I LES CALCULATEURS NUMERIQUES

1° Généralités

Grosso modo, un calculateur numérique est un système ou un ensemble de blocs composés d'éléments et de circuits numériques qui se concertent pour exécuter un programme bien défini. Ce dernier est composé d'instructions codées et conservées dans une mémoire avec des données sur lesquelles ce programme travaille. Quand le calculateur reçoit l'ordre d'exécution d'un programme quelconque, il suit et respecte un certain ordre jusqu'à ce que le programme prenne fin.

D'autre part, nous pouvons dire qu'un calculateur numérique n'est rien d'autre qu'une machine ultra rapide qui traite les données, résout des problèmes, prend des décisions, tout cela sous la gouverne d'un programme. Un calculateur numérique peut être un Automate Programmable Industriel (voir le cours de la première année sur les Automatismes et l'Automatisation), un micro ordinateur ou tout autre système à base d'un ou de plusieurs microprocesseurs.

2° Eléments de base d'un calculateur numérique

La figure suivante représente schématiquement la structure d'un calculateur numérique, où l'on peut voir ses différents éléments de base :

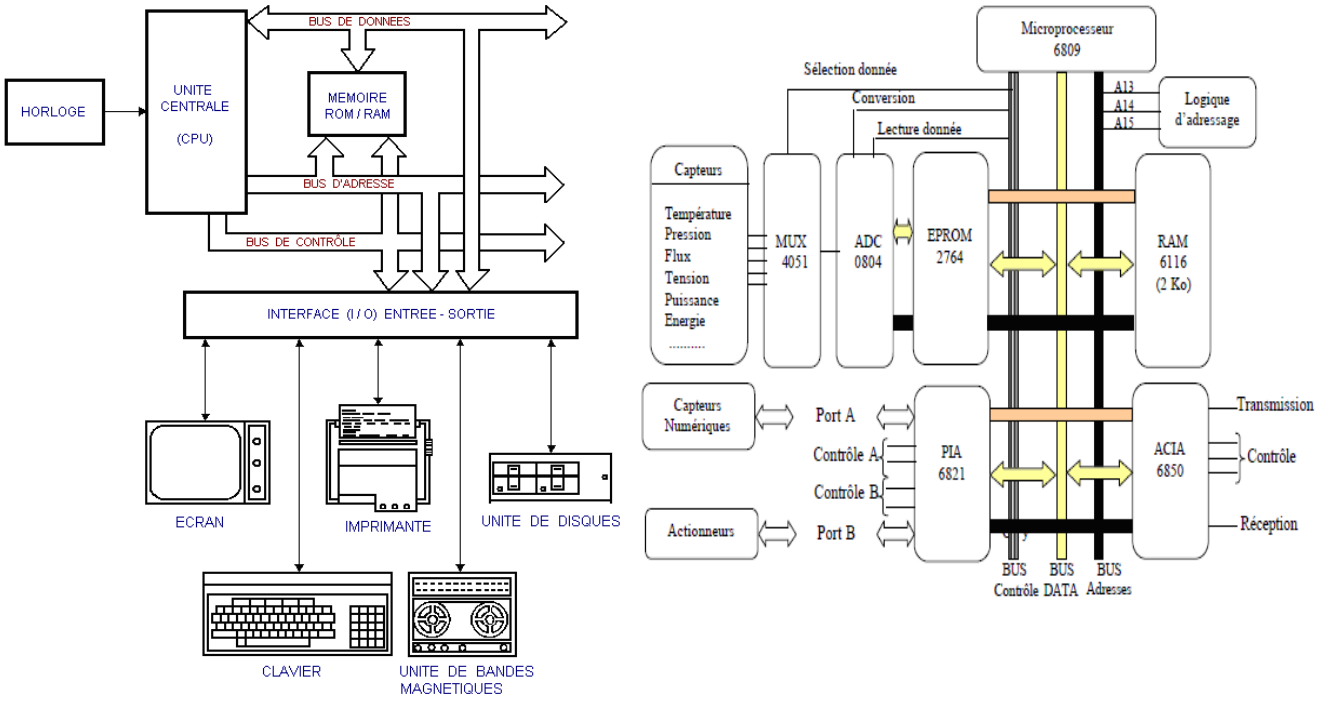


Figure n° 1 : Eléments de base d'un calculateur numérique

→ Le microprocesseur, appelé souvent Unité Centrale « UC ». C'est le cerveau du calculateur numérique, il sera étudié en détail par la suite.

→ L'unité de mémoire, comprenant aussi bien de la mémoire vive « RAM » que de la mémoire morte « ROM, EPROM, ... ». La section de la mémoire vive contient une ou plusieurs puces « LSI : Large Scale Integration » montées de manière à réaliser la capacité de la mémoire prévue. Cette section reçoit le programme et les données, ces derniers étant modifiés au fur et à

mesure que se déroule le traitement. Cette section sert également à la mémorisation des valeurs intermédiaires et des valeurs finales des calculs réalisés pendant l'exécution du programme.

La section de la mémoire morte comprend aussi une ou plusieurs puces « LSI » qui contiennent les instructions et les données inaltérables, par exemple les instructions du programme qui explore sans arrêt le clavier ou les données d'une table « ASCII » nécessaire pour sortir les informations sur un écran ou sur une imprimante.

→ L'interfaçage Entrée/Sortie est constitué de deux modules : module entrée et module sortie.

Le module d'entrée contient tous les dispositifs servant à prélever des informations et des données de l'extérieur du calculateur et à les transférer dans sa mémoire : c'est par cette unité que l'on introduit les programmes, les données avec un dispositif extérieur comme le clavier, les interrupteurs, les unités de disquettes ou de disques durs, des Convertisseurs Analogiques Numériques « CAN », etc.

Le module de sortie, regroupe tous les éléments que l'on a prévu pour transférer des données et des informations du calculateur vers le monde extérieur. Ces dispositifs de sortie reçoivent leurs ordres de l'unité de commande pour envoyer des données gardées en mémoire ou des résultats calculés par l'Unité Arithmétique et Logique « UAL » vers l'extérieur. Parmi les exemples d'éléments de sortie, nous pouvons citer les différents afficheurs, les différents voyants, les écrans de visualisation, les imprimantes, les unités de disquettes ou de disques durs, des Convertisseurs Numériques Analogiques « CNA », etc.

→ Un bus de données, permettant les transferts d'informations sur un faisceau de plusieurs conducteurs parallèles. Le nombre de fil est l'une des caractéristiques essentielles de l'UC. En principe, la largeur du bus de données est égale à la taille des mots manipulés par le microprocesseur (8 bits, 16 bits, ...). Le bus est unidirectionnel pour les trois liaisons suivantes :

Mémoire Morte → Microprocesseur,
Microprocesseur → Unité de Sortie,
Unité d'Entrée → Microprocesseur

et bidirectionnel pour la communication du microprocesseur avec la mémoire vive.

→ Un bus d'adresses, est aussi un ensemble de conduits parallèles unidirectionnel permettant de pointer toutes les cases mémoires adressables par le calculateur numérique. Le nombre de lignes de ce bus détermine la taille maximale de la mémoire. Si par exemple ce nombre est de 16, le champ mémoire qu'on peut adresser est de 64 Kilo Octets.

→ Un bus de commande et de contrôle bidirectionnel, comportant :

- *Des lignes qui permettent à l'UC de spécifier à la RAM ou aux ports d'entrée et de sortie si elle veut faire une écriture ou une lecture.*
- *Des lignes utilisées par l'UC pour répondre aux périphériques, par exemple : acceptation d'une demande d'interruption ou d'un accès direct mémoire, ...*

N.B. *Le nombre de lignes dépend du type de microprocesseur utilisé !*

3° Définition et évolution des microprocesseurs

Les microprocesseurs sont parmi nous ! Apparus vers les années 70, les microprocesseurs « µP » ont rapidement envahi presque tous les domaines. Ils sont utilisés actuellement dans des millions de

systèmes : les terminaux bancaires, les instruments de mesure, les lecteurs de disques compactes, les magnétoscopes, les pompes à essence, les appareils électroménagers, ..., sans oublier les microordinateurs et des gros systèmes informatiques.
Un µP est généralement une puce intégrée programmable renfermant tous les circuits de l'unité de commande et de l'unité arithmétique et logique. On l'appelle souvent, le cerveau du calculateur numérique car c'est lui qui se charge des fonctions suivantes :

→ *Fournir les signaux de synchronisation et de commande à tous les éléments du calculateur.*
→ *Prendre en charge les instructions et les données en mémoire.*

→ *Transférer les données entre la mémoire et les dispositifs d'Entrée/Sortie et vice versa.*

→ *Décoder les instructions des programmes.*

→ *Effectuer les opérations arithmétiques et logiques correspondant aux instructions.*

→ *Réagir aux signaux de commande produits par les E/S comme le signal d'initialisation (Reset), les signaux correspondant aux interruptions, ...*

D'après le site web : http://fr.wikipedia.org/wiki/Microprocesseur, le premier micro processeur a été inventé, en 1971, par l'ingénieur d'Intel Mr Marcian Hoff (surnommé Ted Hoff). Et le premier microprocesseur qui a été commercialisé, le 15 novembre 1971, est l'Intel 4004 (à 4-bits). Il fut suivi par l'Intel 8008. Ce microprocesseur a servi initialement à fabriquer des contrôleurs graphiques en mode texte, mais jugé trop lent par le client qui en avait demandé la conception, il devint un processeur d'usage général. Ces processeurs sont les précurseurs des Intel 8080 Zilog Z80 et de la future famille des Intel x86. Le tableau suivant décrit les principales caractéristiques des microprocesseurs fabriqués par Intel et montre la fulgurante évolution des microprocesseurs autant en augmentation du nombre de transistors, en miniaturisation des circuits et en augmentation de puissance.

Date	Nom	Transistors	Finesse de gravure (µm)	Fréquence	Largeur des données	MIPS
1971	4004	2 300			4 bits / 4 bits bus	
1974	8080	6 000	6,0	2 MHz	8 bits / 8 bits bus	0,64
1979	8088	29 000	3,0	5 MHz	16 bits / 8 bits bus	0,33
1982	80286	134 000	1,5	6 MHz	16 bits / 16 bits bus	1,00
1985	80386	275 000	1,5	16 MHz	32 bits / 32 bits bus	5,00
1989	80486	1 200 000	1,0	25 MHz	32 bits / 32 bits bus	20,0
1993	Pentium	3 100 000	0,8	60 MHz	32 bits / 64 bits bus	100
1997	Pentium II	7 500 000	0,35	233 MHz	32 bits / 64 bits bus	300
1999	Pentium III	9 500 000	0,25	450 MHz	32 bits / 64 bits bus	510
2000	Pentium 4C	42 000 000	0,18	1,5 GHz	32 bits / 64 bits bus	1 700
2004	Pentium 4D	125 000 000	0,09	3,6 GHz	32 bits / 64 bits bus	9 000
2006	Core 2™ Duo	291 000 000	0,065	2,4 GHz	64 bits / 64 bits bus	22 000
2007	Core 2™ Quad	2 x 291 000 000	0,065	3 GHz	64 bits / 64 bits bus	2 x 22 000
2008	Core 2™ Duo (Penryn)	410 000 000	0,045	3,16 GHz	64 bits / 64 bits bus	Env. 24 200

Tableau n° 1 : Principales caractéristiques des microprocesseurs fabriqués par Intel

***Date** : l'année de commercialisation du microprocesseur.*
***Nom** : le nom du microprocesseur.*
***Transistors** : le nombre de transistors contenus dans le microprocesseur.*
***Finesse de gravure** : le diamètre (en micromètres) du plus petit fil reliant deux composantes du microprocesseur. En comparaison, l'épaisseur d'un cheveu humain est de 100 microns !*
***Fréquence de l'horloge** : la fréquence de l'horloge de la carte mère qui cadence le microprocesseur. MHz = millions de cycles par seconde. GHz = milliards de cycles par seconde.*
***Largeur des données** : le premier nombre indique le nombre de bits sur lequel une opération est faite. Le second nombre indique le nombre de bits transférés à la fois entre la mémoire et le microprocesseur.*
***MIPS** : le nombre de millions d'instructions complétées par le microprocesseur en une seconde.*

4° Etude de trois microprocesseurs de la famille Motorola

4.1 Le microprocesseur 6802

Le µP 6802 de Motorola est un processeur 8 bits conçu en technologie N-MOS. Il est tout à fait compatible TTL et il est encore utilisé sur certains systèmes. Il est né deux ans après le 6800, ce microprocesseur intègre sur le même circuit une mémoire RAM de 128 Octets dont les 32 premiers Octets peuvent être sauvegardés en mode faible consommation, par pile. Tous les registres internes (qui sont de l'ordre de six) sont analogues à ceux du 6800, ce qui implique même jeu d'instructions (72 instructions) et même mode d'adressage.

4.2 Le microprocesseur 6809

Le µP 6809 est un processeur 8 bits de haute gamme, dont l'organisation interne est orientée 16 bits. Il est tout à fait compatible au niveau source avec les programmes écrits pour le 6800 ou le 6802 et il est conçu en technologie N-MOS. Il existe deux versions de ce processeur, le 6809 (avec horloge interne) et le 6809E (avec horloge externe). Leur brochage diffère ainsi que leurs signaux mais le jeu d'instruction est identique. Dans ce polycopié, nous allons étudier le 6809.

4.2.1 Description générale

Le circuit correspondant comporte 40 broches (voir la figure suivante). La signification des différentes broches du boîtier est résumée ci-dessous :

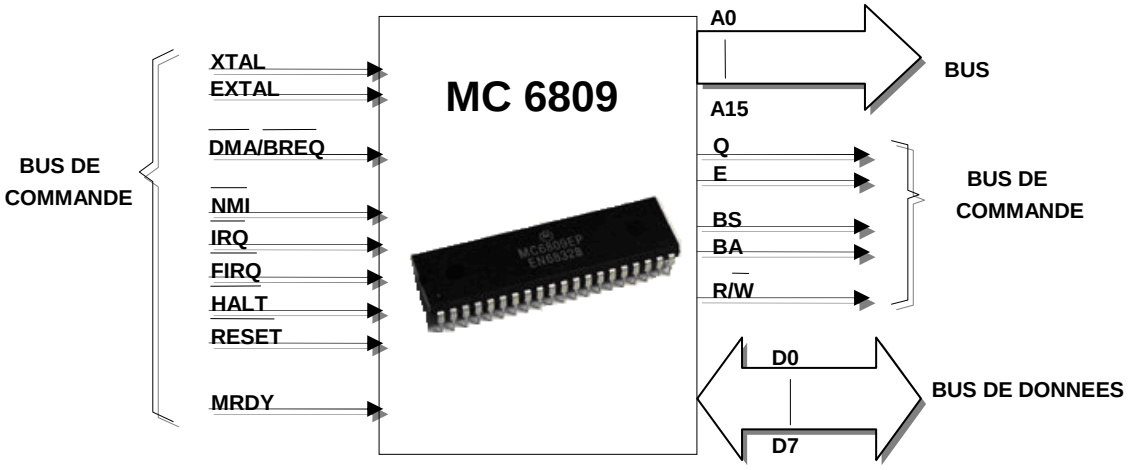


Figure n° 2 : Circuit correspondant au microprocesseur 6809

Numéro des broches	Désignation	Signification et fonction
1	Vss	Masse, à relier à 0 V
2	$\overline{\text{NMI}}$	Non Maskable Interrupt, c'est une ligne d'entrée d'interruption non masquable, active au niveau bas. Elle permet d'exécuter une routine d'interruption dont l'adresse est contenue dans le vecteur $FFFC-$FFFD. Cette ligne est dévalidée par Reset et n'est revalidée qu'après un chargement du registre S en mode immédiat. Le contenu de la totalité des registres du 6809 est sauvegardé dans la pile système.
3	$\overline{\text{IRQ}}$	Interrupt ReQuest, c'est une ligne d'entrée des demandes d'interruption masquables, active à l'état bas. Elle fonctionne comme sur le 6802. Elle a son vecteur en $FFF8-$FFF9. Elle est conditionnée par le bit 1 du registre CCR et tous les registres sont sauvegardés. Le déroulement de la routine peut être interrompu par FIRQ ou NMI.

4	$\overline{\text{FIRQ}}$	Fast Interrupt ReQuest, c'est une ligne d'entrée. Elle est masquée ou non suivant l'état du bit F du registre CCR. Les registres du 6809 ne sont sauvegardés que partiellement.
5	BS	Bus Status : cette ligne, en conjointe avec BA indique l'état du microprocesseur. Les quatre configurations possibles de ces lignes et leur signification sont : BA = 0 et BS = 0, le µP fonctionne normalement, BA = 0 et BS = 1, le µP reconnaît une interruption, BA = 1 et BS = 0, le µP reconnaît une synchronisation, BA = 1 et BS = 1, le µP est à l'arrêt. La reconnaissance d'interruption se produit pendant la recherche des vecteurs d'interruption à Reset, NMI, IRQ, FIRQ, SWI1, SWI2 et SWI3. La reconnaissance d'une synchronisation indique que le µP attend une synchronisation externe sur une ligne d'interruption.
6	BA	Bus Available = Bus de données disponible : cette ligne signale que les bus trois états sont passés en haute impédance.
7	VCC	Alimentation, à relier à +5V
de 8 à 23	de A0 à A15	Bus d'adresse sur 16 bits
de 24 à 31	de D7 à D0	Bus de données sur 8 bits
32	R/$\overline{\text{W}}$	Read/Write : c'est avec ce signal qu'on détermine le sens de transfert des données. S'il est à l'état bas, il s'agit d'une écriture et pour l'état inverse, il correspond à une lecture.
33	$\overline{\text{DMA/BREQ}}$	Direct Memory Adress/Bus REQuest : si cette ligne est au niveau bas, ceci permet de libérer le bus pour faire de l'accès direct mémoire ou du rafraîchissement de mémoire. Elle est lue pendant le front descendant de « Q », le µP termine l'instruction en cours et répond en mettant BA et BS au niveau haut.
34	E	C'est une sortie horloge pour le timing des bus (synchronisation avec la périphérie) dont la fréquence est celle de base du µP.
35	Q	C'est une sortie horloge en quadrature avec « E ». Les adresses sur le bus seront validées sur le front montant de « Q », tandis que les données seront verrouillées sur le front descendant de « E ».
36	MRDY	Memory ReaDY : c'est une entrée de commande qui permet l'allongement de la durée du signal « E » et ceci pour utiliser les mémoires lentes. En effet, lorsque MRDY = 1, « E » est en fonctionnement normal et pour MRDY = 0, « E » s'allonge de multiples entiers de ¼ de cycles de bus, ce qui permettra l'utilisation des mémoires lentes.
37	$\overline{\text{Reset}}$	C'est une entrée qui permet d'initialiser le µP. Si Reset = 0, l'instruction en cours est arrêtée, le registre de page est mis à zéro, les interruptions IRQ et FIRQ sont masquées et l'interruption non masquée NMI est désarmée.
38 et 39	Xtal et EXtal	Ce sont deux entrées de connexion d'un quartz externe de 4, 6 ou 8 Mhz, de caractéristiques identiques au 6802.
40	$\overline{\text{Halt}}$	Ligne d'entrée pour le µP, lorsqu'elle est au niveau haut, le 6809 travaille normalement. Mais si par un signal électrique, on la fait passer à « 0 » et tant qu'on l'y maintient, on arrête le fonctionnement du microprocesseur. Ce dernier termine l'instruction en cours puis positionne : BA (Bus Available) et BS (Bus Status) à 1. Le déroulement reprend dès que la broche Halt est à 1 et sans perte d'informations. Les lignes d'interruption IRQ et FIRQ sont dévalidées : Reset et NMI sont valides mais leur traitement ne se fera qu'à la libération du 6809. Les horloges ne sont pas affectées.

