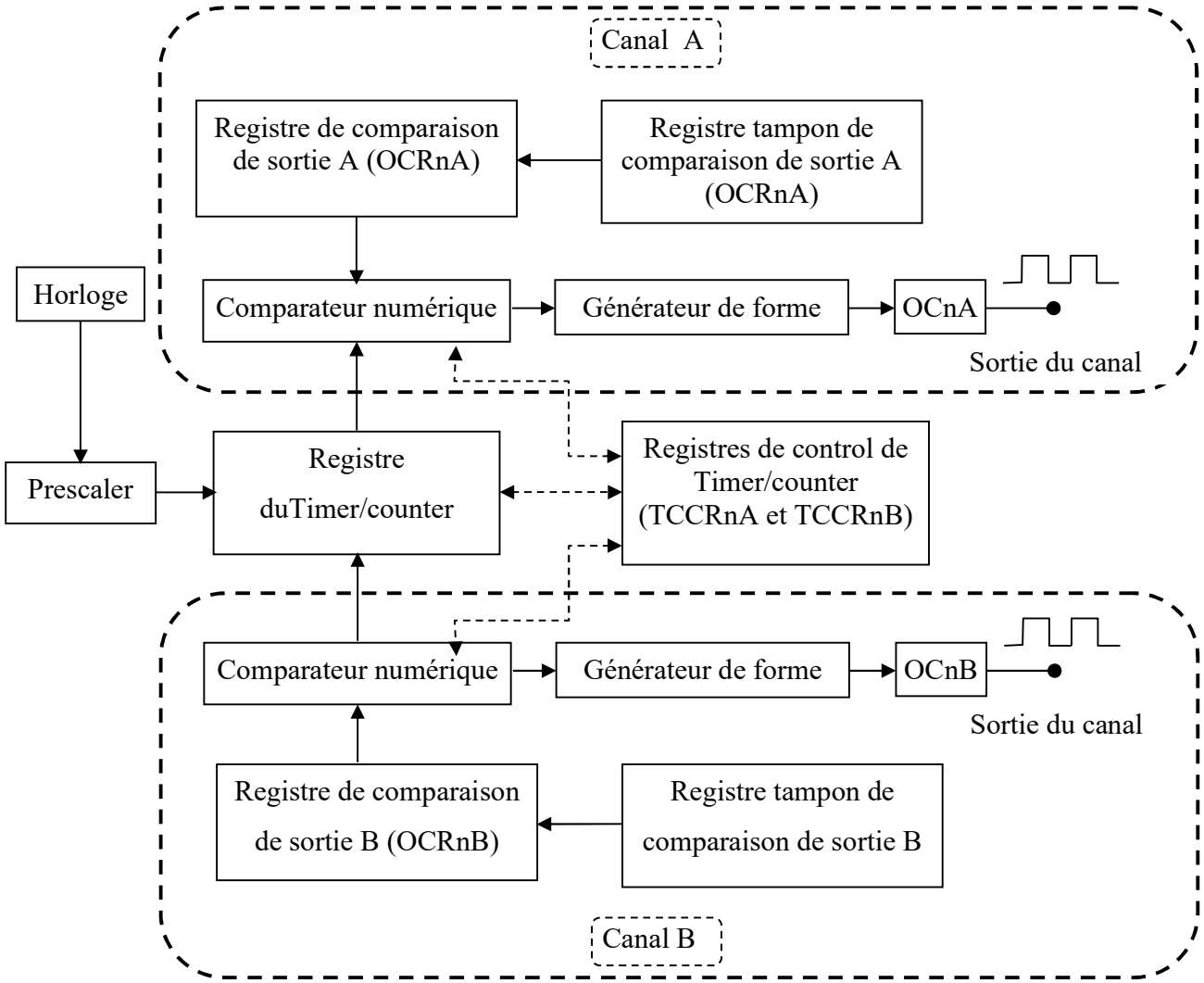


Figure2. Schéma block simplifié du timer/counter0 et timer/counter1

Le registre de contrôle TCCR0A

Bit	7	6	5	4	3	2	1	0	
	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Le registre de contrôle TCCR0B

Bit	7	6	5	4	3	2	1	0	
	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Le registre du timer/counter0

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	

Le registre de sortie de comparaison A

Bit	7	6	5	4	3	2	1	0	
	OCR0A[7:0]								OCR0A
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	

Le registre de sortie de comparaison B

Bit	7	6	5	4	3	2	1	0	
	OCR0B[7:0]								OCR0B
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	

Le registre des indicateurs d'interruption

Bit	7	6	5	4	3	2	1	0	
	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	TIFR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Le registre des masques d'interruption

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 3. Les registres du timer/counter0.