Tableau n° 2 : La signification des différentes broches du microprocesseur 6809

4.2.2 Architecture interne

Les éléments essentiels constituant le µP 6809 sont :

→ *Un bus de données, bidirectionnel de 8 bits.*

→ *Un bus d'adresses, de 16 conduits monodirectionnels.*

→ *<u>Un bus de contrôle et de commande</u>, gérant les signaux suivants : R/$\overline{W}$, MRDY, IRQ, NMI, FIRQ,FIRQ, Xtal, Extal, Reset, Halt, DMA/BREQ, NMI, BA, BS, E et Q.*

D'autre part, le microprocesseur 6809 est composé de neuf registres internes que nous allons décrire ci dessous (voir figure n° 3).

→ *<u>Les accumulateurs A et B</u> sont deux registres de taille 8 bits jouant des rôles identiques, à part pour les deux instructions ABX et DAA.*

→ *<u>Le double accumulateur D</u> est en fait la concaténation des deux registres A et B, l'accumulateur A représentant les poids forts et B les poids faibles. L'accumulateur D dispose d'instructions spécifiques : addition, soustraction, comparaison, etc. permettant de travailler directement sur 16 bits.*

→ *<u>Le registre DP (Direct Page)</u> : est un registre 8 bits. Il forme la partie haute de l'adresse à pointer dans le cas d'un adressage direct. Il est automatiquement remis à zéro par un Reset.*

→ *<u>Les registres d'index X et Y</u> : Ils sont de taille 16 bits. D'autre part, ils permettent d'adresser tout l'espace mémoire avec en plus la capacité d'être pré-décrémenté ou post-décrémenté (ce qui facilite le traitement de variables en tables).*

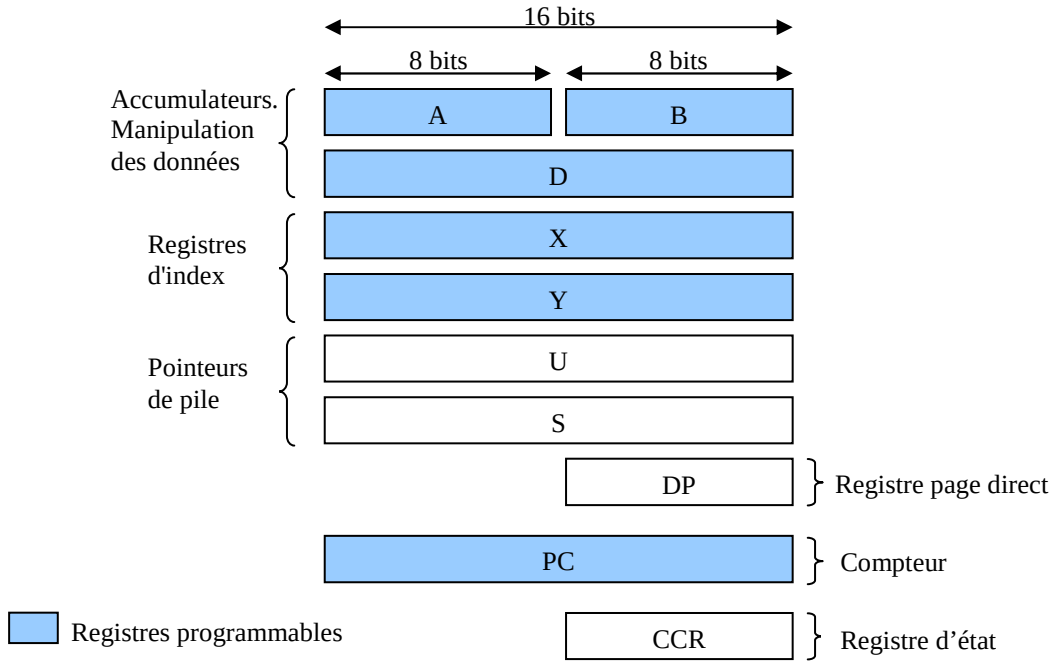


Figure n° 3 : Description des registres internes du µP 6809

→ *<u>Le compteur programme (Program Counter - PC)</u> : est un registre de 16 bits. Il peut pointer sur toutes les cases mémoires. Il contient l'adresse de la prochaine instruction à exécuter. Pour vos applications, il faut l'initialiser à l'adresse de départ du programme à exécuter.*

→ *<u>Les pointeurs de pile U et S</u> : Ce sont deux registres 16 bits qui fonctionnent tous deux identiquement. Ils opèrent en mode « dernier entré →premier sorti ». Ces deux registres peuvent à l'occasion servir de registre d'index avec la totalité des possibilités de X et Y. Le registre S (System) est utilisé par le 6809 pour toutes les opérations de sauvegarde en cas d'interruption ou de saut à un sous-programme (adresse de retour). Le registre U (User) est entièrement réservé à l'utilisateur.*

→ *Le registre d'état (Code Condition Register - CCR) : Ce registre définit à tout instant l'état du µP résultant d'une instruction. Il est composé de 8 bits jouant chacun un rôle important pour les instructions de sauts ou de branchements conditionnels, … :*

« C » ***Carry*** *: Ce bit prend la valeur 1 chaque fois que le résultat d'une instruction arithmétique ou logique dépasse 8 bits, c'est à dire nous avons une retenue. On peut considérer cette retenue comme le 9ème bit pour les accumulateurs A et B.*
N.B. *« C » reste à zéro dans les cas contraires !*

« V » ***Overflow*** *: C'est un bit de débordement. Il est mis à 1 si le résultat d'un complément à 2 déborde : c'est à dire dépasse l'octet. Sinon, il reste à 0.*

« Z » ***Zero*** *C'est un bit qui indique simplement si le résultat d'une instruction ou d'une opération est nul. Dans ce cas, il est mis à 1. Pour tout autre cas, il est mis à zéro.*

« N » ***Negative*** *: C'est un indicateur de signe. Il est positionné à 1 si le résultat d'une instruction ou d'une opération est négatif. Sinon, il reste à 0.*

« I » ***Interrupt*** *: C'est un indicateur d'interruption. Il est positionné, en général, par le programmeur sauf quelque fois sur initiative du microprocesseur. Il s'agit du masque pour la prise en compte des demandes d'interruption masquables : IRQ. Il sera automatiquement positionné à 1 si une demande d'interruption IRQ ou NMI.*

« H » ***Half*** *Carry : C'est un indicateur de demi-retenue. Lors d'une opération, le µP traite les octets bit par bit. Si en traitant le 4ème bit de l'octet, il doit faire une retenue vers le bit 4. Il a l'obligeance de nous avertir en mettant H à 1.*
N.B. *Cet indicateur est exploité par l'instruction « DAA » (Decimal Adjustment of Accumulator A) pour formater en code BCD des additions de nombres BCD.*

« F » ***(FIRQ mask)*** *Ce bit conditionne le traitement de la ligne d'interruption FIRQ. Si F = 1, les interruptions seront masquées. Il est à zéro après un Reset et dans ce cas, FIRQ est dévalidée. Si l'utilisateur le force à 1, FIRQ est traitée.*
N.B. *Cette indicateur est positionné également par : NMI et SWI !*

« E » ***(Entire flag)*** *Ce bit nous renseigne sur le nombre de registres rangés dans la pile. Il est utilisé par l'instruction RTI pour déterminer le nombre d'octets que la pile doit restituer.*

→ L'unité arithmétique et logique (Arithmetic and Logic Unit - ALU) : C'est l'organe le plus complexe du µP. Son rôle est d'effectuer les opérations arithmétiques et logiques sur les données qui lui sont fournies par ses entrées. Parmi ces opérations, nous pouvons citer les logiques (ET, OU, AND, NAND, NOR, ...), les opérations arithmétiques (addition, soustraction, ...), les opérations de traitements particuliers (différents décalages, tests, ...), etc.

→ Le bloc de décodage d'instruction et de contrôle : C'est ce bloc qui génère les microcommandes propres à chaque instruction à exécuter, c'est à dire une fois que l'instruction se trouve dans le registre d'instruction, ce bloc se charge du décodage et de la génération des signaux internes et externes correspondant à cette instruction.

4.2.3 Différents modes d'adressage

Le µP 6809 possède 59 instructions 'de base'. Cela peut paraître faible, mais combinées aux différents modes d'adressage, elles offrent 1464 possibilités. Signalons au passage qu'une instruction comporte de un à quatre octets. Généralement, le premier octet indique l'action à effectuer, les suivants précisent les opérandes ou sur quelques registres cette action agira. Dans ce qui suit, nous allons présenter, succinctement, quelques modes d'adressage correspondant à ce microprocesseur.

→ **Adressage inhérent ou implicite** : Le code opération contient toute l'information nécessaire à l'exécution de l'instruction. Les instructions correspondant sont codées :

- soit sur un seul octet, exemple : ABX, ASLA, RORA, NEGA, COMA, DEÇA, ...
- soit sur deux octets, exemple : SWI2, TFR, PSHS, PULU, CWAY, ...

Exemples : **CLRA**, initialisation de l'accumulateur **A** par $00 (figure 4)
INCB, incrémentation de **1** de l'accumulateur **B.**
AAX, addition de l'accumulateur **A** au registre **X**.

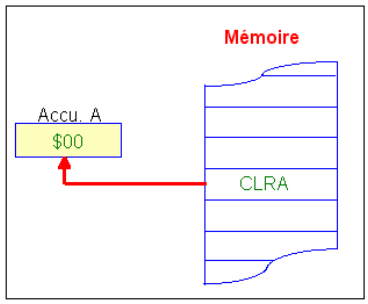


Figure n° 4 : Exemple d'utilisation du mode d'adressage inhérent pour le µP 6809

→ **Adressage immédiat** : Dans ce mode d'adressage, le code opératoire est suivi d'une valeur qui est l'opérande de l'instruction (sur un ou deux octets). Ceci permet de charger les registres internes du µP avec la valeur de l'opérande. Il existe trois types d'instructions dans ce mode d'adressage.

*Exemple : **LDA #$80,** Chargement de **A** avec la valeur hexadécimal **80***

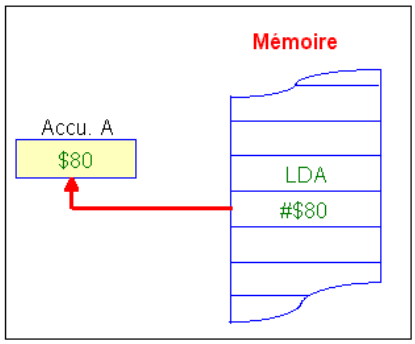


Figure n° 5 : Exemple d'utilisation du mode d'adressage immédiat pour le µP 6809

Le symbole « **$** » précise que la donnée est en hexadécimal. Pour les instructions **LDr** (**r** : registre quelconque), la valeur à charger dans le registre doit être du même type que l'accumulateur (8 bits pour les accumulateurs **A, B** et 16 bits pour les registres **X, Y, U, S et D**). Ce type d'adressage permet d'initialiser les registres internes du microprocesseur.

N.B : *Ces instructions peuvent atteindre quatre octets comme LDS (Chargement du pointeur de Pile par avec le contenu mémoire) ou CMPU (Comparaison mémoire avec le pointeur de Pile).*

→ **Adressage direct** : Ce mode d'adressage présente l'avantage de ne nécessiter que 2 octets pour avoir accès à des données situées sur l'ensemble de l'espace mémoire du µP. Le premier octet définit le code opératoire, le second représente les 8 bits de poids faibles de l'adresse effective dont les 8 bits de poids fort se trouvent dans le registre de page du µP

(DPR). Il suffit donc d'initialiser le registre de page (DPR) pour pouvoir travailler en adressage direct sur 256 octets de la page choisie ; au delà, il faut de nouveau accéder au DPR. A la mise sous tension, le registre de page est mis à zéro. On aura donc accès aux 256 octets de la page 0. Il existe deux types d'instructions dans ce mode d'adressage. On notera que l'adressage direct sera spécifié par le signe « < » placé devant l'opérande, dans la syntaxe assembleur.

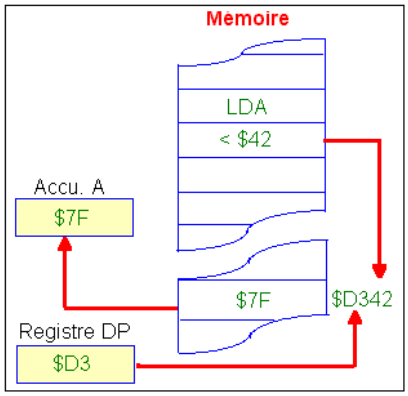


Figure n° 6 : Exemple d'utilisation du mode d'adressage direct pour le µP 6809

Exemple : *Si le registre de page vaut **$D3** : l'instruction **LDA <$42** chargera l'accumulateur « **A** » par le contenu de la case mémoire **$ D342** qui est **$7F**.*

NB : Ce mode présente l'avantage d'être exécuté rapidement

→ **Adressage étendu** : Dans ce cas, le champ adresse qui suit l'instruction contient l'adresse effective sur deux octets, ça permet d'atteindre toute la mémoire. Il existe deux types d'instructions dans ce mode d'adressage.

Exemple : *Charger l'accumulateur A avec la valeur hexadécimal $F5 qui est le contenu de l'adresse $15CD (LDA $15CD ou LDA > $15CD).*

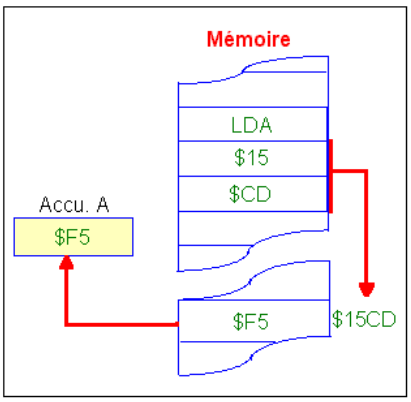


Figure n° 7 : Exemple d'utilisation du mode d'adressage étendu pour le µP 6809

Autres exemples :

LDA $ 1000 → charge A avec le contenu de l'adresse $ 1000

SBCB $ 29D0 → soustrait à B le contenu de l'adresse $ 29D0 et le contenu du bit C de CCR

CMPX $ 4800 → compare X au contenu de $ 4800-$ 4801

→ **Adressage relatif court** : Ce mode d'adressage est réservé pour les instructions de branchement qui sont d'une longueur de 2 octets. Le premier octet détermine le code opératoire qui spécifie le type de branchement en même temps que le test correspondant. Le deuxième octet correspond au déplacement qui est soit positif ou négatif : il est stocké sous forme binaire d'un nombre en complément à 2 de 8 bits dont la valeur décimale se situe entre -128 et +127. Une instruction de branchement est toujours précédée par une opération qui peut s'assimiler à un test. Le résultat de cette opération affecte 0 ou 1 sur un ou plusieurs bits du registre CCR. Le branchement se fait, ou se fait pas, suivant l'état 0 ou 1 d'un ou plusieurs bits du registre codes conditions (CCR).

→ **Adressage relatif long** : Ce mode d'adressage est identique au précédent. Les instructions sont codées sur 4 octets, les 2 premiers déterminent le code opération. Les 3ème et 4ème octets correspondent à la valeur signée du déplacement (le déplacement est codé sur 16 bits : la valeur décimale est comprise entre -32 768 et +32 767).

Exemple : Comparer le contenu de l'accumulateur A avec le contenu de l'emplacement mémoire $2000. S'il y a égalité entre les deux contenus, exécuter un branchement à l'adresse $52C9 (étiquette SAUT), sinon continuer le programme. En assembleur il faut :

Donner une étiquette : SAUT EQU $52C9. Puis, écrire que le branchement doit se faire s'il y a égalité de deux contenus, après comparaison : CMPA $2000 BEQ SAUT

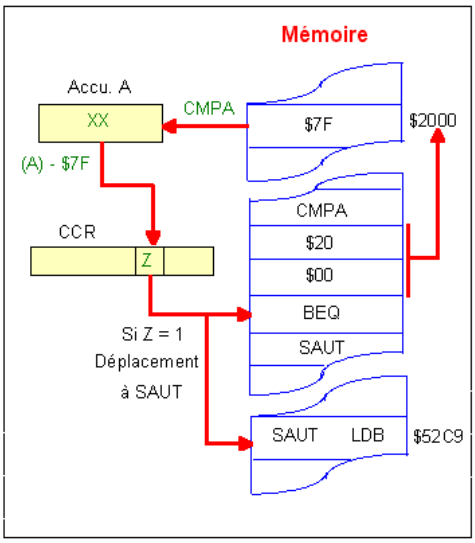


Figure n° 8 : Exemple d'utilisation du mode d'adressage relatif pour le µP 6809

→ **Adressage indexé** : Dans ce mode d'adressage, l'adresse effective/absolue de l'opérande s'obtient en faisant la somme entre un déplacement (offset) associé au code opération et une base qui est contenue dans l'un des registres suivants : *Index (X ou Y), Pointeur de pile (U et S) ou Compteur ordinal (PC).*

Adresse Effective = Base + Déplacement

Enfin ce mode d'adressage permet aussi la **pré-décrémentation** et la **post-incrémentation** simple ou double. Ca permet de travailler sur des emplacements mémoires adjacents, possibilités intéressantes pour le traitement des tables de données.

→ **Adressage indexé (Déplacement Nul)** : C'est le fonctionnement le plus simple de ce mode d'adressage. Le registre d'index contient l'adresse effective de l'octet à manipuler, Exemple :

LDA 0, X → Charge A avec le contenu de l'adresse pointée par la valeur de X.

LDA , X → Charge A avec le contenu de l'adresse pointée par la valeur de X.

→ **Adressage Indexé (avec auto-incrémentation)** : Dans ce mode, l'incrémentation du Registre d'Index aura lieu après l'exécution de l'opération, Exemple :

LDA , X+ Charge l'Accumulateur A avec le contenu de l'adresse $ 1000, mais IX contiendra $ 1001 après l'instruction terminée.

LDD , X++ Charge le Registre D avec les contenus des deux cases d'adresses adjacentes : $ 1000 & $ 1001 et incrémente deux fois IX après, donc IX = $ 1002.

→ **Adressage Indexé (avec auto-décrémentation)** : Dans ce mode, le contenu du Registre d'Index est décrémenté avant de pointer l'adresse effective, Exemple :

Si le Registre d'Index IX contient $ 1000, LDB,-X Commence par décrémenter le Registre IX. Donc IX = $ 0FFF et charge l'Accumulateur B avec le contenu de $ 0FFF

N.B *Le travail en auto-incrémentation/décrémentation simple facilite le traitement de tables de données sur 8 bits et le double de tables sur 16 bits.*

→ **Adressage Indexé (avec un déplacement constant)** : Le post-octet définit la puissance du déplacement par le positionnement de son bit **B7**.

B7 = 0 *Le déplacement est codé sur 5 bits (B0 - B4) et sera additionné en valeur algébrique au registre d'index avant le pointage de l'adresse effective.*

B7 = 1 *Le déplacement est stipulé par un ou deux octets qui suivront le post-octet.*

Bit du registre post-octet								*Mode d'adressage indexé*
7	6	5	4	3	2	1	0	AE = , Base (R) ± Déplacement
0	R	DÉPLACEMENT						AE = , R ± 4 bits
1	R	0	0	0	0	0		AE = , R +
1	R	1	0	0	0	1		AE = , R++
1	R	0	0	0	1	0		AE = , − R
1	R	1	0	0	1	1		AE = , − − R
1	R	1	0	1	0	0		AE = , R ± 0
1	R	1	0	1	0	1		AE = , R ± Acc B
1	R	1	0	1	1	0		AE = , R ± Acc A
1	R	1	1	0	0	0		AE = , R ± 8 bits
1	R	1	1	0	0	1		AE = , R ± 16 bits
1	R	1	1	0	1	1		AE = , R ± D (Acc A + Acc B)
1		1	1	1	0	0		AE = PC ± 7 bits
1		1	1	1	0	1		AE = PC ± 15 bits
1	R	1	1	1	1	1		AE = Adresse

Base R	*b6*	*b5*
Index X	0	0
Index Y	0	1
Pointeur U	1	0
Pointeur S	1	1
Compteur Programme		

→Indifférent, la sélection de la base PC se fait à l'aide des bits 2 et 3 (1, 1).

Tableaux n° 3 : Tableau de correspondance relatif au post-octet

Les bits B5 et B6 du post-octet désignent le registre devant servir d'index.

Dans le cas d'un déplacement sur 8 bits, les bits B0-B3 du post-octet vaudront % 1000 et dans le cas d'un déplacement sur 16 bits ils vaudront % 1001, exemples :

LDA 4, X → Charge A avec le contenu de l'adresse X+4

LDA $05, X+ cette instruction va **c**harger l'accumulateur A en mode indexé puis auto-incrémentation de 1, du contenu de X (voir la figure suivante).

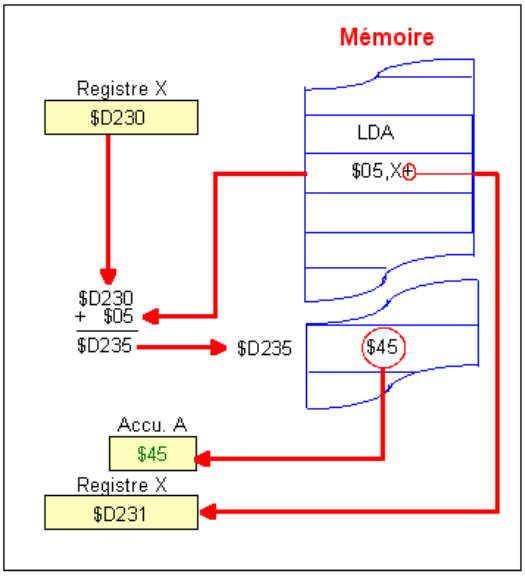


Figure n° 9 : Exemple d'utilisation du mode d'adressage indexé pour le µP 6809

CMPD - 5, U → Compare D avec le contenu des adresses U-5, U-4

→ **Adressage Indexé (avec un déplacement Accumulateur)** : Dans ce cas, c'est le contenu des accumulateurs A, B ou D qui est additionné au registre d'index avant de pointer l'adresse effective. Le bit B7 du post-octet est à 1, les bits B5-B6 précisent l'index et les bits B0-B3 précisent l'accumulateur concerné :

% 1010 ACCA
% 1001 ACC B
% 1011 ACCD

Ex : LDA A, Y → Additionne Y et A pour obtenir l'adresse effective où l'on doit trouver le nouveau contenu de A.

→ **Adressage Indexé (avec un PC comme base)** : Ce mode permet seulement un adressage indexé à déplacement constant sur 8 ou 16 bits. Les bits B5-B6 du post-octet n'ont plus de sens, ce sont les bits B2-B3 qui forcent ce mode de travail : **B2 = B3 = 1** → Index = PC

Le déplacement est précisé par un ou deux octets qui suivent le post-octet.

→ **Adressage Indexé indirect** : Ce mode d'adressage est une combinaison du mode indexé et du mode indirect. C'est le contenu de l'adresse pointée par l'index qui va servir d'adresse effective, exemple :

LDB [- 6.S] → Va chercher à l'adresse S-6, S-5 l'adresse effective à partir de laquelle on va charger B.

La notation « [ ] » précise le mode indirect et l'indexation offre les mêmes possibilités que le mode indexé simple.

4.2.3 Exemple de mise en application

Donnez l'organigramme et le programme Assembleur d'une routine permettant de programmer une temporisation de 1mn10s avec un signal d'horloge de fréquence égale à 1 Mhz.

Avec un signal d'horloge de fréquence 1 Mhz, 1 NC = 1µs. Pour 1mn10s : $70 / (1.10^{-6}) = 70\ 10^{6}$ NC

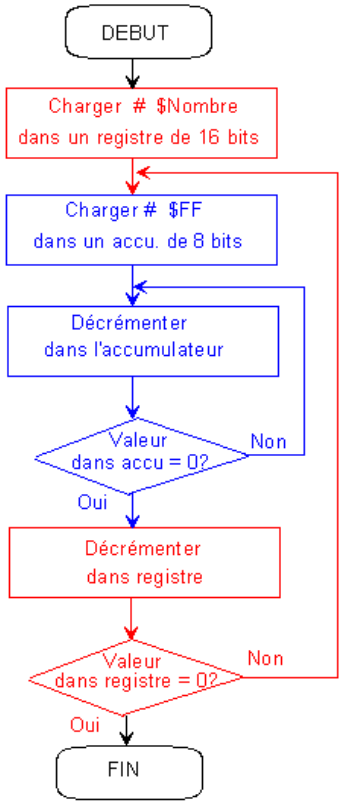


Figure n° 10 : Organigramme correspondant à la temporisation de 1mn10s

4.2.4 Conclusion

La multitude des combinaisons entre les 9 modes d'adressages et les 59 instructions de base du 6809, fait de ce microprocesseur le meilleur des microprocesseurs de 8 bits sur le marché.

4.3 Le microprocesseur 68000

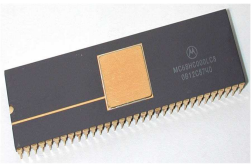
→ *Le microprocesseur 68000 est un processeur 32 bits,*

→ *Il est mis sur le marché en 1980.*

→ *La famille des processeurs 68000 comprend également les processeurs 32 bits 68020, 68030, 68040 ainsi que de nombreux types de micro-contrôleurs comprenant un processeur 68000 amélioré (micro-contrôleurs 68331, 68332, 68328, 68360).*

4.3.1 Architecture du µP 68000

Il existe, sur le marché, plusieurs variantes de circuits 68000 (voir un échantillon ci-dessous).



Motorola MC68HC000LC8, sa fréquence est de : 8 MHz

Motorola MC68HC001CFN8, sa fréquence est de : 8 MHz

Motorola MC68EC000FN10, sa fréquence est de : 10 MHz

Hitachi HD68000Y10, sa fréquence est de : 10 Mhz

Thomson TS68000CFN16, sa fréquence est de : 16 Mhz

Figure n° 11 : Quelques formes de circuits 68000

→ *Les données sont stockées sur 8 bits (un octet), 16 bits ou sur 32 bits.*

→ *Les adresses sont définies sur 32 bits, avec 24 lignes d'adresse sortant du microprocesseur. L'espace d'adressage est donc de 2^{24} octets, c'est-à-dire de 16 Moctets.*

Le microprocesseur comporte différents registres (voir la figure suivante) :

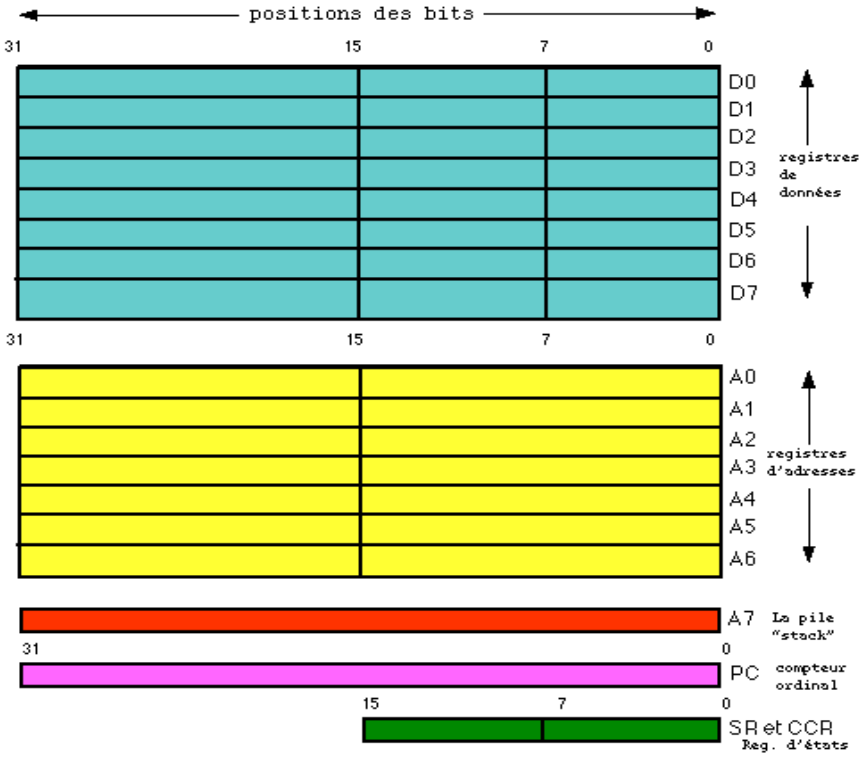


Figure n° 12 : Les différents registres de la famille 68000

→ Un *compteur ordinal* (PC),

→ Huit registres de données D_0, D_1, …, D_7. On peut accéder au registres *de données* sur :

8 bits (octet, byte, bits $d_7 ... d_0$),

16 bits (doublet, word, bits $d_{15} ... d_0$),

32 bits (quadlet, longword, bits $d_{31} ... d_0$).

Un accès sur un seul bit parmi 32 bits est également possible. Les accès aux registres sont prévus pour manipuler des données et effectuer des opérations logiques et arithmétiques.

→ 8 registres d'adresse A_0, A_1, …, A_7 de 32 bits.

→ Un registre spécial, le registre *de fanions F*, contient les fanions.

→ Les registres d'adresse sont supposés être utilisés comme *pointeurs* sur des positions mémoires. Ils contiennent en général l'adresse d'une position mémoire.

→ Le registre A_7 joue un rôle particulier car il est employé comme *Pointeur de Pile.*

→ Le *compteur ordinal* (PC) donne l'adresse de l'instruction à rechercher en mémoire.

→ Le registre *fanions (F)* contient les fanions principaux C, X, V, N, Z, …

4.3.2 Les instructions du µP 68000

Instructions de chargement

Move.8 #VAL, D_0 **[N, Z, V = 0, C = 0, X = 0]**	*Chargement de D_0 par une valeur immédiate, càd une valeur qui se trouve dans l'instruction. Exemple :* VAL = $ 1B ; définition du symbole VAL Move.8 #VAL, D_0 ; transfère la valeur hexadécimale 1B dans D_0

Move.8 D_1, D_0 **[N, Z, ...]**	*Chargement de D_0 par une valeur se trouvant dans D_1*

Move.8 {A_0}, D_0 **[N, Z, ...]**	*Chargement de D_0 par une valeur se trouvant à une position mémoire pointée par le registre A_0*

Move.8 ADMEM, D_0 **[N, Z, ...]**	*Chargement de D_0 par une valeur se trouvant à la position mémoire ADMEM (ADMEM est une adresse symbolique qui doit qui être définie dans le programme).* Exemple : ADMEM = $ 0700 ; définition de l'étiquette ADMEM Move.8 ADMEM, D_0 ; transfère la valeur qui se trouve à la position mémoire ADMEM = $ 0700 dans D_0

Move.8 D_0, {A_1} **[N, Z, ...]**	*Chargement d'une position mémoire pointée par le registre A_1 par une valeur se trouvant dans le registre D_0*
Move.8 D_0, ADMEM **[N, Z, ...]**	*Chargement d'une position mémoire ADMEM par une valeur se trouvant dans le registre D_0. Si ADMEM est spécifiée sur 16 bits, les bits de poids forts de l'adresse sont obtenus par extension du signe, càd que les bits 2^{16} à 2^{32} prennent la même valeur que le bit de poids 2^{15}*

Les mêmes instructions permettent d'effectuer des transferts sur 16 ou 32 bits :

Move.16 #VAL, D_3 **[N, Z, ...]**	*Chargement de D_3 par une valeur immédiate 16 bits (la valeur fait partie de l'instruction). Les bits de poids fort de D_3 conservent leur valeur.*

Move.32 D_3, D_5 **[N, Z, ...]**	*Chargement du registre D_5 par une valeur (32 bits) se trouvant dans D_3*

→ Les instructions de *post-incrémentation* et de *pré-décrémentation* permettent de combiner le transfert d'opérande et l'incrémentation, respectivement la décrémentation du registre d'index. L'incrémentation, respectivement la décrémentation, requiert l'addition, respectivement la soustraction, de 1, 2, ou 4 à la valeur du registre d'index, en fonction de la taille de l'opérande (8 bits, 16 bits ou 32 bits).

Move.8 D_0, {A_1 +} **[N, Z, ...]**	*Chargement du contenu de D_0 à la position mémoire pointée par le registre A_1, puis incrémentation de A_1 (+1. car un seul octet a été transféré).*

Move.16 D_3, {- A_1} **[N, Z, ...]**	*Décrémentation du registre d'index A_1 par soustraction de 2, puis transfert du contenu du D_3 à la position mémoire pointée par la nouvelle valeur du registre d'index A_1.*

Move.32 {A_1 +}, D_4 **[N, Z, ...]**	*Transfert du contenu de la position mémoire pointée par A_1 dans le registre D_4, ensuite ajout de 4 à la valeur de A_1 (4 car la taille de l'opérande transférée est de 4 octets).*

***N.B** : L'instruction **PUSH.16 D_0**, équivalente à **MOVE.16 D_0, {-A_7}**, permet de sauver le contenu de **D_0** (une valeur 16 bits) sur la pile. L'instruction **POP.16 D_0** équivalente à **MOVE.16 {A_7 +}, D_0** permet de récupérer cette valeur de la pile et de la transférer dans **D_0**.*

→ Les instructions pour opération arithmétiques : Les instructions *arithmétiques et logiques* permettent d'effectuer une opération arithmétique ou logique entre *l'opérande source* et *l'opérande destination* et de stocker le résultat à la position de *l'opérande destination.*

Les instructions *arithmétiques et logiques* permettent d'effectuer une opération arithmétique ou logique entre *l'opérande source* et *l'opérande destination* et de stocker le résultat à la position de *l'opérande destination.*

ADD.8 #VAL, D_0 **[N, Z, V = 0, C = X]**	*Addition d'une valeur immédiate et de la valeur contenue dans D_0. Le stockage du résultat sera fait dans le registre de D_0*

ADDX.8 D_1, D_0 **[N, Z, V = 0, C = X]**	*Addition, en tenant compte du fanion de report (eXtension) de la valeur contenue dans le registre D_1 et de celle contenue dans le registre D_0. Le stockage du résultat sera fait dans le registre de D_0. Si le fanion (eXtension)) était actif avant l'addition, l'opération donne le résultat suivant : $D_1 + D_0 \rightarrow D_0$*

SUB.8 #VAL, D_0 **[N, Z, V = 0, C = X]**	*Soustraction de la valeur immédiate #VAL, de celle contenue dans le registre D_0. Le stockage du résultat sera fait dans le registre de D_0*

SUB.8 {A_2}, D_1 **[N, Z, V = 0, C = X]**	*Soustraction, en tenant compte du fanion d'empreint (eXtension), de la valeur se trouvant à la position mémoire pointée par {A_2} de celle contenue dans le registre D_1 et de celle contenue dans le registre D_0. Le stockage du résultat sera fait dans le registre de D_1. Si le fanion d'empreint (eXtension)) était actif avant la soustraction, l'opération est la suivante :* $D_1 - (\{A_2\}) - 1 \rightarrow D_1$

COMP.16 D_3, D_4 **[N, Z, V = 0, C = X]**	*Comparaison de la valeur se trouvant dans D_3 et de celle contenue dans le registre D_4. Il s'agit d'une opération similaire à la soustraction, mais seuls les fanions sont mis à jour. La valeur se trouvant dans D_4 n'est pas modifiée.*

ADD.16 D_3, D_0 **[N, Z, V = 0, C = X]**	*Addition 16 bits de la valeur se trouvant dans D_3 et de celle se trouvant dans D_0. Le stockage du résultat sera fait dans le registre de D_0.*

INC.32 D_1 **[N, Z, V = 0, C = X]**	*Incrémentation de la valeur du registre $D_1 + 1 \rightarrow D_1$ (sur 32 bits)*

DEC.16 D_1 **[N, Z, V = 0, C = X]**	*Décrémentation de la valeur du registre $D_1 - 1 \rightarrow D_1$ (sur 16 bits). Les 16 bits de poids forts restent identiques.*

DEC.8 D_2 **[N, Z, V = 0, C = X]**	*Décrémentation de la valeur du registre $D_2 - 1 \rightarrow D_2$ (sur 8 bits). Les 24 bits de poids forts restent identiques.*

Instructions pour opérations logiques

AND.8 D_1, D_0 **[N, Z, V = 0, C = 0]**	*ET logique bit à bit (sur 8 bits) entre l'opérande se trouvant dans le registre D_1 et de celle contenue dans le registre D_0. Le résultat est stocké dans le registre de D_0.*

OR.16 D_2, D_3 **[N, Z, V = 0, C = 0]**	*OU logique bit à bit (sur 8 bits) entre l'opérande se trouvant dans le registre D_2 et de celle contenue dans le registre D_3. Le résultat est stocké dans le registre de D_3.*
XOR.8 D_1, D_0 **[N, Z, V = 0, C = 0]**	*OU EXCLUSIF logique bit à bit (sur 8 bits) entre l'opérande se trouvant dans le registre D_1 et de celle contenue dans le registre D_0. Le résultat est stocké dans le registre de D_0.*
NOT.8 D_1 **[N, Z, V = 0, C = 0]**	*Inversion logique bit à bit (sur 8 bits) entre l'opérande se trouvant dans le registre D_1. Le résultat est stocké dans le registre de D_1.*

Instructions de décalage et de rotation

Les opérations de décalage entraînent une perte d'information : les bits de poids faibles ou forts, selon la direction de décalage, sont perdus.

Les opérations de rotation présentent l'information en réinjectant les bits de poids forts sur les poids faibles et vice-versa (voir la figure ci-dessous) et les tableaux suivants.

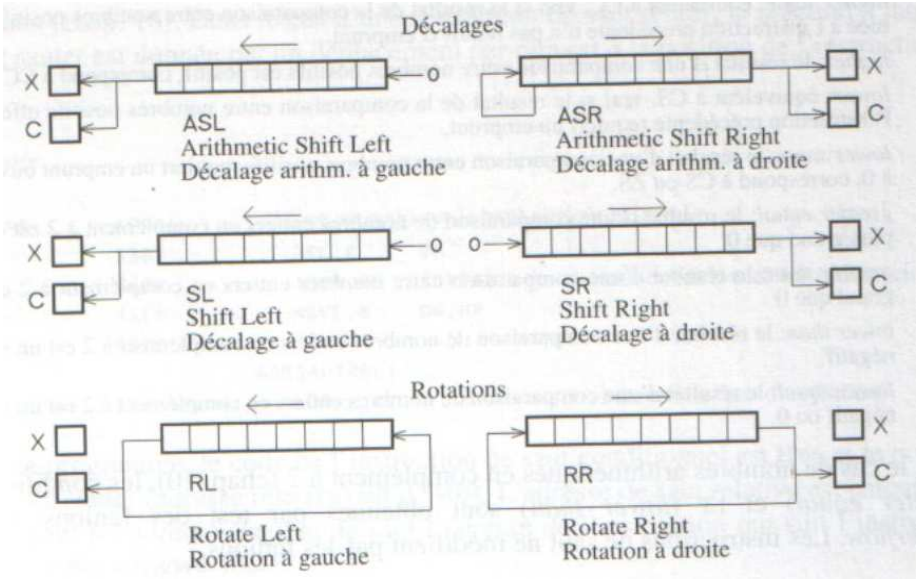


Figure n° 13 : Les différents registres à décalage de la famille 68000

RR.8 D_1 **[N, Z, C, V = 0]**	*Rotate Right : rotation à droite de la valeur du registre D_1. Le bit de poids faible devient le bit de poids fort du résultat. Tous les autres bits sont décalés d'une position à droite.*

RL.8 D_2 **[N, Z, C, V = 0]**	*Rotate Left : rotation à gauche de la valeur du registre D_2. Le bit de poids fort devient le bit de poids faible du résultat. Tous les autres bits sont décalés d'une position à gauche.*

SL.8 #VAL, D_1	*Décalage à gauche selon le nombre de position indiqué par la valeur immédiate VAL.*

SR.8 #VAL, D_1	*Décalage à droite selon le nombre de position indiqué par la valeur immédiate VAL.*

ASR.8 #VAL, D_1	*Décalage à droite selon le nombre de position indiqué par la valeur immédiate VAL. Les bits de poids forts réinjectés à gauche sont identiques au bit de poids fort avant le décalage (extension du bit de signe).*

Instructions de saut

Les instructions de saut permettent d'effectuer des débranchements à l'intérieur des programmes. Il peut s'agir de sauts inconditionnels à une adresse donnée, de sauts conditionnels, d'appels de procédure et d'instructions de retour de procédure. Les instructions de saut conditionnel utilisent le contenu des fanions pour décider si le branchement doit avoir lieu ou pas. Le contenu courant des fanions dépend en général des résultats provenant de l'exécution de l'instruction précédente. Les mnémoniques suivantes sont définies pour le test direct des fanions du processeur 68000.