Tableau 2

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	TOP	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Tableau 3. Fonctionnalités des bits COM0A0-1 dans les différentes modes d'opération du timer/counter0

Mode d'opération	COM0A1	COM0A0	Description
Mode normal et CTC	0	0	OC0A déconnectée (opération du port normale)
	0	1	Inverser OC0A sur Compare Match
	1	0	Mise à 0 de OC0A sur Compare Match
	1	1	Mise à 1de OC0A sur Compare Match
Mode Fast PWM	0	0	OC0A déconnectée (opération du port normale)
	0	1	- WGM02 = 0: OC0A déconnectée (opération du port normale). - WGM02 = 1: Inverser OC0A sur Compare Match
	1	0	- Mise à 0 de OC0A sur Compare Match - Mise à 1 de OC0A sur TOP
	1	1	- Mise à 1 de OC0A sur Compare Match - Mise à 0 de OC0A sur TOP
Mode Phase correct PWM	0	0	OC0A déconnectée (opération du port normale)
	0	1	- WGM02 = 0: OC0A déconnectée (opération du port normale). - WGM02 = 1: Inverser OC0A sur Compare Match
	1	0	- Mise à 0 de OC0B sur Compare Match lors d'un comptage. - Mise à1 de OC0B sur Compare Match lors d'un décomptage.
	1	1	- Mise à 1 de OC0B sur Compare Match lors du comptage. - Mise à 0 de OC0B sur Compare Match lors de décomptage.

Tableau 4. Fonctionnalités des bits COM0B0-1 dans les différentes modes d'opération du timer/counter0

Mode d'opération	COM0B1	COM0B0	Description
Mode normal et CTC	0	0	OC0B déconnectée (opération du port normale)
	0	1	Inverser OC0B sur Compare Match
	1	0	Mise à 0 de OC0B sur Compare Match
	1	1	Mise à 1de OC0B sur Compare Match
Mode Fast PWM	0	0	OC0B déconnectée (opération du port normale)
	0	1	Réservée
	1	0	- Mise à 0 de OC0B sur Compare Match - Mise à 1 de OC0B sur TOP
	1	1	- Mise à 1 de OC0B sur Compare Match - Mise à 0 de OC0B sur TOP
Mode Phase correct PWM	0	0	OC0B déconnectée (opération du port normale)
	0	1	Réservée
	1	0	- Mise à 0 de OC0B sur Compare Match lors d'un comptage. - Mise à1 de OC0B sur Compare Match lors d'un décomptage.
	1	1	- Mise à 1 de OC0B sur Compare Match lors du comptage. - Mise à 0 de OC0B sur Compare Match lors de décomptage.

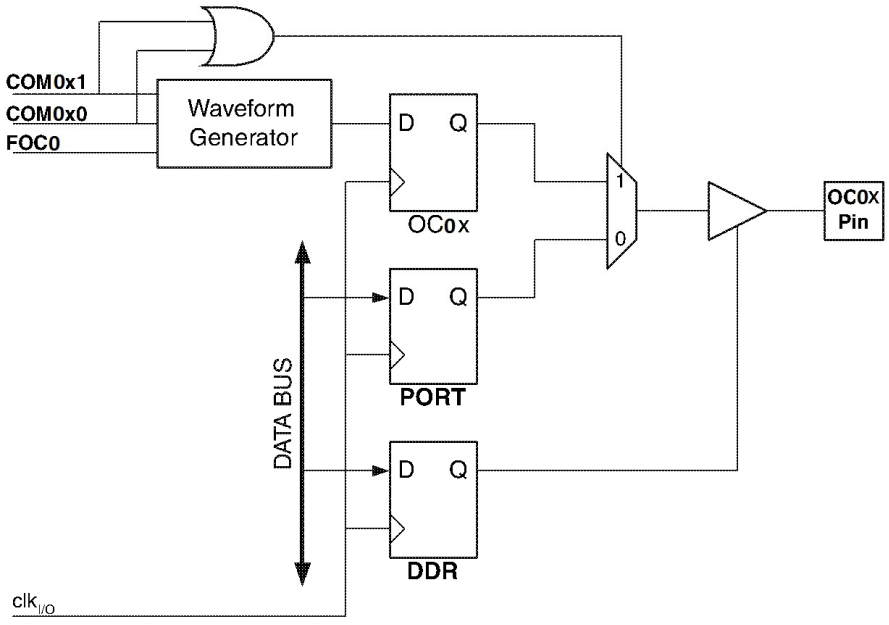


Figure 4. Schéma synoptique de l'unité de sortie de comparaison.