Codes	Signification en Anglais	Signification en Français
ZS	Zero Set	Fanion Zéro actif
ZC	Zero Clear	Fanion Zéro inactif
CS	Carry Set	Fanion Carry actif
CC	Carry Clear	Fanion Carry inactif
NS	Negative Set	Fanion Négatif actif
NC	Negative Clear	Fanion Négatif actif
XS	eXtension Set	Fanion eXtension actif
XC	eXtension Clear	Fanion eXtension actif
VS	oVerflow Set	Fanion oVerflow actif
VC	oVerflow Clear	Fanion oVerflow actif
EQ	Equal	Equivalent à ZS, actif si la comparaison effectuée à l'instruction précédente montre que les deux opérandes sont égaux.
NE	Not Equal	Equivalent à ZC.
HS	Higher Same	Equivalent à CC, vrai si le résultat de la comparaison entre nombres positifs effectuée à l'instruction précédente n'a pas requis d'emprunt.
HI	Higher	Le résultat d'une comparaison entre nombres positifs est positif, correspond à CC et ZC
LO	Lower	Equivalent à CS, vrai si le résultat de la comparaison entre nombres positifs effectuée à l'instruction précédente requiert un emprunt.
LS	Lower Same	Le résultat d'une comparaison entre nombres positifs requiert un emprunt ou est égal à 0, correspond à CS ou ZS.
GE	Greather Equal	Le résultat d'une comparaison de nombres entiers en complément à 2 est égal ou plus grand que 0.
GT	Greather Than	Le résultat d'une comparaison de nombres entiers en complément à 2 est plus grand que 0.
LT	Lower Than	Le résultat d'une comparaison de nombres entiers en complément à 2 est un nombre négatif.
LE	Lower Equal	Le résultat d'une comparaison de nombres entiers en complément à 2 est un nombre négatif ou 0.

Tableau n° 4 Les instructions de saut

Dans le cas de nombres arithmétiques en complément à 2, les conditions GE *(Greather Equal)* et LT *(Lower Than)* sont obtenues par test des fanions ***N****égatif* et *o****V****erflow*. Les instructions de saut ne modifient pas les fanions.

Instructions	Signification
JUMP ETIQ_REL	Saut inconditionnel à l'adresse symbolique ETIQ_REL
JUMP {A_0}	Saut inconditionnel à à la position mémoire pointée par le registre A_0
JUMP, ZS ETIQ_REL	Saut conditionnel *(Zero Set)* à l'adresse symbolique ETIQ_REL. Le saut ne s'effectue que si le fanion zéro est actif *(zéro set).*
JUMP, EQ ETIQ_REL	Saut conditionnel *(Equal)* à l'adresse symbolique ETIQ_REL. Le saut ne s'effectue que si le fanion zéro est actif (*Zéro Set = Equal*). Les conditions ZS et EQ sont deux représentations différentes du fanion zéro actif.
JUMP, GE ETIQ_REL	Saut conditionnel *(Greather Equal)* à l'adresse symbolique ETIQ REL. Le saut ne s'effectue que si, après comparaison de nombres en complément à 2, le résultat est un nombre positif ou égal à zéro.
CALL ETIQ_REL	Appel de la routine se trouvant à la position mémoire donnée par ETIQ REL. La valeur courante du *PC* est sauvée sur la pile. Le programme effectue donc un saut a l'adresse donnée par ETIQ REL.
RET	*Retunr* : Retour de la routine appelée à la routine appelante, en rechargeant dans le PC la valeur qui avait été précédemment sauvée sur la pile. Le programme effectue donc un saut a l'adresse de retour sauvée sur la pile.

Tableau n° 5 Les instructions de saut (Suite)

Les instructions de saut peuvent spécifier une adresse de saut absolue ou relative. En utilisant systématiquement des sauts relatifs, on peut créer des programmes *relogeables*, c'est-à-dire des programmes exécutables qui peuvent être placés à n'importe quelle position mémoire. Ceci est particulièrement important sur les systèmes multitâches qui permettent de charger en mémoire et d'exécuter plusieurs programmes indépendants. Dans le cas d'une instruction de saut relatif, la position à laquelle il faut sauter est donnée par un déplacement par rapport à la position de l'instruction de saut.

5° Etude de certains circuits d'interface

Nous allons nous consacrer dans cette partie à certains composants fondamentaux des systèmes microinformatiques : il s'agit des circuits périphériques appelés aussi circuits d'interface. Cette classe de composants permet les échanges d'informations ou de données entre un microprocesseur et les différents périphériques : clavier, écran, imprimante, lecteurs de disquettes ou de disques, ... (voir la figure n° 1 donnée plus haut). Selon le processus de transmission des informations entre le µP et le(s) périphérique(s) associé(s), on distingue les coupleurs parallèles et les coupleurs séries.

Parmi les circuits périphériques que nous allons étudier dans ce qui suit, nous pouvons citer : le coupleur d'entrées/sorties parallèles (portant les codes : 6820/6821/6822) de la famille Motorola (le circuit d'interface parallèle de la famille Intel porte le code : 8255), le transmetteur monodirectionnel (dont le code est : 74 LS 244), le transmetteur bidirectionnel (74 LS 245), le circuit de temporisation (74 LS 373), le comparateur numérique (74 LS 688) et le multiplexeur analogique à huit voies (4051).

5.1 Le Coupleur d'E/S parallèle de la famille Motorola

Le coupleur d'entrée/sortie parallèle est un circuit d'interface programmable qui porte aussi le nom d'adaptateur d'interface périphérique (ou PIA : Peripheral Interface Adapter). Le circuit est prévu pour être connecté à un bus de type 6800, 6802, 6809, ... et, de ce fait, ne nécessite aucun circuit d'adaptation autre qu'un éventuel décodage d'adresse pour placer le PIA dans l'espace mémoire adressable par le µP. Le MC 6820 fût le premier coupleur d'entrée sortie commercialisé par la firme Motorola. Il n'est plus fabriqué ces derniers temps car le MC 6821 le remplace en corrigeant quelques défauts. Le MC 6822 est identique au MC 6821 sauf au niveau des sorties qui peuvent débiter un courant plus important. Le PIA communique avec le µP à l'aide des signaux (compatibles TTL) représentés sur la figure ci dessous.

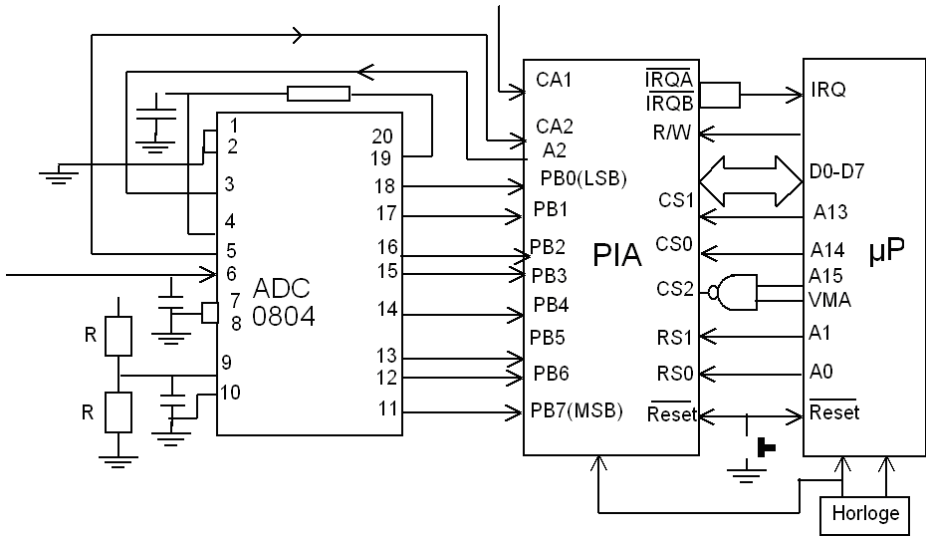


Figure n° 14 : Interconnexion du PIA avec les bus du µP et l'extérieur

Les différents signaux échangés entre le PIA et le µP sont :

→ D0 à D7 du PIA sont reliées à D0 à D7 du µP et servent à véhiculer les données.

→ E (Enable) du PIA est connectée à l'entrée d'horloge du µP. Elle joue le rôle d'entrée d'horloge pour le PIA.

→ R/W du PIA est reliée à la ligne R/W du µP. Elle sert pour gérer les opérations de lecture ou écriture entre le PIA et le µP.

→ Reset du PIA est connectée à la fois à l'entrée Reset du µP et au poussoir de Reset (ou à la circuiterie de remise à zéro automatique à la mise sous tension). Cette entrée permet la remise à zéro du PIA.

→ CS0, CS1 et CS2 sont des lignes de sélection du boîtier PIA qui ne sera activé que lorsque CS0 = CS1 = 1 et CS2 = 0. Ces lignes sont reliées à la circuiterie de décodage d'adresse du système afin de fixer l'adresse de base du PIA.

→ RS0 et RS1 permettent de sélectionner les registres internes du PIA selon le tableau ci-dessous. Généralement, RS0 est reliée à A0 et RS1 est reliée à A1, ce qui place les quatre registres les uns à la suite des autres.

→ IRQA et/ou IRQB seront ou non reliées à la ligne IRQ du µP 6802 ou à IRQ, FIRQ ou NMI du µP 6809 selon que l'on souhaitera ou non que le PIA puisse générer des interruptions avec son côté A, ou B, ou les deux.

5.1.1 Description générale

Comme le schématise la figure suivante, le PIA est un composant électronique à 40 broches réalisé en technologie N-MOS. Ce circuit dispose côté « extérieur » de 16 lignes d'entrées/sorties programmables individuellement et indépendamment les unes des autres en entrées ou en sorties. Cette programmation se fait par logiciel et non par des mises à la masse ou au +5V de broches (ce qui implique qu'elle peut être changée de manière dynamique dans un programme). D'autre part, une ligne d'entrée/sortie peut jouer alternativement le rôle d'entrée et de sortie, comme nous le verrons lors des exemples d'utilisation par la suite.

Il dispose, de plus, de quatre lignes de dialogue pouvant générer des interruptions dont deux sont des entrées (CA1 et CB1), les deux autres étant programmables en entrée ou en sortie (CA2 et CB2).

Comme tous les circuits périphériques de la famille Motorola, le PIA est vu par le µP, comme quatre positions mémoires : le dialogue avec le circuit et avec ce qui est relié aux lignes d'entrées/sorties du PIA se fait donc par des lectures et des écritures mémoires aux quatre adresses correspondantes.

5.1.2 Structure interne

Le PIA est un système pratiquement symétrique comportant deux ports de communication appelés port « A » et port « B » (voir la figure suivante). Chaque port comprend 8 lignes programmables en entrée/sortie, et ceci une à une. Ces lignes s'appellent PA0 à PA7 pour le port « A » et PB0 à PB7 pour le port « B ».

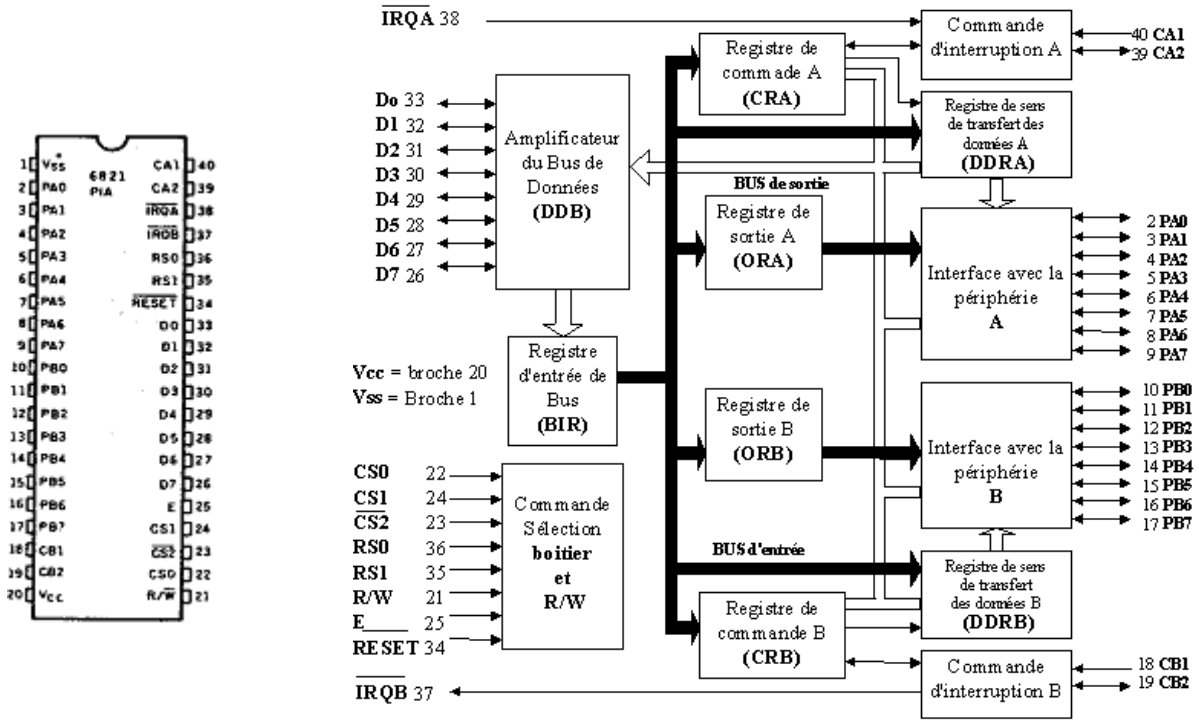


Figure n° 15 : Schéma synoptique interne du PIA

Le PIA possède six registres internes (voir la figure ci-dessus), soit trois registres par port :

→ DDRA Data Direction Register A : C'est un registre de direction de données pour le port « A ». Il contient le mot fixant le sens de transfert (en entrée ou en sortie) pour chacune des lignes de données. Un état « 1 » définit une broche en sortie et un état « 0 » définit une broche en entrée. Exemple :

→ ORA (Output Register A) ou registre de sortie pour le port « A » : Ce nom est assez mal choisi ; en effet, ce registre est celui dans lequel le µP viendra placer les données à faire sortir du PIA (là le nom est correct) mais c'est dans ce même registre que l'on viendra lire les données présentes sur celles des lignes PA0 à PA7 qui auront été programmées en entrée. En résumé, ORA est en fait un registre « image » des lignes PA0 à PA7.

→ CRA Control Register A : C'est un registre de contrôle permettant, pour le port « A », de définir le mode de fonctionnement des lignes d'interruption et de dialogue CA1 et CA2 ainsi que les possibilités de génération d'interruption via la ligne IRQA.

→ DDRB idem, mais pour le port « B ».
→ ORB idem, mais pour le port « B ».
→ CRB idem, mais pour le port « B ».

N.B. *Quand on effectue un Reset, le contenu de tous les registres du PIA est mis à 0, y compris le registre DDRA ou DDRB. Il en résulte que toutes les lignes se trouvent configurées en entrée.*

Le PIA est composé également :

→ Des Amplificateurs de bus bidirectionnels à trois états.
→ D'une ligne de contrôle interprétant les divers signaux de commande issus du bus du µP.
→ Des Amplificateurs d'interface pour les lignes PA0 à PA7 pour le port « A » et PB0 à PB7 pour le port « B ».
→ D'une ligne de contrôle qui à partir de CA1 et CA2 pour le port « A » et CB1 et CB2 pour le port « B » génère une interruption IRQA ou IRQB qui sera envoyée vers le µP.

5.1.3 Principe de fonctionnement

Toute la « science » du PIA repose sur la manipulation du contenu de ses six registres internes vus ci-dessus. Voyons maintenant quelles sont les adresses affectées au PIA et comment on les obtient en fonction de RS1 et RS0.

	Registre	Adresse	RS1	RS0
Port A	Direction	$ E480	0	0
	Données	$ E480	0	0
	Contrôle	$ E481	0	1
Port B	Direction	$ E482	1	0
	Données	$ E482	1	0
	Contrôle	$ E483	1	1

Bus d'adresses		MPU	A15 ..... A2			A1	A0			
			Logique de décodage							
		PIA	CS0	CS1	$\overline{CS2}$	RS1	RS0	CRA2	CRB2	*Adresses*
REGISTRES	A	CRA	1	1	0	0	1	—	—	ADR + 1
		DDRA	1	1	0	0	0	0	—	ADR
		ORA	1	1	0	0	0	1	—	ADR
	B	CRB	1	1	0	1	1	—	—	ADR + 3
		DDRB	1	1	0	1	0	—	0	ADR + 2
		ORB	1	1	0	1	0	—	1	ADR + 2

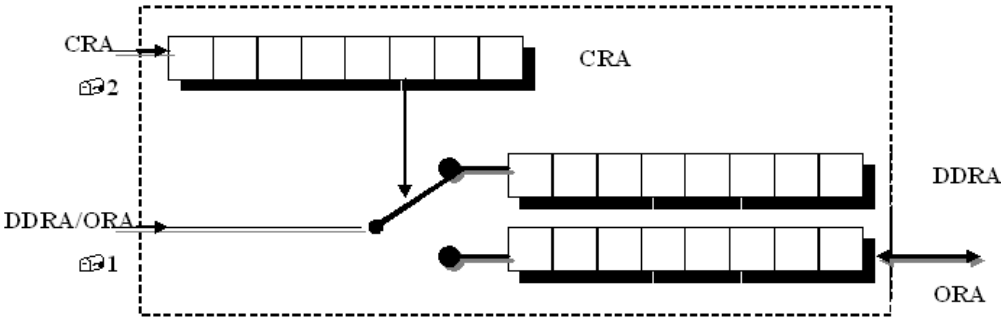
* An = ligne adresse du MPU.
* CSn = ligne sélection de boîtier.
* RSn = ligne sélection de registre.
* CRX = registre de contrôle.
* ADR = adresse de base résultante de la logique de décodage.

Figure n° 16 : Adresses mémoires réservées au PIA en fonction de RS0 et RS1

Voyons maintenant comment on peut accéder à ORA ou à DDRA puisque ces deux registres occupent la même adresse $E480. Pour se faire, reportons nous à la figure suivante, qui précise le rôle de chaque bit du registre de contrôle CRA ou CRB.

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	Contrôle de CA2			Accès à DDRA	Contrôle de CA1	
CRB	IRQB1	IRQB2	Contrôle de CB2			Accès à DDRB	Contrôle de CB1	

Exemple : Accéder au DDRA



Exemple : Accéder au ORB

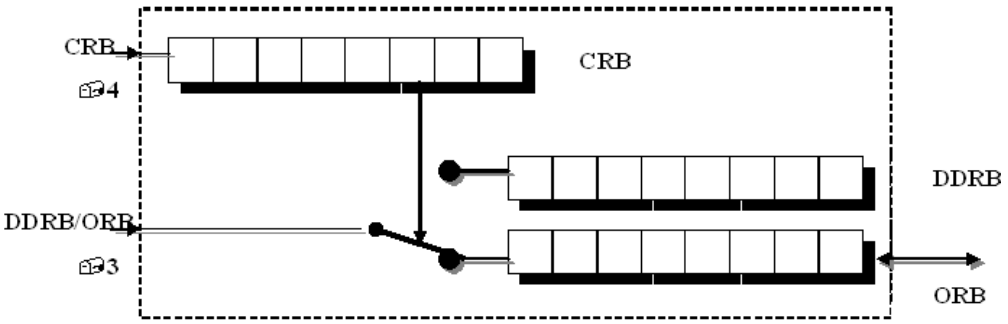


Figure n° 17 : Sélection de ses registres internes du PIA

Le bit c2 permet de sélectionner l'un des registres ORA ou DDRA : si c2 = 0, on accède au registre DDRA ; si ce bit est à 1, on accède au registre ORA.

N.B. *i) Le même raisonnement étant valable pour le port (B) du PIA.*
ii) Cette façon de faire permet d'une part de réduire l'espace mémoire occupé par le PIA (quatre emplacements au lieu de six) et d'autre part, gagner une broche au niveau de son boîtier (puisqu'il suffit de deux lignes RS0 et RS1 pour sélectionner les six registres, alors qu'il en faudrait trois sans cette astuce !).

5.1.4 Procédure d'initialisation du PIA

La procédure d'initialisation du PIA est généralement la suivante :

1°) On accède au registre de contrôle (CRA ou CRB) du port concerné et l'on y place $00 = (0000 0000), ce qui autorisera l'accès au registre (DDRA ou DDRB) puisque le bit « c2 » est à 0.

2°) On accède ensuite au registre de direction de données (DDRA Ou DDRB) où l'on écrit la configuration désirée.

3°) On accède de nouveau au registre de contrôle (CRA ou CRB) dans lequel on écrit le mot de contrôle correspondant à la fonction désirée et où le bit « c2 » est à 1 : cela permet ensuite d'accéder aux lignes d'entrées/sorties via le registre de sortie (ORA ou ORB) :

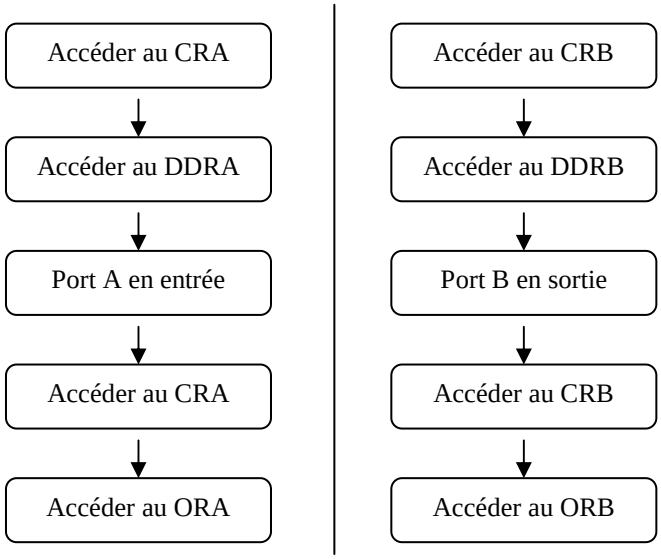
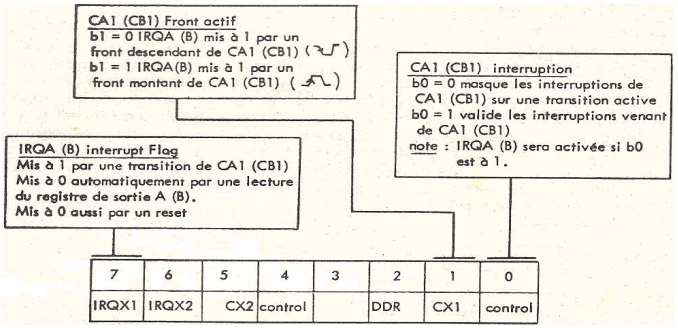


Figure n° 18 : Procédure d'initialisation du PIA

5.1.5 Etude détaillée du registre de contrôle

Nous allons voir, dans ce qui suit, au moyen de la figure n° 19 le rôle détaillé des bits des registres de contrôle CRA et CRB. Comme les deux moitiés du PIA sont « identiques », nous n'allons parler que du port « A » et d'autre part, nous allons passer en revue les bits un par un :



N°	CR^A_B c_1	CR^A_B c_0	transition active de l'entrée d'int. CA_1	indicateur d'interruption	sortie d'interruption $\overline{IRQA}$ (vers MPU)
1	0	0		mis à 1 par	interruption $\overline{IRQA}$ = 1 masquée
2	0	1		mis à 1 par	passe à 0 quand CRA₇ passe à 1
3	1	0		mis à 1 par	interruption $\overline{IRQA}$ = 1 masquée
4	1	1		mis à 1 par	passe à 0 quand CRA₇ passe à 1

Figure n° 19 : Fonction détaillée de chaque bit des registres de contrôle (partie n° 1)

→ « c0 », contrôle les interruptions en provenance de la ligne CA1 (voir la figure n° 19) :
Si « c0 » est à 0, les interruptions sont interdites.
Si « c0 » est à 1, les interruptions sont autorisées.

→ « c1 », sélectionne le front actif de CA1 : c'est à dire le front sur lequel cette entrée d'interruption va réagir :
Si « c1 » est à 0, le front actif sera descendant.
Si « c1 » est à 1, le front actif sera montant.

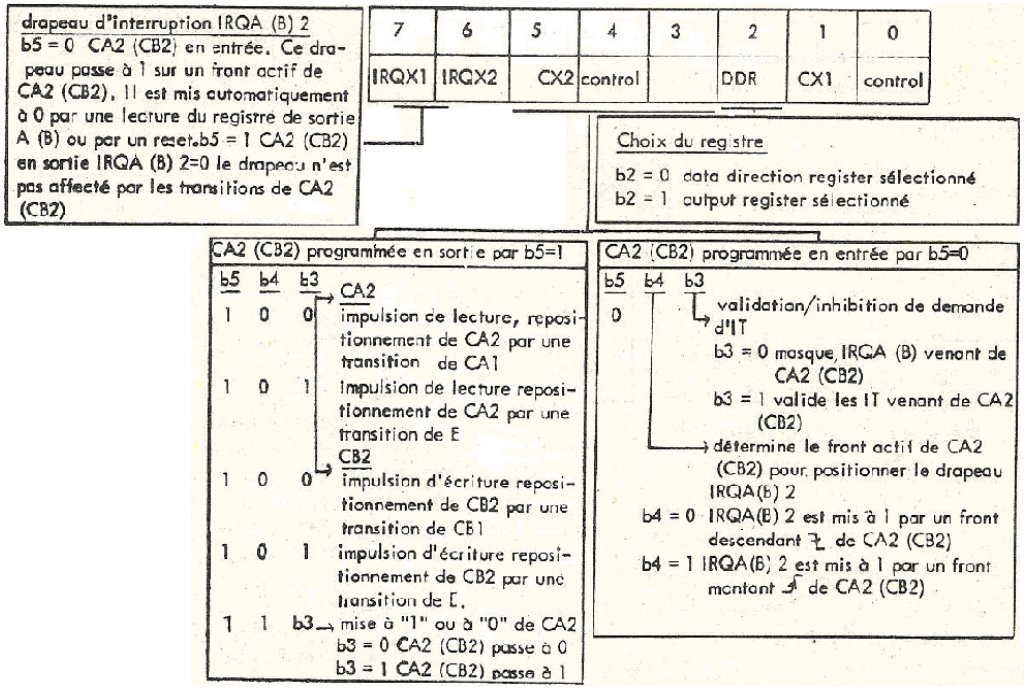


Figure n° 20 : Fonction détaillée de chaque bit des registres de contrôle (partie n° 2)

→ « c2 », sélectionne l'accès à l'un des deux registres ORA ou DDRA (comme il a été expliqué plus haut).

→ « c3 », validation des interruptions en provenance de CA2 :
Si « c3 » est à 0, les interruptions CA2 sont interdites ou masquées.
Si « c3 » est à 1, les interruptions CA2 sont autorisées.

N.B *Ceci est valable lorsque CA2 est en entrée ; lorsque CA2 est en sortie, «c3 » est l'image de CA2 : c'est à dire que si «c3 » est à 1, CA2 est à 1 et si «c3 » est à 0, CA2 est à 0.*

→ « c4 », sélectionne le front actif de CA2. Lorsque CA2 est en entrée, si « c4 » est à 0, le front actif sera descendant et si « c4 » est à 1, le front actif sera montant. Lorsque CA2 est en sortie, « c4 » sélectionne le fonctionnement de CA2 entre les modes dialogue et impulsionnel. Si « c4 » est à 1, le mode dialogue est choisi ; si « c4 » est à 0, le mode impulsionnel est choisi.

→ « c5 », sélectionne le fonctionnement de CA2 en entrée ou en sortie. Si « c5 » est à 1, CA2 est en sortie ; si « c5 » est à 0, CA2 est en entrée.

→ « c6 » est un bit d'état : c'est à dire qu'il ne peut qu'être lu par le microprocesseur. On l'appelle drapeau d'interruption associé à CA2. Si CA2 est en entrée, ce bit passe à « 1 » pour toute transition active de CA2 ; il est remis à « 0 » par une lecture du registre ORA ou par un Reset. Si CA2 est en sortie, ce bit est inactif et reste à « 0 ».

→ « c7 » est un bit d'état comme « c6 » : c'est le drapeau d'interruption associé à CA1. Il passe à « 1 » pour toute transition active de CA1 et remis à « 0 » par une lecture du registre ORA ou par un Reset.

N.B *Les deux bits « c6 » et « c7 » fonctionnent toujours, que les interruptions aient été autorisées par « c0 » et « c3 » ou non ! Lorsque les interruptions sont interdites, leur état n'est pas transmis aux lignes IRQA et IRQB.*

5.1.6 Mode de fonctionnement de la ligne d'interruption CA1

La programmation du mode de fonctionnement de la ligne d'interruption CA1 est décrite dans le tableau de la figure n° 20. Par exemple, si nous prenons la première ligne de ce tableau, lorsque les bits « c1 » et « c0 » de CRA sont à « 0 », une demande d'interruption est prise en compte sur un front descendant de CA1, l'indicateur d'interruption « c7 » associé à CA1, est mis à « 1 » et le PIA génère un signal de sortie d'interruption IRQA vers le µP.

L'indicateur d'interruption « c7 » est mis à « 0 » par une lecture du registre de données de la périphérie « A » par le µP.

Une demande d'interruption masquée par « c0 » égale à « 0 » est cependant mémorisée et devient active lorsque « c0 » passe à « 1 » ; IRQA est activée à « 0 ». L'illustration graphique des modes de fonctionnement des lignes d'interruption est donnée par la figure n° 21.

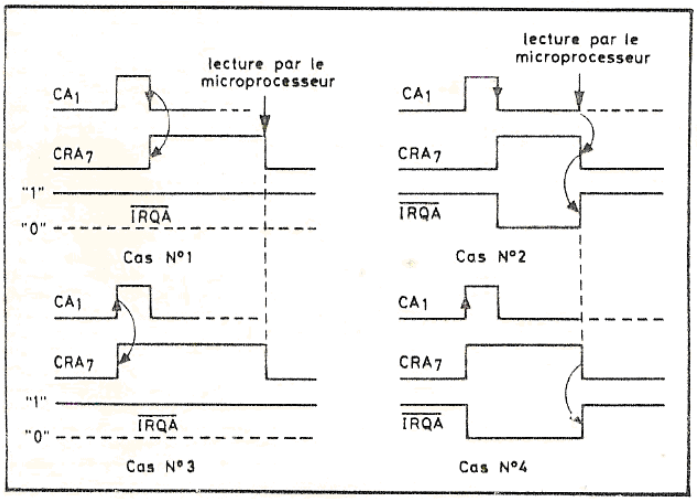


Figure n° 21 : Mode de fonctionnement de la ligne d'interruption CA1

5.1.7 Mode de fonctionnement de la ligne d'interruption CA2

La programmation du mode de fonctionnement de la ligne d'interruption CA2 mise en entrée est identique à CA1 si le bit « c5 » du mot de commande est à « 0 ». Dans ce cas, les bits « c3 », « c4 » et « c6 » jouent le même rôle que les bits « c0 », « c1 » et « c7 ».

La programmation de CA2 en sortie de commande s'obtient en écrivant un « 1 » dans « c5 ». Les bits « c4 » et « c3 » permettent de définir les modes d'actions de CA2.

Selon la programmation des bits « c4 » et « c3 », on distingue trois modes de fonctionnement qui sont : le mode programmé (Set/Reset), le mode de dialogue (handshake) et le mode impulsionnel (Pulse - Strobe). Pour ces deux derniers modes, la ligne CA2 sera associée à une lecture.

→ **Dans le mode programmé**, la sortie correspondant à la ligne CA2 suit l'état du bit « c3 » du registre CRA (voir la figure n° 22) :
Si « c3 » = 0 → CA2 = 0.
Si « c3 » = 1 → CA2 = 1.

Figure n° 22 : Fonctionnement de CA2 en mode programmé

N.B. *Il faut donc modifier le contenu du registre de contrôle CRA pour changer l'état de la ligne d'interruption CA2 !*

→ **Dans le mode impulsionnel**, la ligne d'interruption CA2 passe à l'état bas sur la première transition descendante de « E » (Enable) suivant un ordre de lecture du registre ORA. Elle repasse à l'état haut par le front descendant de la première impulsion « E » qui suit une désélection du PIA. Les chronogrammes de fonctionnement sont donnés en figure n° 23.

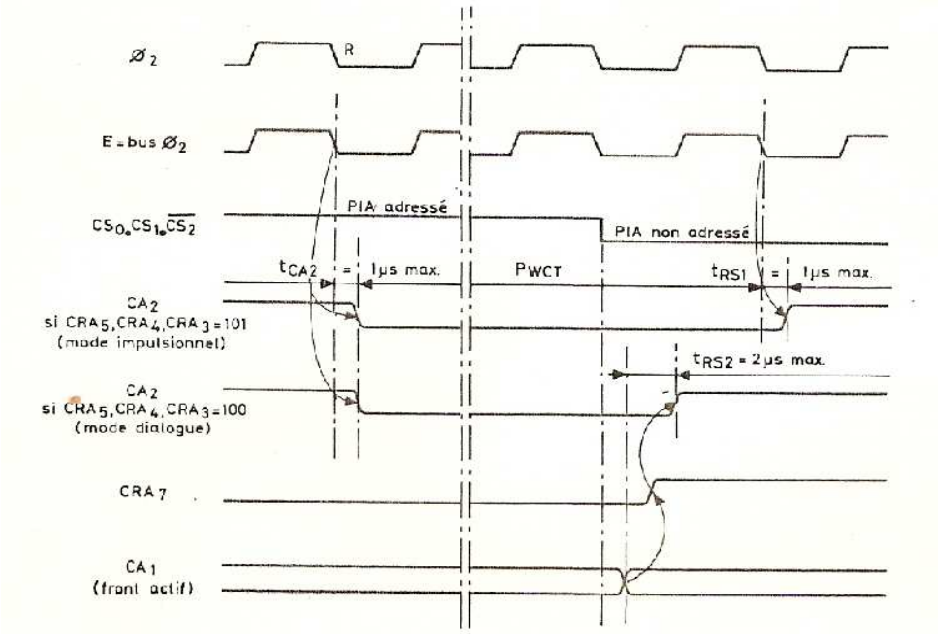


Figure n° 23 : Chronogrammes correspondant aux modes pulsionnels et dialogue de CA2 (qui est associée en lecture)

→ **Dans le mode de dialogue**, la ligne d'interruption CA2 passe à l'état bas sur la première transition descendante de « E » (Enable) qui suit un ordre de lecture du registre ORA. Elle est remis au niveau haut par l'indicateur d'interruption « c7 », lui même mis à un par le front actif de « c1 », ceci de façon asynchrone. Les chronogrammes de fonctionnement sont donnés en figure n° 23.

N.B *De manière conceptuelle, on a bien à faire à un dialogue entre le périphérique et le µP qui doit lire les données via le PIA.Dans ce mode de fonctionnement, le port est particulièrement adapté en entrée !*

5.1.8 Mode de fonctionnement de la ligne d'interruption CB1

Idem que pour le port « A ». On travaille sur le registre CRB !

5.1.9 Mode de fonctionnement de la ligne d'interruption CB2

De même que pour CA2, la programmation de cette ligne d'interruption en sortie de commande s'obtient en écrivant « c5 » du registre CRB à « 1 ». Mais les bits « c3 » et « c4 » permettent de définir des modes d'action de CB2 différentes de CA2. Contrairement à CA2, dans les modes impulsionnel et dialogue, CB2 sera associée à une écriture.

→ **Dans le mode programmé**, la sortie correspondant à la ligne CB2 suit l'état du bit « c3 » du registre CRB (voir la figure n° 24) :
Si « c3 » = 0 → CB2 = 0.
Si « c3 » = 1 → CB2 = 1.

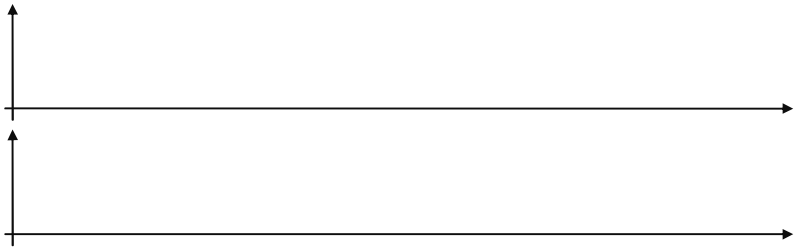


Figure n° 24 : Fonctionnement de CA2 en mode programmé

N.B. *Il faut donc modifier le contenu du registre de contrôle CRB pour changer l'état de la ligne d'interruption CB2 !*

→ **Dans le mode impulsionnel**, la ligne d'interruption CB2 est remise au niveau haut par le front montant de la première impulsion de « E » qui suit une désélection du PIA. Les chronogrammes de fonctionnement sont donnés en figure n° 25.

→ **Dans le mode de dialogue**, la ligne d'interruption CB2 est remise au niveau haut par l'indicateur d'interruption « c7 » du registre CRB, lui même mis à un par le front actif de « c1 », ceci de façon asynchrone. Les chronogrammes de fonctionnement sont donnés en figure n° 25.

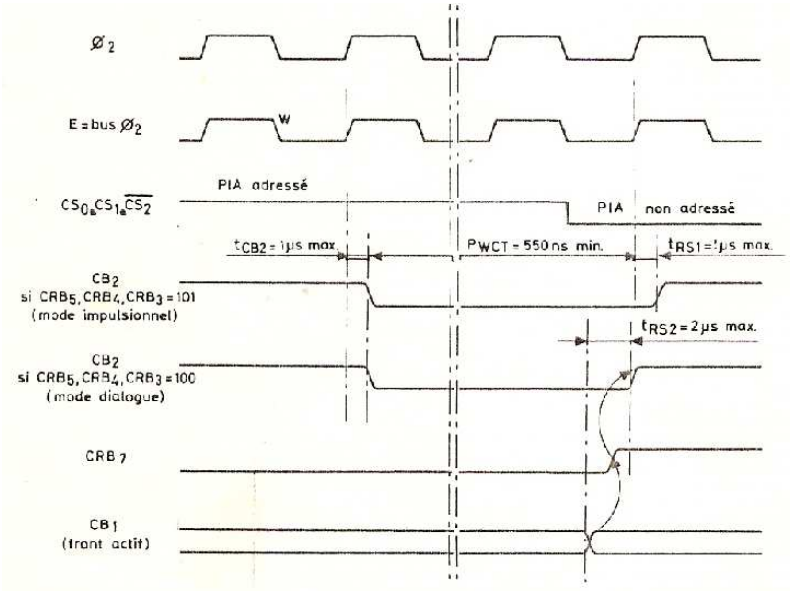


Figure n° 25 : Chronogrammes correspondant aux modes pulsionnels et dialogue de CB2 (associée en écriture)

5.2 Composants périphériques avec sorties à trois états

Nous avons vu plus haut que pour être utile, un µP doit pouvoir recevoir des informations du monde extérieur et communiquer le résultat des traitements qu'il effectue. Toutes ces communications se font par l'intermédiaire des ports d'E/S et différents bus. Dans cette partie du polycopier, nous étudierons certains composants électroniques à trois états. Ces derniers sont également très utiles pour réaliser des projets d'automatisation ou d'informatique industrielle.