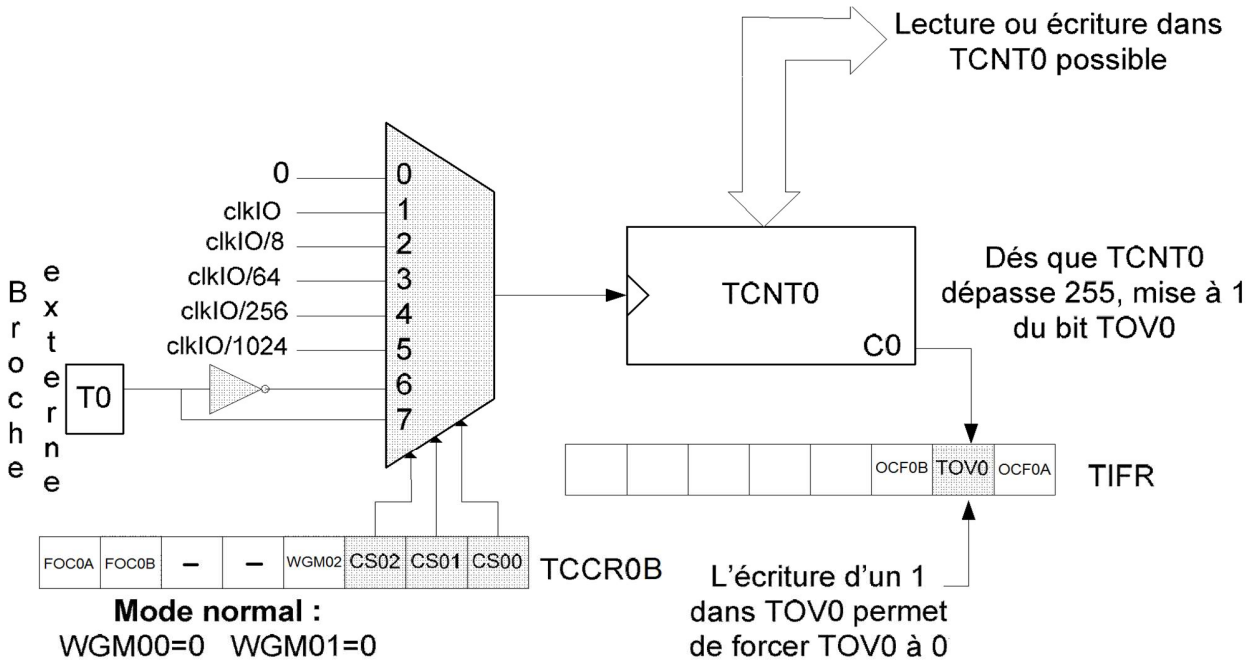


Figure 5. Timer/counter0 en mode normal.