Ces composants présentent la particularité de posséder outre un état bas et un état haut, un état dit « haute impédance ». Lorsque le niveau de l'entrée « commande » est haut, le composant se comporte comme un circuit normal (ici il s'agit de la fonction logique « Oui »), c'est à dire que sa sortie donne toujours le même niveau appliqué à son entrée : S = f(E) = E. Par contre, lorsque le niveau de « commande » est bas, la sortie se met à l'état appelé « haute impédance ».

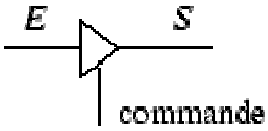


Figure n° 26 : Schéma et principe de fonctionnement d'un circuit à trois états

5.2.1 Le Transmetteur monodirectionnel à 3 états : 74LS244

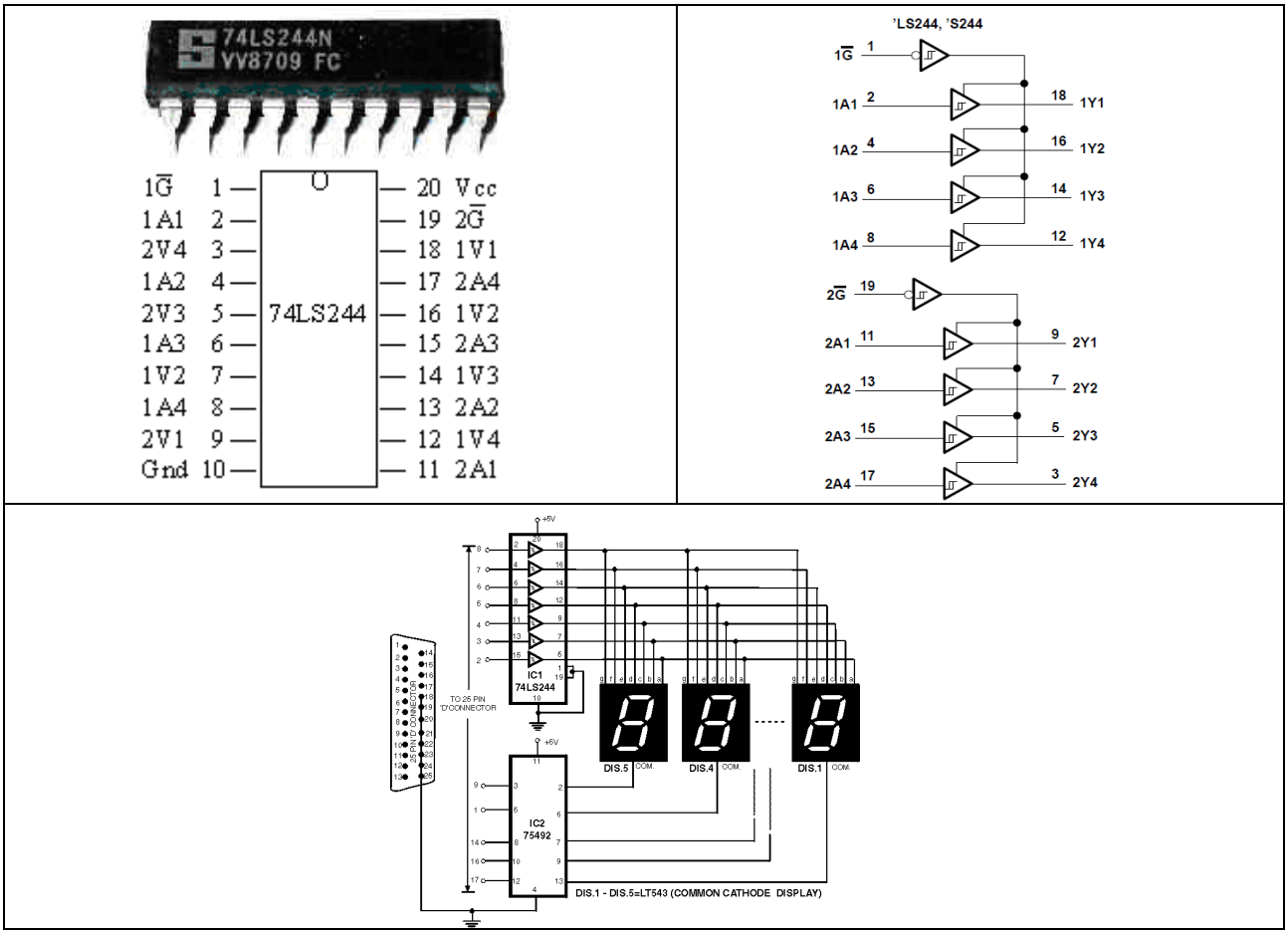


Figure n° 27 : Schéma de brochage et exemple d'utilisation d'un buffer monodirectionnel à trois états

Ces circuits sont spécialement conçus pour améliorer à la fois le fonctionnement et l'intégration des systèmes de commande d'adresses mémoires à haute impédance, de commande d'horloge et des émetteurs récepteurs de bus. Un niveau logique haut appliqué sur les entrées de validation « G1 » ou « G2 » place les sorties dans l'état de haute impédance. Ce circuit est disponible en boîtier de 20 broches. Le schéma de brochage et un exemple d'utilisation de ce circuit est donné par la figure n°27.

5.2.2 Le Transmetteur bidirectionnel à 3 états : 74LS245

Ces circuits, composés de huit émetteurs récepteurs, sont destinés aux communications asynchrones bidirectionnelles entre bus de données. Le niveau logique appliqué sur l'entrée de commande de direction « DIR » détermine le sens de transmission des données. Un niveau logique bas appliqué sur cette entrée permet la transmission des données du bus « B » vers le bus « A » et un niveau logique haut permet la transmission des données du bus « A » vers le bus « B ». L'entrée de validation « G » sert à déconnecter le boîtier de telle manière que les deux bus soient isolés. Ce circuit est disponible en boîtier de 20 broches. Son schéma de brochage est donné par la figure ci-dessous.

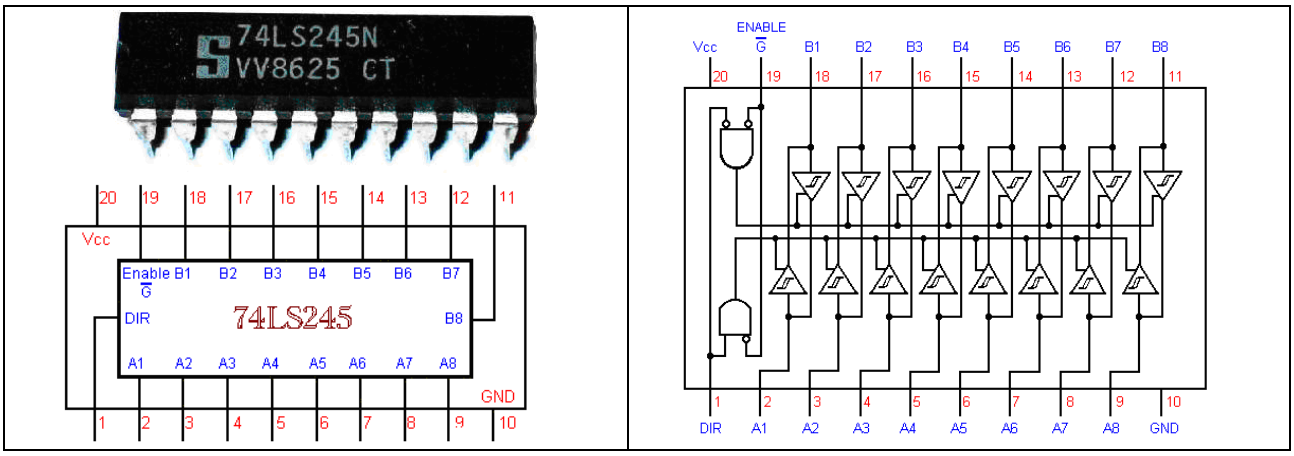


Figure n° 28 : Schéma de brochage du buffer bidirectionnel 74LS245 à trois états

5.2.3 Le circuit de temporisation à 3 états : 74LS373

C'est un registre composé de huit bits. Il possède des sorties à totem-pole haute impédance spécialement destinées à la commande de charges fortement capacitives ou à impédance relativement faible. Le troisième état de haute impédance et la commande au niveau logique haut permettent de connecter ce type de registres directement aux lignes de bus et de pouvoir les commander, dans les systèmes organisés, avec ce type de structure et ce, sans interfacage ni composants de rappel à la tension d'alimentation. Ils sont particulièrement intéressant dans la constitution de registres tampons, de ports d'E/S, de commande de bus bidirectionnels et de registres de travail.

Les huit verrous du circuit 74LS373 sont des verrous transparents de types bascules « D » ce qui signifie que lorsque l'entrée de validation « G » est au niveau logique haut, les sorties « Q » 'suivent' les entrées « D ». Lorsque cette entrée est mise au niveau logique bas, les sorties sont verrouillées et reflètent les niveaux logiques présents sur les entrées « D » avant la transition. Ce circuit est disponible en boîtier de 20 broches. Son schéma de brochage est donné par la figure ci-dessous.

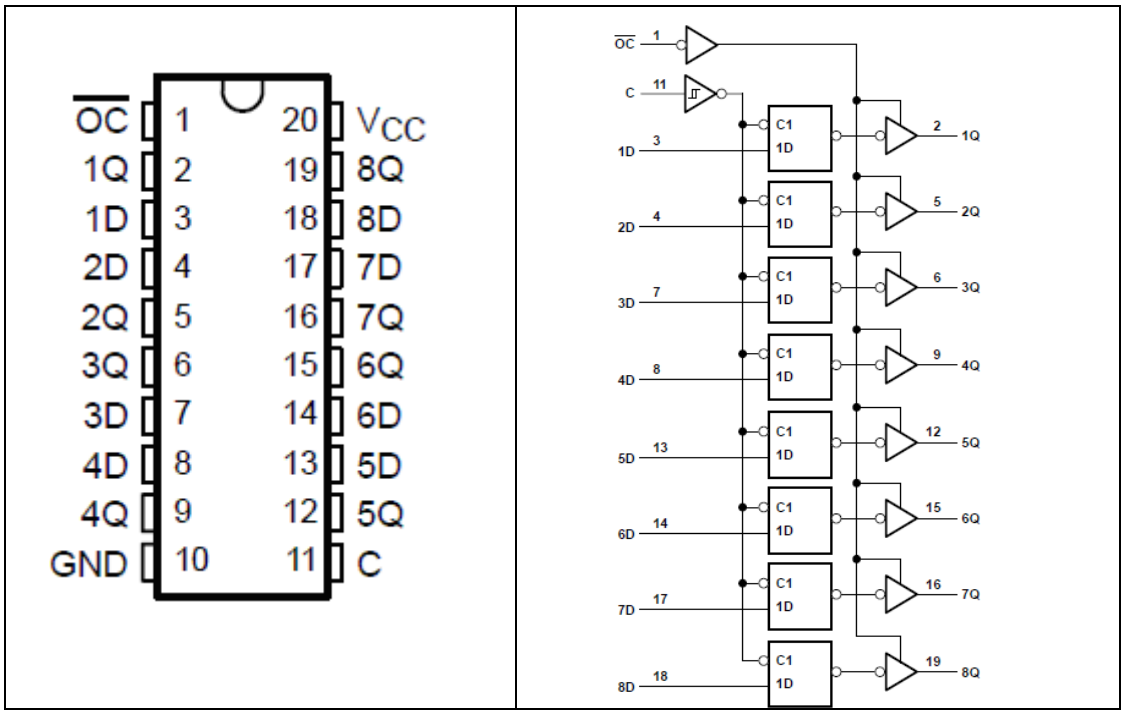


Figure n° 29 : Schéma de brochage d'un circuit de temporisation à trois états

5.3 Le comparateur numérique : 74LS688

Ce composant électronique permet de comparer 8 niveaux logiques (généralement présentés par les lignes d'adresses) à ceux fixés physiquement par l'opérateur (à l'aide des switchs par exemple). Une fois la comparaison est faite, la sortie de validation « 19 » délivre un signal logique qui peut être utilisé pour activer d'autres circuits. Le 74LS688 est disponible en boîtier de 20 broches. Son schéma de brochage est donné par la figure n° 30.

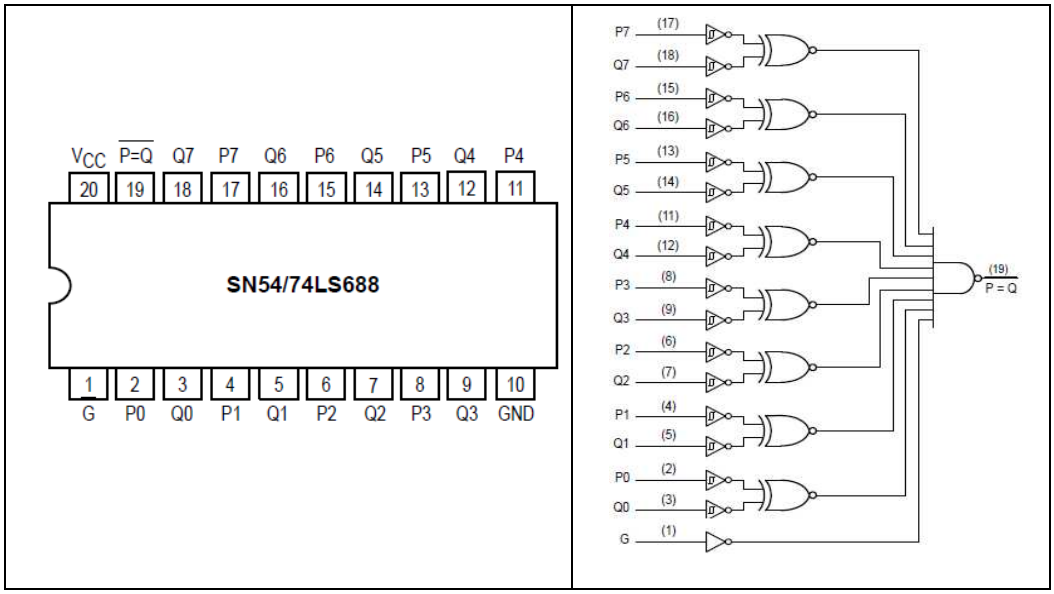
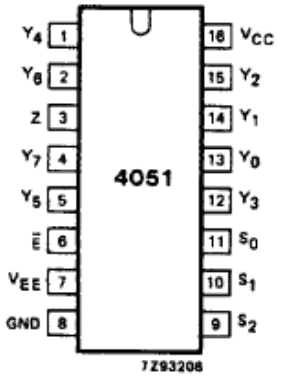
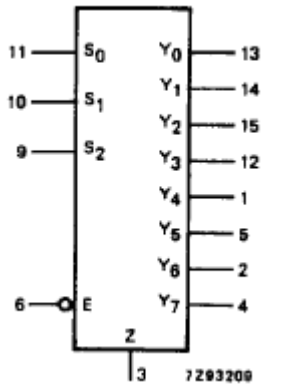


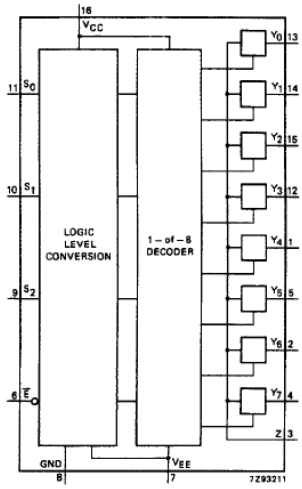
Figure n° 30 : Schéma de brochage d'un comparateur numérique

5.4 Le multiplexeur analogique à 8 voies : 4051

Le circuit « 4051 » est constitué de 8 commutateurs analogiques bidirectionnels dont un pôle est connecté à une entrée/sortie indépendante (Y0, ..., Y7) et l'autre pôle à une entrée/sortie commune « Z » (Figure suivante). L'entrée de validation « E » étant au niveau logique bas, un des commutateurs sélectionné (basse impédance, étant 'passant') par les entrées d'adresses (S_0, S_1 et S_2). Si « E » est au niveau haut, tous les commutateurs sont en état de haute impédance (état bloqué) indépendamment des entrées d'adresses. Il est possible de commander des signaux analogiques d'amplitude supérieure à 15V crête à crête avec des signaux numériques d'amplitude comprise entre 3 et 15V. Par exemple, si V_{CC} = 5V, GND = 0 et V_{EE} = -5V, on peut commander des signaux analogiques d'amplitude comprise entre -5 et +5V avec des signaux numériques allant de 0 à +5V. La valeur limite des tensions d'alimentation est : V_{CC} - V_{EE} = 18 V.







INPUTS				channel ON
$\overline{E}$	S_2	S_1	S_0	
L	L	L	L	$Y_0 - Z$
L	L	L	H	$Y_1 - Z$
L	L	H	L	$Y_2 - Z$
L	L	H	H	$Y_3 - Z$
L	H	L	L	$Y_4 - Z$
L	H	L	H	$Y_5 - Z$
L	H	H	L	$Y_6 - Z$
L	H	H	H	$Y_7 - Z$
H	X	X	X	none

Figure n° 31 : Schéma de brochage du multiplexeur analogique à 8 voies

6° Etude des microcontrôleurs de la famille Motorola

Pour les applications industrielles, il est essentiel que la place prise par les circuits intégrés soit minimale. Chaque composant discret prend de la place sur le circuit imprimé, doit être monté, consomme de l'énergie et peut tomber en panne. On cherche donc à minimiser le nombre de puces d'une électronique de commande. L'idéal est de travailler avec une seule puce dont les signaux d'entrée-sortie sont directement connectés à l'électronique de mise en forme du signal.

A cette fin, les fabricants de puces (Motorola. Intel, Hitachi, Texas Instruments, etc.) ont créé des *microcontrôleurs* qui incorporent en plus du processeur les éléments nécessaires à la réalisation de

systèmes de commande complets. Il s'agit de mémoire morte, de mémoire vive, de ports d'entrée-sortie parallèles, de ports pour la communication par ligne série de logique de gestion du temps et des événements et d'interfaces spécialisées, par exemple des convertisseurs analogiques/numériques ou des interfaces qui engendrent des signaux à largeur d'impulsion variable pour la commande des moteurs.

→ *Les microcontrôleurs 4 bits servent essentiellement à des tâches simples. De tels microcontrôleurs sont par exemple utilisés au sein d'objets ménagers tels que des cuisinières, machines à laver ou aspirateurs.*

→ *Les microcontrôleurs 8 bits sont capables de répondre à des exigences plus élevées et sont utilisés pour la commande de dispositifs informatiques tels que des souris et des tablettes graphiques. Ils sont également utilisés pour la programmation de petits robots ainsi que pour l'acquisition de données (convertisseurs A/D, etc.).*

→ *Les microcontrôleurs 16/32 bits sont utilisés pour la commande de machines ou le contrôle de processus, lorsque les contraintes temps réel sont sévères ou lorsque les algorithmes de réglage nécessitent une puissance de calcul importante. Des variantes de microcontrôleurs avec canaux d'accès mémoire direct offrant un grand débit entre mémoire et entrées-sorties sont utilisés dans les applications multimédia et pour le contrôle d'imprimantes laser.*

6.1 Etude du microcontrôleur 8 bits « 68HC11 » ou « MC68HC11 »

Notons tout au début qu'il existe plusieurs versions différentes de microcontrôleurs 8 bits. Le microcontrôleur 68HC11 est basé sur une version améliorée du processeur 8 bits 6801 développé à la fin des années 1970 par Motorola. Il comprend, en plus du processeur, de la mémoire morte (jusqu'à 24 Kilo Octets), de la mémoire vive (256 Octets à 1 Kilo Octet), un compteur/rythmeur similaire à celui de la famille 68331, une ligne série, un sous-système de communication série, un convertisseur analogique-digital, un chien de garde (watchdog) et jusqu'à 6 ports d'entrée-sortie programmables 8 bits.

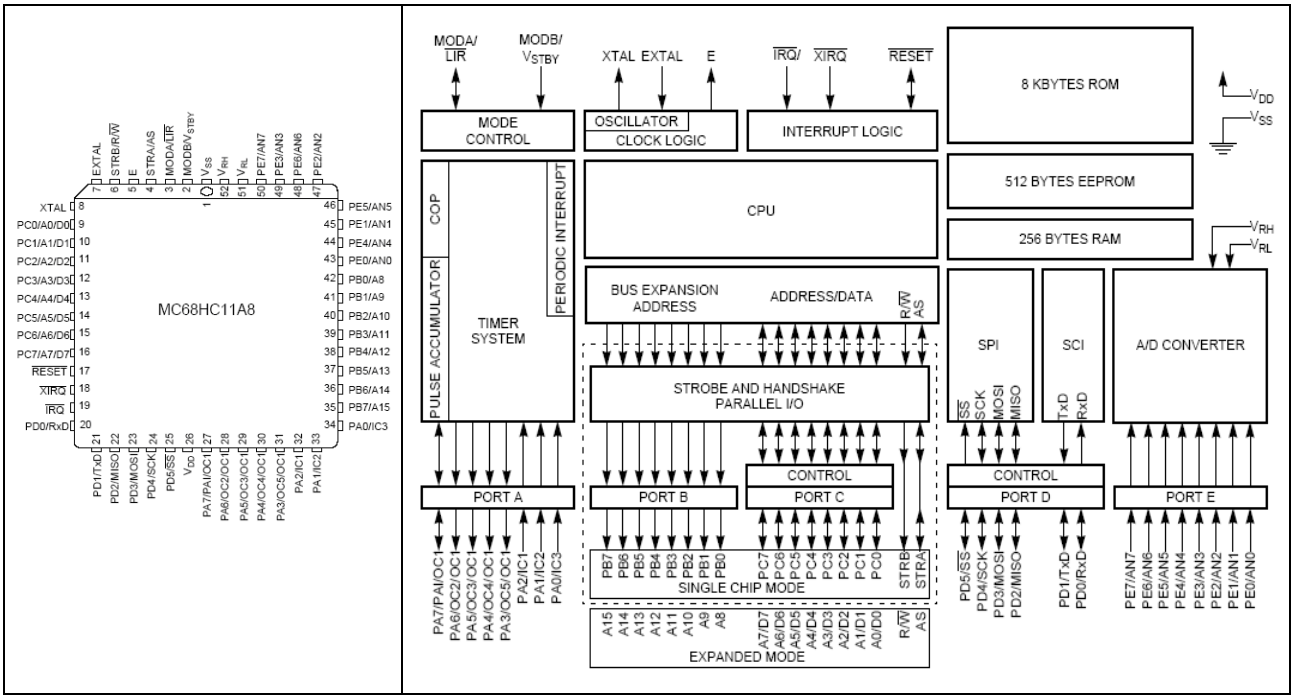


Figure n° 32 : Exemple d'architecture d'un microcontrôleur

La figure 32 donne l'architecture du microcontrôleur 68HC11 : les signaux du port « A » sont multiplexés avec les signaux des comparateurs et unités de capture pour la gestion du temps et des événements. Les signaux du port « D » sont multiplexés avec les signaux de l'interface série et du sous-système de communication série. Les signaux d'entrée analogiques multiplexés sur le convertisseur A/D sont également multiplexés avec les signaux d'entrée-sortie binaires du port « E ».

Grâce à l'augmentation de densité d'intégration sur une puce, on peut s'attendre à ce que la famille des microcontrôleurs 68HC11 continue à se développer et intègre d'avantage de mémoire vive et morte. Les familles qui succèdent au 68HC11 sont les microcontrôleurs HC12etHC16.

NB : Des périphériques supplémentaires peuvent soit être directement branchés sur les broches des ports d'entrée-sortie, soit être connectés au sous-système de communication série.

CHAPITRE I (TP)
LES MICROCONTROLEURS PIC

Les microcontrôleurs **PIC 16Cxx** disposent des principales ressources internes suivantes :

Mémoire de programme
Mémoire de données
ENTREE /SORTIE
Et éventuellement : Port série
CAN

DEUX STRUCTURES POUR LES SYSTEMES MICROPROGRAMMES

Architecture **Von Neumann:** PC, **6809**, 68HC11...	Architecture **Harvard:** **PIC**, DSP...
Dans le cas d' une architecture **Von Neumann**, le traitement d 'une instruction et son opérande nécessite donc la lecture d 'au moins **deux cases mémoires**.	Dans le cas de l' architecture **Harvard** que possèdent les PIC, la lecture d '**une seule case mémoire** permet le traitement entier d 'une instruction et de son opérande.

LES REGISTRES INTERNES DES PIC

Registre de travail : **W**
Registre d'E /S : **PORT**
Registre de direction : **TRIS**
Registre d'état : **STATUS**
Registre compteur programme : **PC**

En plus de bénéficier d 'une architecture dite **Harvard**, les microcontrôleurs **PIC** sont constitués autour d 'une architecture appelée **RISC**.

RISC = Circuit à jeu d'instructions réduit.Ainsi, contrairement à de nombreux circuits mettant en jeu une centaine d 'instructions différentes, les **PIC** voient leur nombre d 'instructions limités à **33 ou 35**.

5.2 Etude du microcontrôleur 16/32 bits : le « MC68331 »

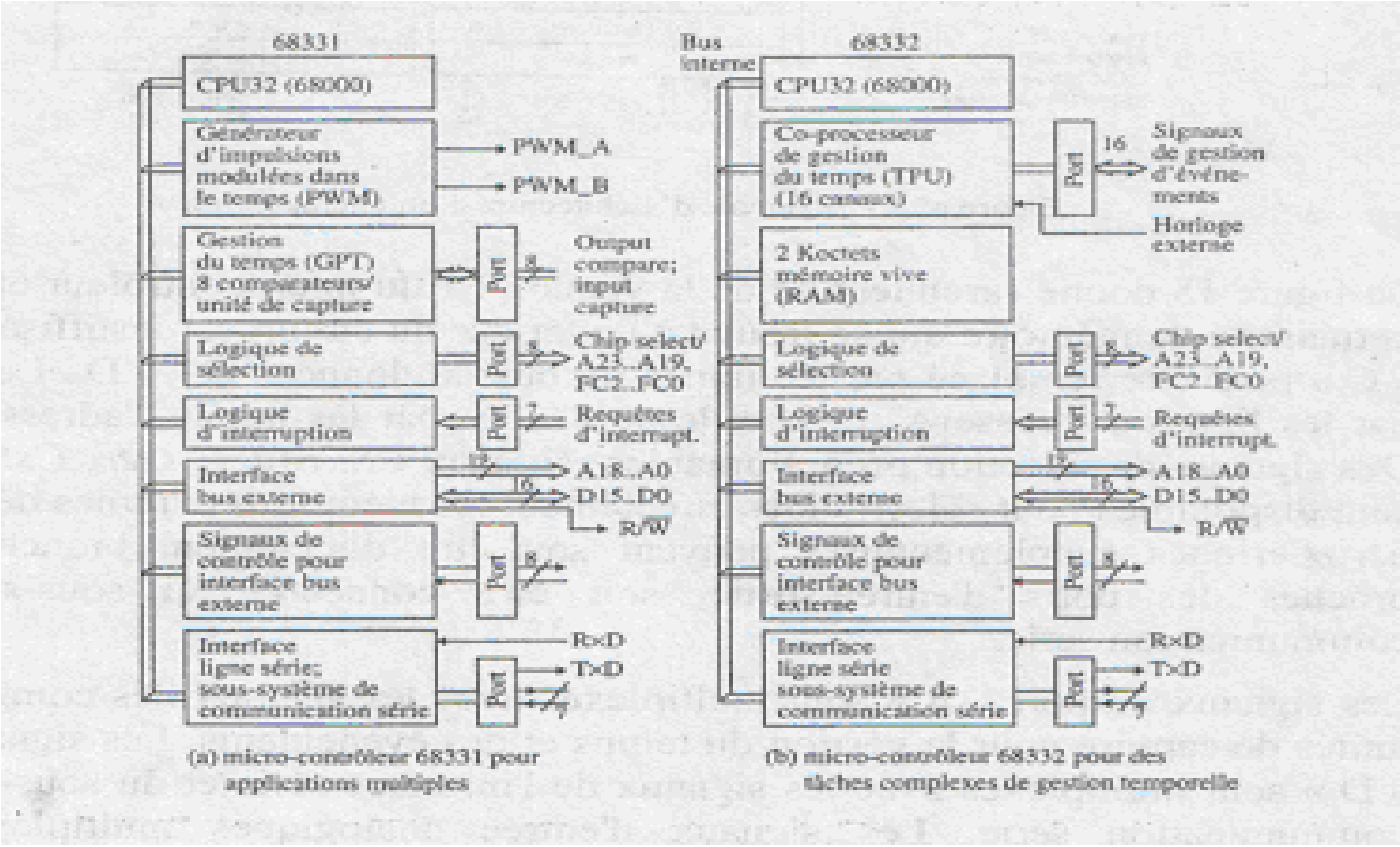


Figure n° 16 : Microcontrôleur 68331 pour application de contrôle diverses & Microcontrôleur 68332 pour le gestion d'événements temporels

Le microprocesseur 68000 était au milieu des années 80 un des processeurs le plus utilisés pour la réalisation de cartes industrielles performantes. Motorola s'est basé sur le succès du 68000 pour développer les microcontrôleurs de la famille 68300 qui incorporent un microprocesseur 68000 amélioré (CPU32) ainsi que diverses fonctions d'entrées/sorties. Les figures 16 et 17 donnent les schéma-blocs fonctionnels de quelques microcontrôleurs membres de la famille 68300.

La plupart des broches d'entrée-sortie de ces microcontrôleurs sont à double fonction : soit elles se comportent comme les broches d'un port bi-directionnel, soit elles permettent de répercuter à l'extérieur les adresses, données et signaux de contrôle du processeur ainsi que les signaux provenant des logiques de sélection et de gestion du temps.

Les microcontrôleurs 68331 (Figure 16.a) comporte les éléments qui permettent de concevoir un système de commande industriel performant. En particulier, ses 8

comparateurs et unité de capture d'événements temporels, ainsi que ses possibilités de génération et de masquage d'interruptions internes et externes, en font un processeur performant pour la commande en en temps-réel.

Les microcontrôleurs 68332 (Figure 16.b) est spécialisé dans la commande temps-réel de dispositifs extérieurs nécessitant un séquencement extrêmement sophistiqué (moteur de voiture moderne).

Par contre, le microcontrôleur 68341 (Figure 17.a) ne comporte qu'un seul compteur programmable permettant d'engendrer une base de temps interne. Ce microcontrôleur comporte deux canaux d'accès mémoire direct et peut donc être utilisé pour des tâches de copie de plans de bits, nécessaires pour la commande de dispositifs multimédia, ainsi que pour la commande d'imprimantes.

Le microcontrôleur 68328 (Figure 17.3b) fait partie de la même famille. Il comporte 2 compteurs programmables, une interface PCMCIA (pour carte d'extension ayant la taille d'une carte de crédit) et une interface pour écrans à cristaux liquides *(Liquid Crystal Display).* En outre, le 68328 comporte un mode de fonctionnement à consommation d'énergie réduite. Le 68328 est donc un microcontrôleur conçu pour la conception d'ordinateurs portables et d'assistants personnels numériques.

D'autres microcontrôleurs de la même famille ont été développés pour des domaines d'application spécifiques. Le microcontrôleur MC68356 comporte en plus de son processeur 68000 un processeur de traitement de signal DSP56002, ainsi que des canaux de communication sériels travaillant par accès mémoire direct (DMA). Ce microcontrôleur se prête particulièrement bien à la réalisation d'applications de communications multimédia, où le processeur de traitement des signaux peut être utilisé pour comprimer/décomprimer et traiter des sons et des images. 11 se prête également à la réalisation d'interfaces MODEM pour la communication de données à haut débit sur lignes téléphoniques.

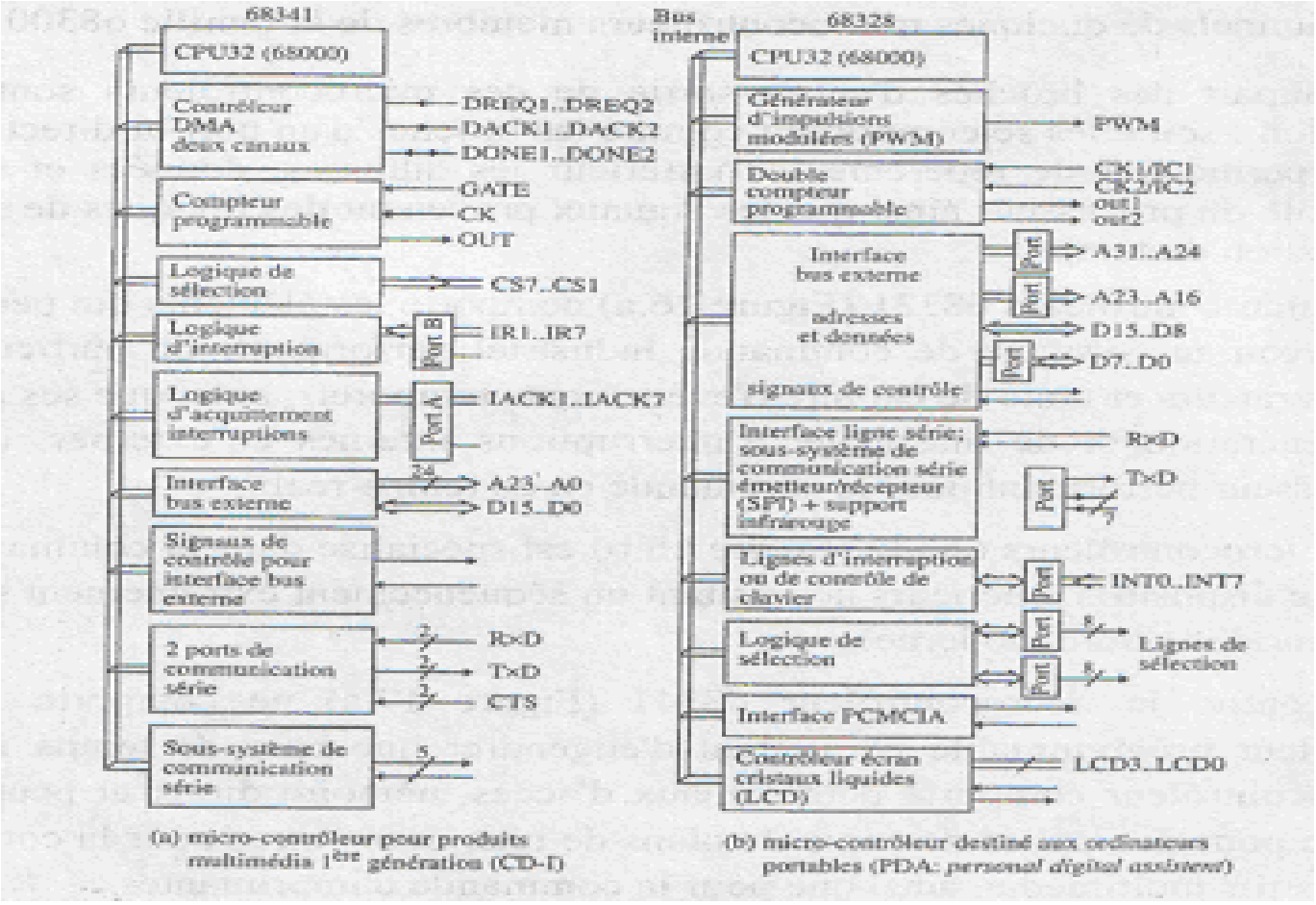


Figure n° 17 : Microcontrôleurs pour produits multimédia et ordinateurs portables

Le microcontrôleur MC68360, qui comporte de nombreux ports de communication série, est destiné aux applications de communications et les protocoles. Il permet par exemple de réaliser des interfaces à des bus sériels, tels que Ethernet et AppleTalk. Il est en particulier approprié pour la réalisation de ponts *(bridges)* entre réseaux différents.

L'architecture du microcontrôleur 68331 est détaillée à la figure 18. Comme indiqué à la figure 17, le 68331 comprend 8 unités de comparaison/capture d'événements, deux unités de génération de signaux à modulation de largeur d'impulsion, une ligne série, un sous-système de communication série synchrone (QSPI), un interface bus externe, 11 lignes de sélection (multiplexées avec un port 8 bits), 7 lignes de requêtes d'interruption (multiplexées avec un port 8 bits), ainsi que 5 ports d'entrée-sortie 8 bits multiplexés avec les signaux mentionnés ci-dessus.

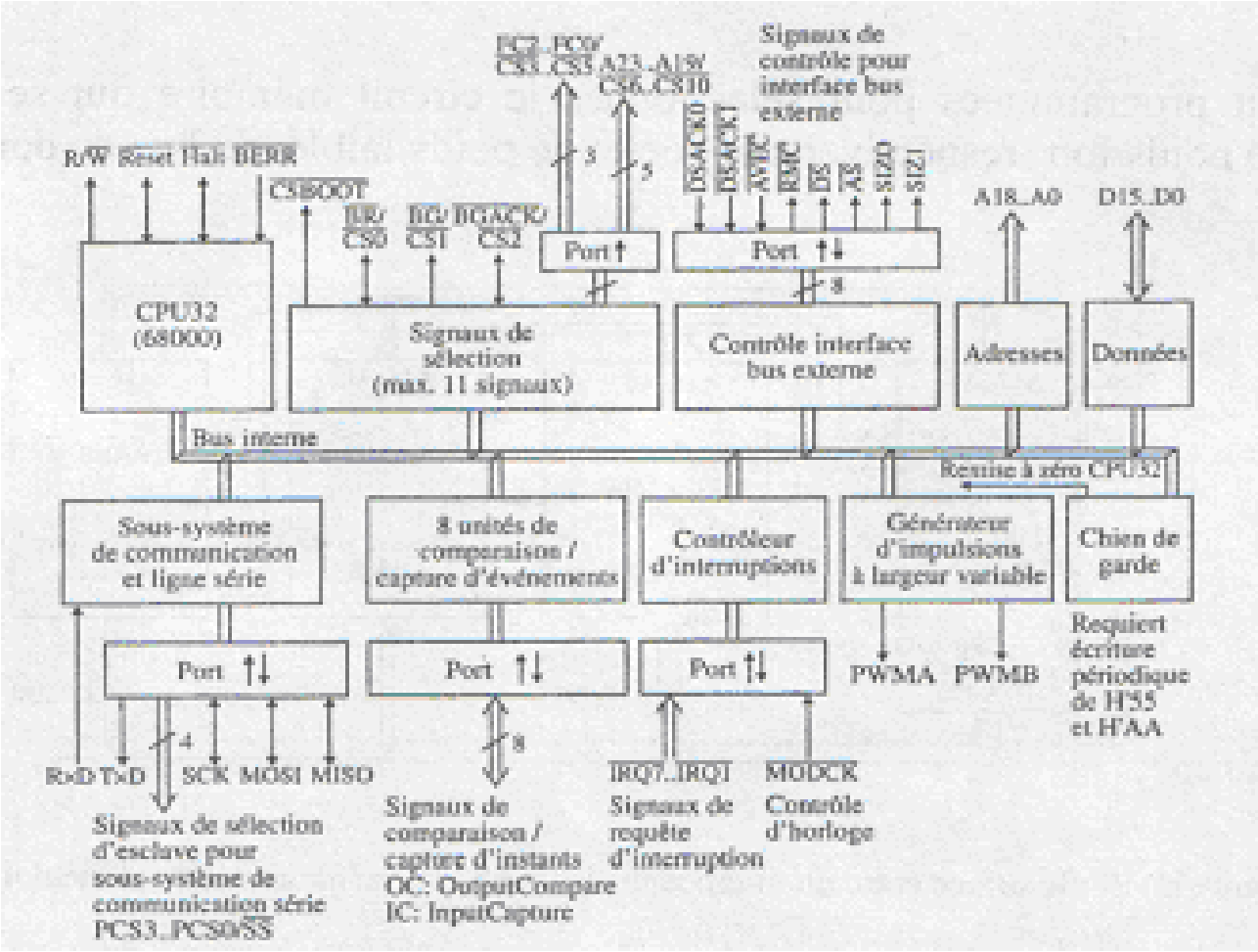


Figure n° 18 : Schéma-bloc avec les principaux signaux du microcontrôleurs 68331

De par la possibilité d'utiliser pratiquement toutes les broches libres comme signaux d'entrée-sortie. Cette architecture est suffisamment flexible pour s'adapter à différents types d'application. Les applications de décompte d'événements et de synthèse de signaux peuvent utiliser les nombreuses unités de comparaison temporelle et de capture du temps, ainsi que les 7 différents signaux de requête d'interruption.