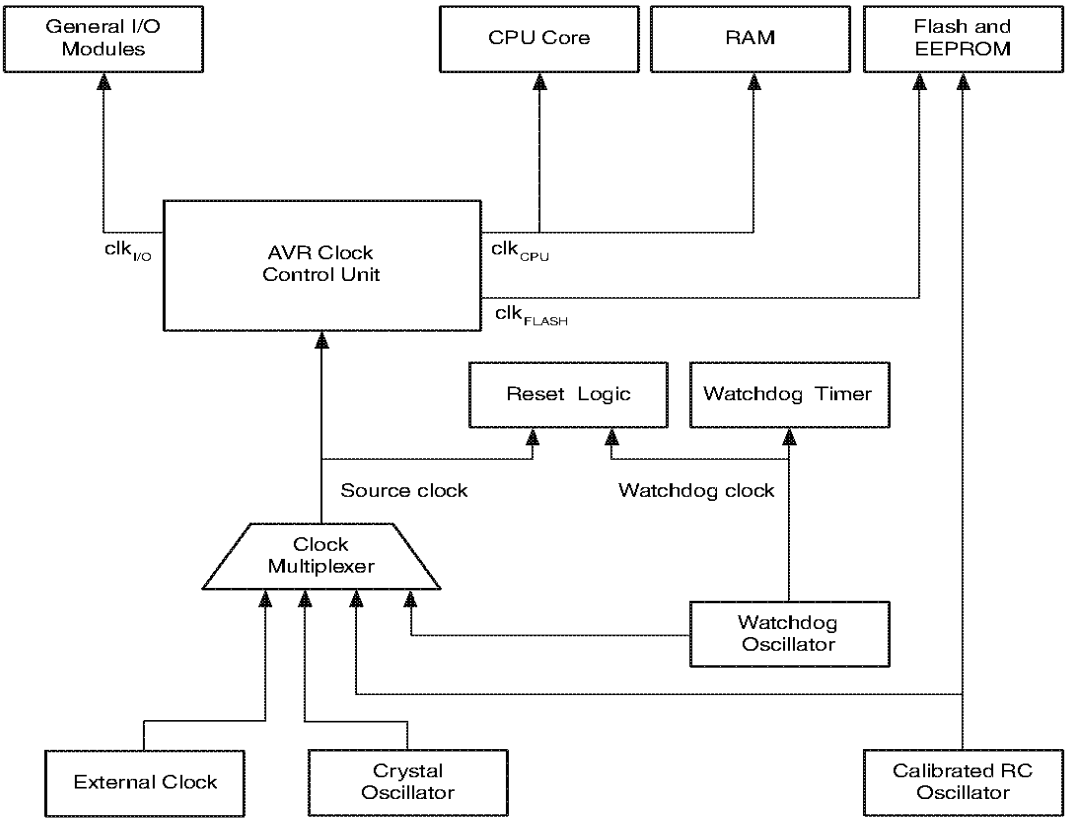


Figure 6. La distribution des signaux d'horloge.

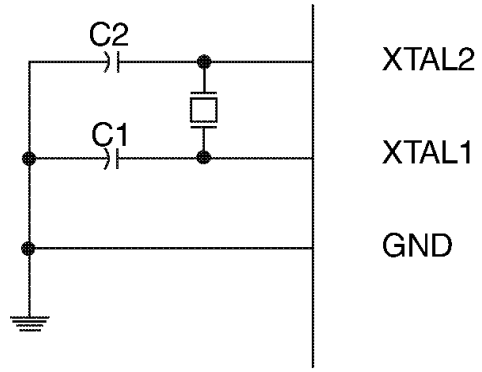


Figure 7 Connexion du ATtiny 2313 à un oscillateur à quartz.

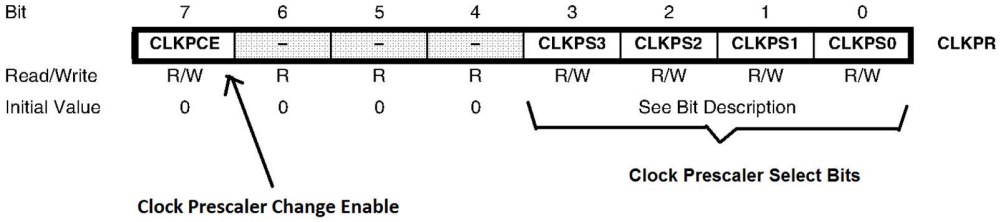


Figure 8. Clock prescale register

Tableau 5 La sélection du facteur de division du signal horloge provenant d'un oscillateur à quartz

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

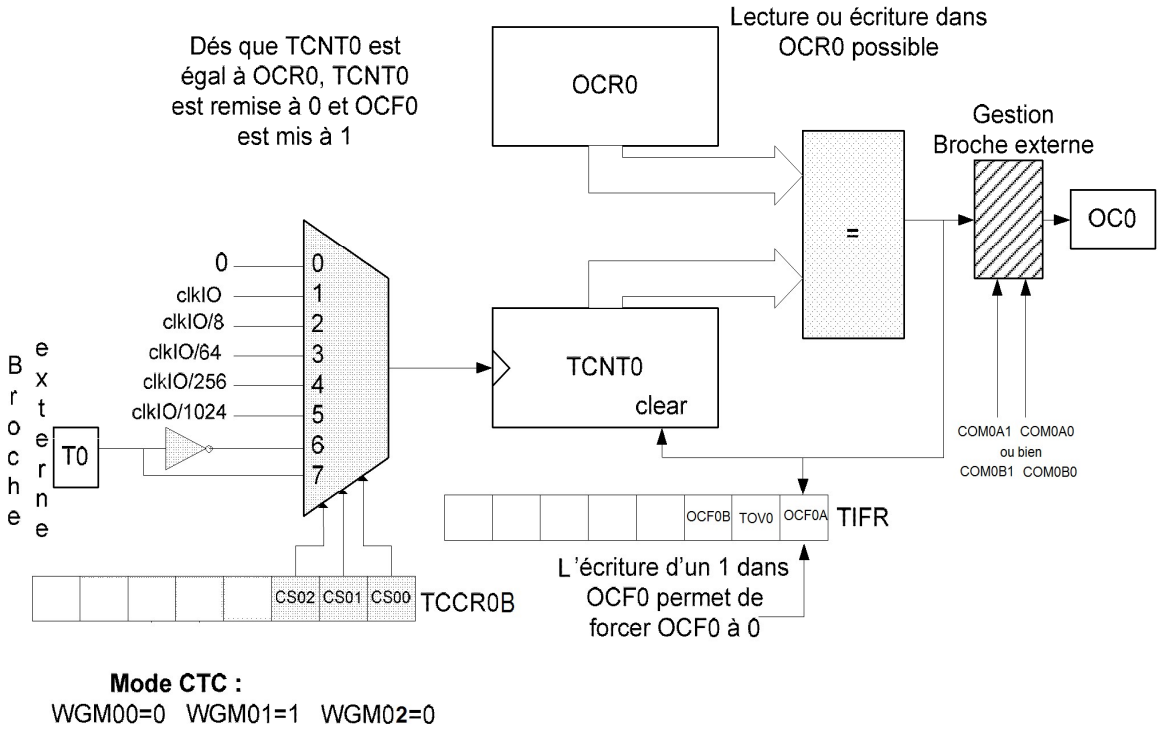


Figure 9 Timer/counter0 en mode CTC.

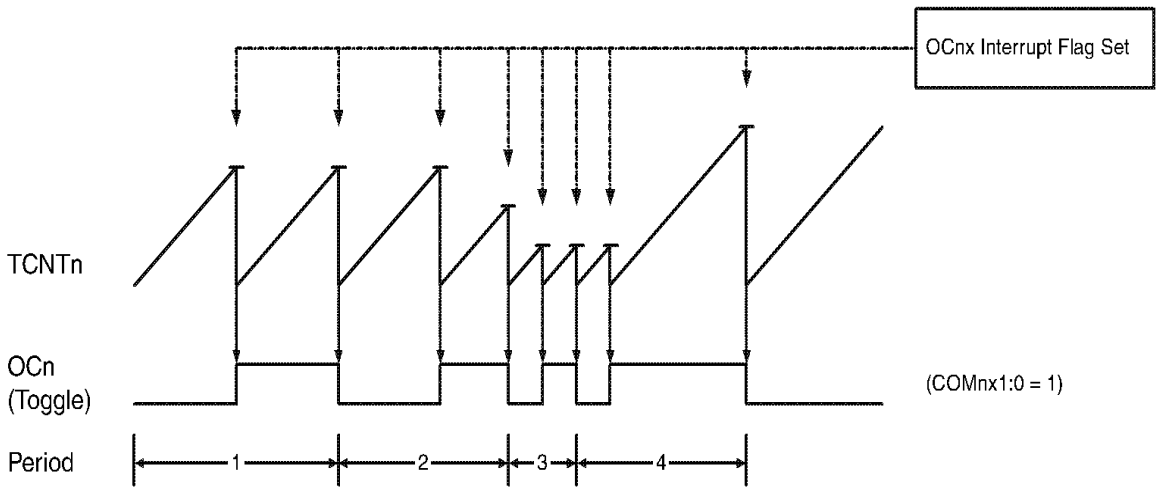


Figure 10 Digramme temporel du timer/counter0 en mode CTC.

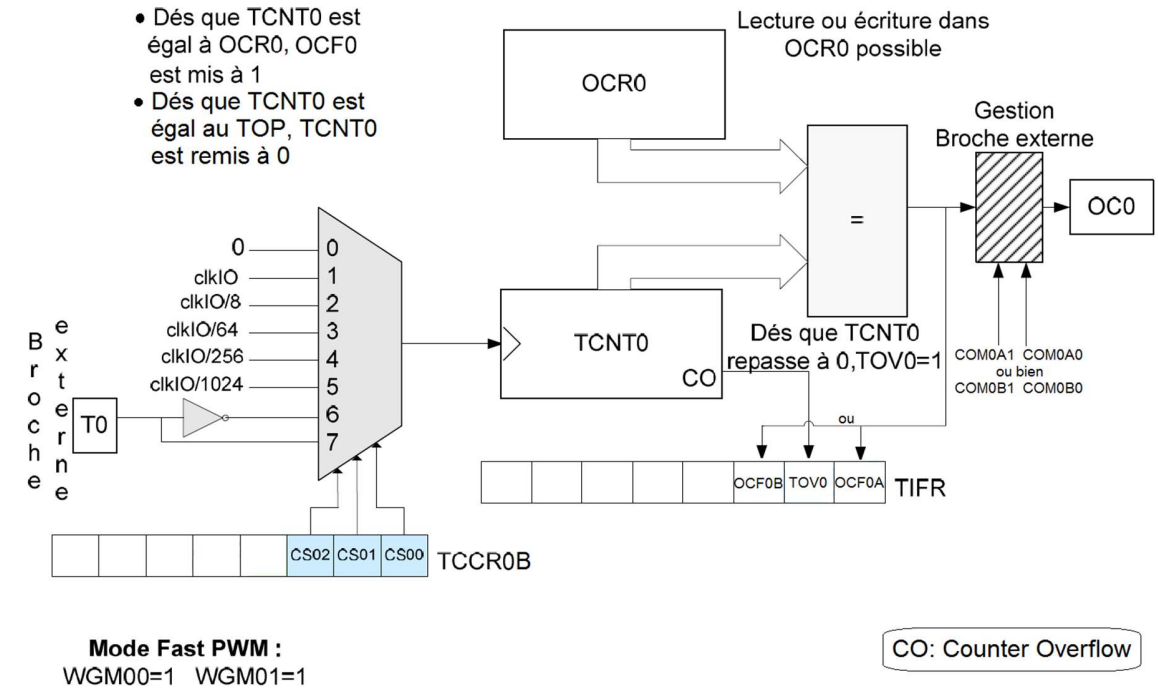


Figure 11 Timer/counter0 en mode Fast PWM (WGM02=0).

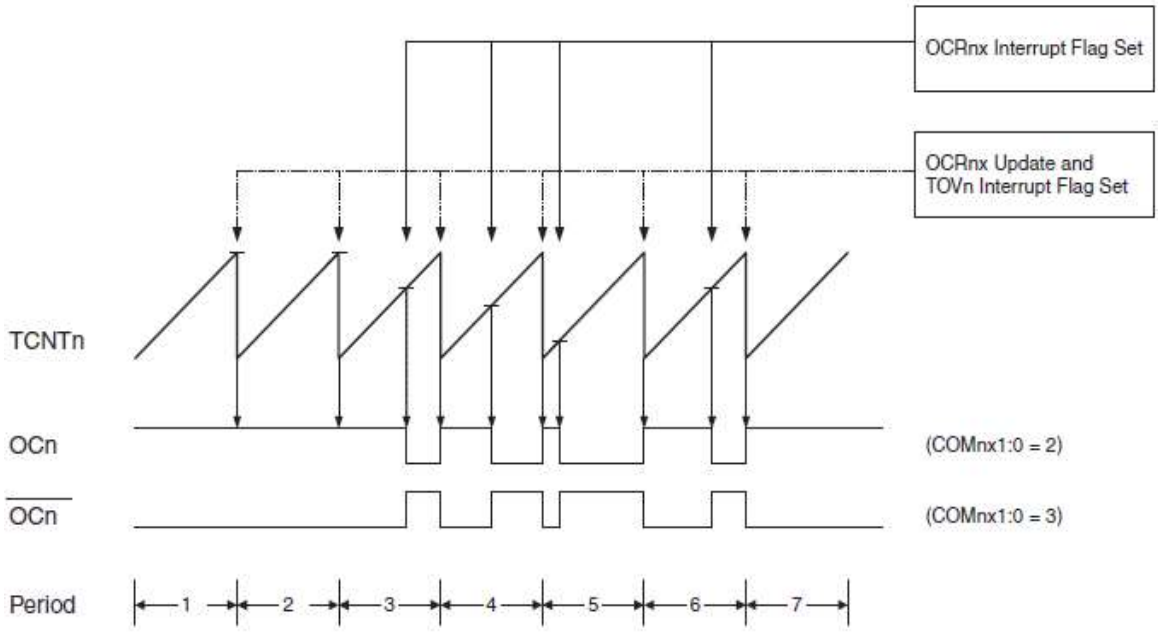


Figure 12 Digramme temporel du timer/counter0 en mode Fast PWM (WGM02=0).

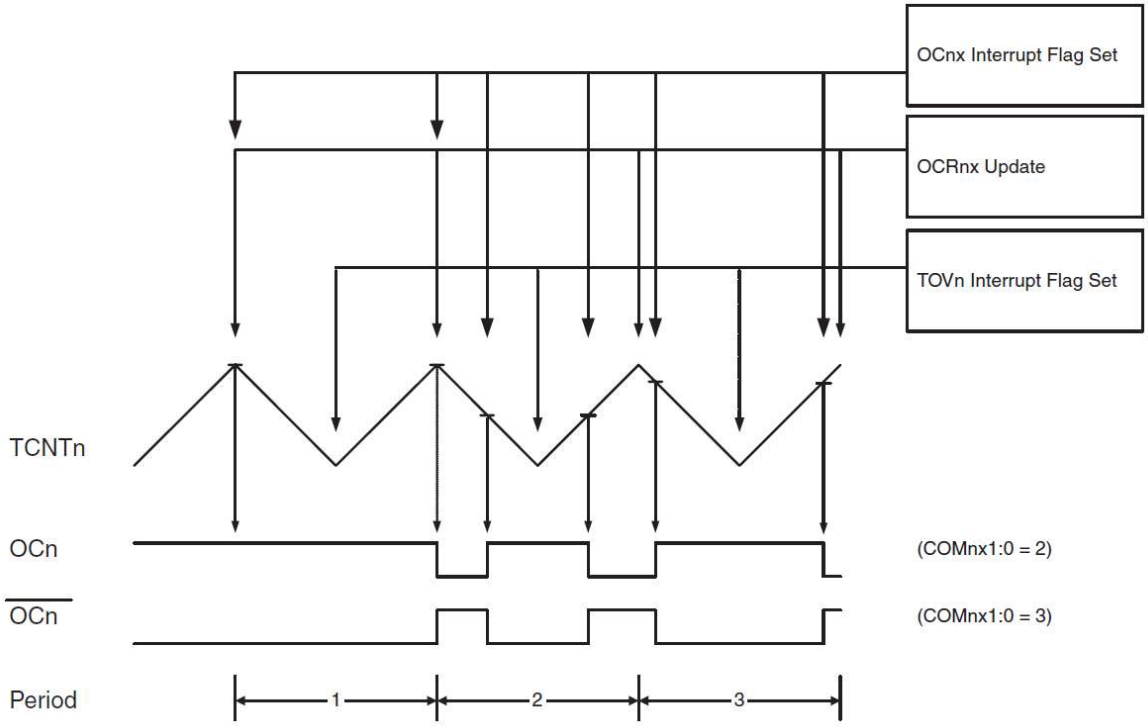


Figure 13 Digramme temporel du timer/counter0 en mode phase correct PWM (WGM02=0).

Bibliographie

Datasheet - ATtiny2313 - Atmel Corporation : www.atmel.com/images/doc2543.pdf

Christian Tavernier, *Microcontrôleurs AVR : des ATtiny aux ATmega - Description et mise en œuvre*, Dunod, Collection Technique et Ingénierie 2ème édition, 2013.

Florian Schaeffer, *Programmation en C des microcontrôleurs RISC AVR*, traduction Jean-Paul Brodier : Publitronic-Elektor International Media, cop. 2009.

Dhananjay V. Gadre, *Programming and Customizing the AVR Microcontroller*. McGraw-Hill, 2000.

Steven F. Barrett, Daniel J. PackAtmel, *AVR Microcontroller Primer: Programming and Interfacing*, Morgan & Claypool Publishers, 2007.

Richard H. Barnett, Sarah A. Cox, Larry D. O'Cull, *Embedded C Programming and the Atmel AVR*, Thomson Delmar Learning, 2002.

John Morton, *AVR: An Introductory Course*, Newnes, 2002.

Claus Kuhnel, *AVR RISC Microcontroller Handbook*, Newnes, 1998.

Richard H. Barnett, *Embedded C Programming And The Atmel AVR*, Delmar Cengage Learning: 2nd edition, 2006.