Des circuits périphériques additionnels tels que des convertisseurs A/D ou des circuits de commande d'afficheurs à cristaux liquides peuvent être facilement interfacés au 68331 par le biais de son sous-système de communication série. De plus, un chien de garde similaire au chien de garde du microcontrôleur 68HC11 assure le redémarrage automatique du système en cas de fonctionnement erroné

(erreur de programmation situation temps-réel imprévue, défaillance passagère du matériel).

II est facile de concevoir un micro-ordinateur complet en interfaçant le microcontrôleur 68331 à une mémoire morte et à une mémoire vive (Figure 19).

Le signal de sélection programmable CSBOOT actif lors de la mise à zéro du processeur *(reset)* sélectionne la mémoire morte (EPROM) qui contient le code de démarrage du système (*boot).* Les mémoires vives RAMH et RAML, placées respectivement sur les lignes de poids fort et de poids faible sont sélectionnées par les lignes de sélection programmable CS0 et CSl. Les lignes de sélection CS0 et CS1 sont programmées pour sélectionner le circuit mémoire qui se trouve sur l'octet de poids fort, respectivement l'octet de poids faible, du bus de données.

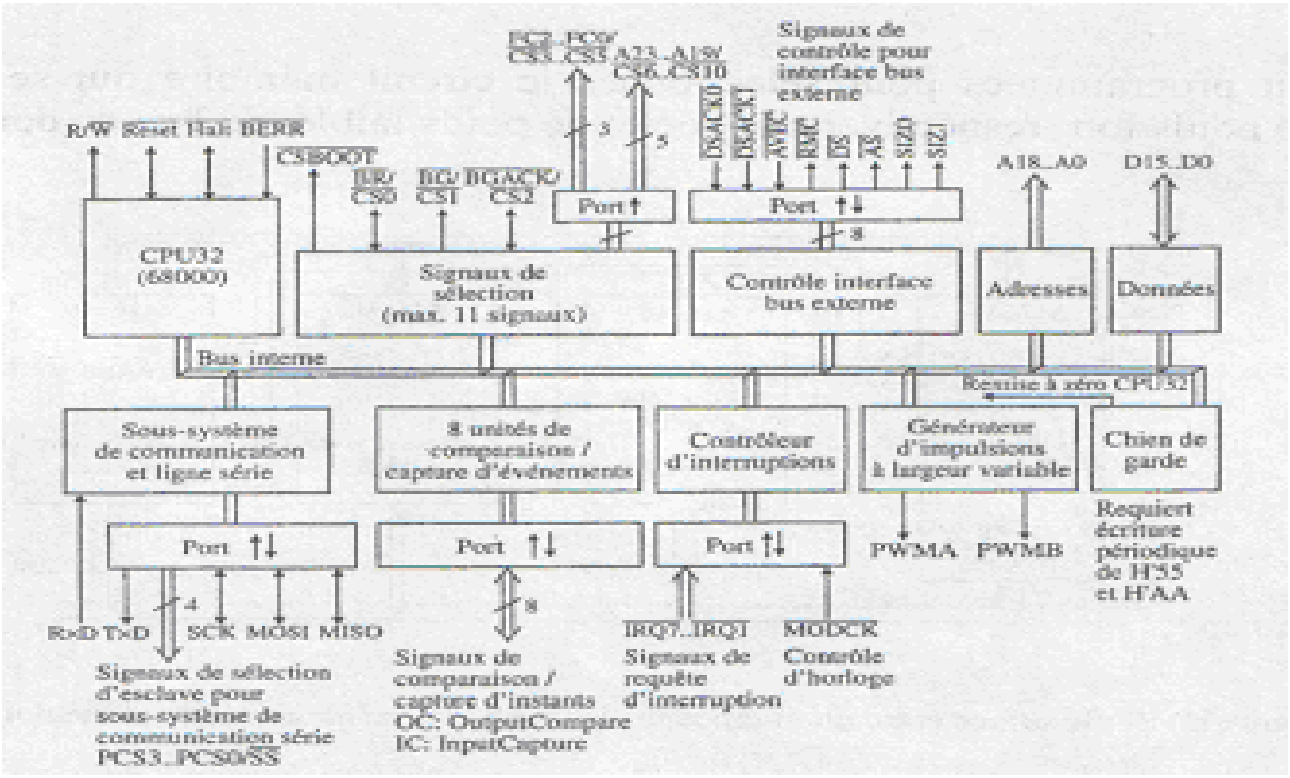


Figure n° 19 : Interface entre un microcontrôleurs 68331, mémoire morte et mémoire vive

Le développement de programmes pour microcontrôleurs est facilité par l'existence d'une interface sérielle additionnelle (omise dans les schémas de principe des figures

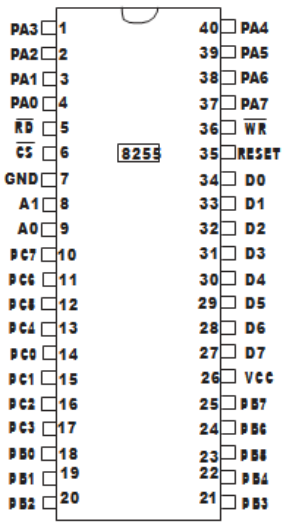
15 à 17) permettant à un système de développement (logiciel sur PC) de communiquer avec le microcontrôleur en mode déverminage *(Background Debug Mode : BDM).*

Le système de développement peut armer le mode de déverminage lors du démarrage *(reset)* du microcontrôleur en activant simultanément au signal *RESET* le signal *BKPT* en entrée sur le microcontrôleur. Ensuite, il est possible de commuter entre mode de déverminage et mode d'exécution normal. En mode de déverminage, des ordres qui arrivent par les lignes BDM permettent d'initialiser les registres internes du microcontrôleur, de lire et écrire dans les registres, de lire et écrire en mémoire du système cible, d'appeler une routine du programme chargé sur le système cible et de remettre à zéro l'état des registres de commande et ports d'entrée-sortie du microcontrôleur. Lorsque le programme s'exécute en mode normal sur le système cible, il est possible de revenir en mode de déverminage par activation de la broche BKPT *(breakpoint)* en entrée sur le microcontrôleur ou par l'exécution de l'instruction BGND ($4AFA), instruction qui est illégale lorsque le mode de déverminage n'est pas armé.

2° Le circuit d'interface parallèle de la famille Intel « 8255 »

Nous allons étudier dans cette partie un circuit de la famille Intel, programmable et particulièrement puissant pour interfacer le monde extérieur à un système à P : il s'agit du composant « 8255 ». Le rôle d'un tel circuit est d'opérer des transferts entre le bus d'un système à P et un ou plusieurs périphériques. Son fonctionnement étant régi par la programmation qui est en faite, il ne nécessite pas de circuits particuliers pour le mettre en oeuvre.

2.1 Description générale



Le circuit « 8255 », schématisé par la figure ci-dessous, est doté de 40 broches :

→ D0 - D7, bus de données (8 conduits parallèles bidirectionnels).
→ $\overline{CS}$, (Chip Select) permet la sélection du circuit.
→ Reset, permet la réinitialisation du circuit : un niveau haut sur cette entrée réinitialise le registre de contrôle et met tous les ports du « 8255 » en mode entrée !

Figure n° 35 : Schéma de brochage du circuit 8255

→ $\overline{RD}$, (ReaD) pour permettre la lecture des données : un niveau bas sur cette entrée autorise le « 8255 » à envoyer son information de données ou de statut vers le P, par le bus de données. En fait, le P lit le « 8255 ».
→ $\overline{WR}$, (WRite) pour permettre l'écriture des données : un niveau bas sur cette entrée permet au P d'écrire des données et des mots de contrôle dans le « 8255 ».
→ A0 - A1, permettent l'adressage des différents ports du circuit : ces signaux d'entrées pour le « 8255 », en conjonction avec les signaux RD et WR contrôlent la sélection d'un des trois ports ou des mots de contrôle des registres. Ils sont généralement reliés directement aux bits de poids faibles du bus d'adresse.
→ PA0 - PA7, huit lignes parallèles correspondant au port « A ».
→ PB0 - PB7, huit lignes parallèles correspondant au port « B ».
→ PC0 - PC7, huit lignes parallèles correspondant au port « C ».
→ Vcc, pour alimenter le circuit « + 5V ».
→ Gnd, c'est la masse du circuit.

N.B *Le 8255 est doté donc de 24 broches d'entrées/sorties qui peuvent être programmées individuellement en deux groupes de 12 lignes et utilisées dans trois modes différents (décrits par la suite).*

Les entrées RD et WR sont connectées respectivement, soit à MER ou I/OR, soit à MEMW ou I/OW suivant que le périphérique est adressé comme un mot mémoire ou comme un périphérique (avec les instructions IN et OUT en Assembleur).

2.2 <u>Structure interne</u>

D'après la figure n° 36, le « 8255 » est composé des blocs suivants :

→ De trois ports de 8 bits appelés : port « A » (dont les sorties sont amplifiées et verrouillées ; les entrées ne sont que verrouillées), port « B » (dont les sorties sont amplifiées et verrouillées ; les entrées ne sont qu'amplifies) et port « C » (dont les sorties sont amplifiées et verrouillées ; les entrées ne sont qu'amplifies). D'autre part, ce dernier port peut être divisé en deux groupes de 4 bits en mode de contrôle. Chaque groupe comporte un verrou de 4 bits et peut être utilisé pour les sorties de signaux de contrôle et des entrées de signaux de statut en liaison avec les ports « A » et « B ».

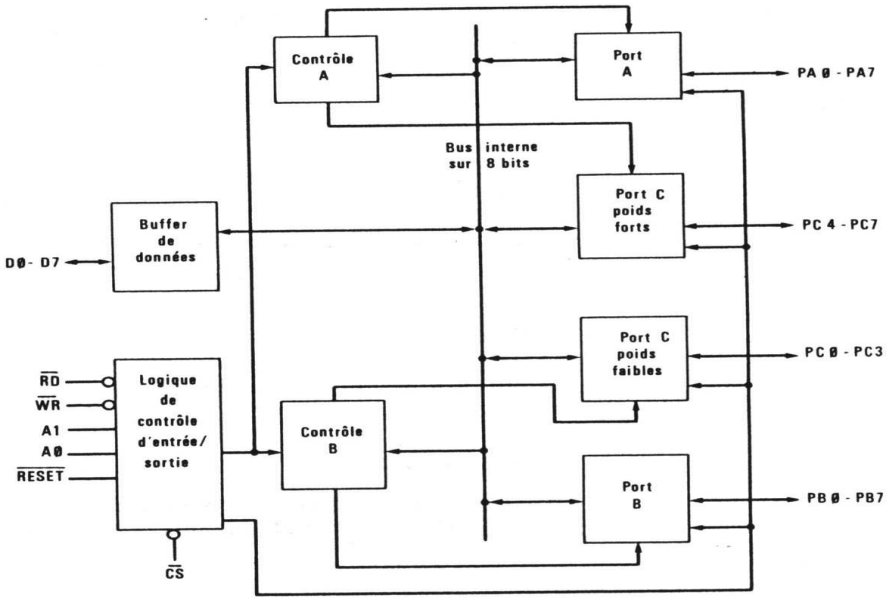


Figure n° 36 : Schéma synoptique interne du circuit « 8255 »

→ D'un Amplificateur de bus bidirectionnel à trois états (de 8 bits). Il permet de relier le circuit « 8255 » au bus de transfert du système. Les données sont transmises ou reçues par cet amplificateur après exécution par le P d'une instruction d'E/S. Notons également que les mots de commande du « 8255 » ainsi que les mots d'état transitent également par cet amplificateur de bus.

→ D'une ligne de contrôle de lecture/écriture. Elle prend en charge le contrôle de tous les transferts de données tant internes qu'externes et ce, qu'il s'agisse de données, de mots de contrôle ou de mots de statut. Elle reçoit en entrée des informations provenant du bus d'adressage et de contrôle du P et génère des commandes pour les deux contrôleurs de groupes.

→ De deux blocs de contrôle de groupes (« A » ou « B »). Notons au préalable que la configuration de fonctionnement de chaque port est déterminée par le programme qui régit le système. Le P envoie un mot de commande au « 8255 ». Ce mot contient des informations telles que le mode, les « bit set » ou « bit reset » qui initialisent la configuration de fonctionnement du 8255. Chaque bloc de contrôle (groupe « A » ou groupe « B ») accepte des commandes de la logique de lecture/écriture, reçoit des mots de commande par le bus interne et génère ses propres commandes pour les ports qui lui sont associés. Le contrôle du groupe « A », gère le port « A » ainsi que le haut du port « C » (les lignes de « c4 » à « c7 » le contrôle du groupe « B », gère le port « B » et le bas du port « C » (les lignes de « c0 » à « c3 »).

2.3 <u>Principe de fonctionnement</u>

A1	A0	D	R	S	

Figure n° 37 : Différentes configurations du « 8255 »

Pour exploiter le circuit « 8255 », nous avons à notre disposition 4 adresses différentes (voir la figure ci-dessous), 3 adresses précisant sur quel bornier on veut travailler (pour CS = 0) : port « A », port « B » ou port « C » et une adresse permet le stockage du mot de commande qui indique dans quel mode le « 8255 » doit travailler. Si CS = 1, le « 8255 » met la liaison avec le bus de données en haute impédance.

D'après le tableau ci-dessus, le circuit « 8255 » ne peut recevoir ou envoyer les données au P que lorsque CS est activée (niveau bas). D'autre part, avant toute utilisation de ce circuit, il faut lui fournir un mot de commande pour lui signaler dans quel mode il va travailler : ce mot de commande sera envoyé par une instruction d'écriture (OUT en assembleur). Il suffira ensuite d'utiliser les instructions de lecture ou d'écriture pour exécuter les transferts. Cependant, si l'entrée Reset a été activée, les borniers « A », « B » et « C » sont automatiquement mis dans le mode entrée !

Les modes de fonctionnement des ports « A » et « B » peuvent être définis séparément et les deux moitiés du port « C » sont configurées suivant les besoins respectifs des ports « A » et « B ». Tous les registres de sortie y compris les bascules de statut sont réinitialisées lors d'un changement de mode.

Les différents modes peuvent être combinés pour optimiser l'utilisation du « 8255 ». Par exemple, le port « B » peut être programmé en mode « 0 » pour piloter des fermetures de contacts ou l'affichage de données pendant que le groupe « A » est programmé en mode « 1 » pour fonctionner avec un clavier ou un lecteur de bande, ... La figure n° 38 donne le format du mot de contrôle correspondant aux trois ports.

Figure n° 38 : Format de définition des différentes modes

Notons également que chacun des huit bits du port « C » peut être positionné « set » ou remis à zéro « reset » en utilisant uniquement une instruction de sortie. Cette possibilité simplifie les contraintes sur le programme dans les opérations de contrôle.

2.3.1 Le mode « 0 »

Ce mode permet de travailler sur chacun des ports « A », « B », « Cbas » et « Chaut » soit en entrée, soit en sortie. D'autre part,

→ nous avons deux ports de 8 bits (port « A » et port « B ») et deux ports de 4 bits (port « Cbas » et port « Chaut »).

Figure n° 39 : Schéma d'interfaçage du « 8255 » en mode « 0 »

→ les sorties sont mémorisées ou verrouillées : les informations sont toujours présentes en sortie du « 8255 ».

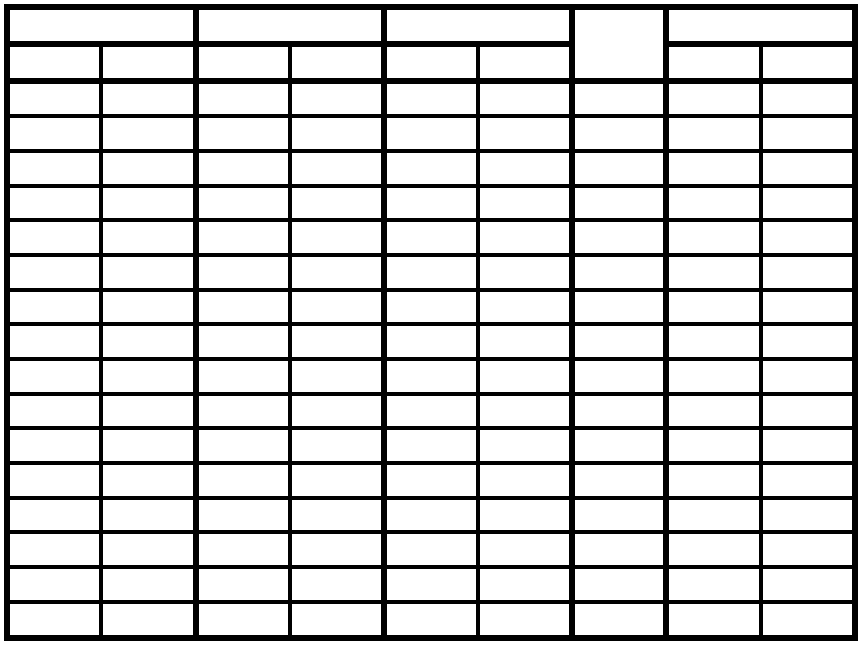


Figure n° 40 : Différentes configurations correspondant au mode « 0 »

→ les entrées ne sont pas mémorisées : il faut donc que l'information en entrée du « 8255 » soit stable quand le P vient la lire.

Figure n° 41 : Différentes configurations possibles en mode « 0 »

Figure n° 41 (suite) : Différentes configurations possibles en mode « 0 »

Figure n° 41 (suite) : Différentes configurations possibles en mode « 0 »

→ en se basant sur le mot de contrôle schématisé sur la figure n° 38, nous pouvons déduire 16 configurations d'E/S différentes. Ceci est résumé sur le tableau de la figure n° 40 et sur les différents schémas de la figure 41.

N.B. *Toute information lue sur un port programmé en sortie est sans signification et toute tentative d'écriture (de sortie) sur un port programmé en entrée est sans action.*

Si on utilise le port « Cbas » en sortie et le port « Chaut » en entrée ou inversement, on effectuera une instruction d'entrée ou de sortie sur le port « C » complet (8 bits). Seuls les 4 bits correspondant à « Cbas » ou à « Chaut » auront une signification. Lors d'une opération de lecture ou une action lors d'une opération d'écriture.

2.3.2 Le mode « 1 »

Dans ce mode de fonctionnement, on peut considérer que le « 8255 » est scindé en deux groupes fonctionnels (voir la figure n° 42): le groupe « A » et le groupe « B ». Le groupe « A » comprend : le bornier « A », utilisé pour le transfert parallèle de l'information en entrée ou en sortie et le bornier « Chaut », utilisé avec les signaux de dialogue (échantillonnage, donnée de saisie, demande d'interruption) ; le groupe « B » comprend : le bornier « B », utilisé pour le transfert parallèle de l'information en entrée ou en sortie et le bornier « Cbas », utilisé avec les signaux de dialogue.

Figure n° 42 : Schéma d'interfaçage du « 8255 » en mode « 1 »

Nous allons définir, dans ce qui suit, les différents signaux de dialogue pour les deux cas suivants :

→ Lorsque le port associé du groupe est en entrée (figure n° 43), **STB** (STRobe input) est une entrée de synchronisation. Elle permet l'échantillonnage de la donnée présente sur le port associé. Un niveau bas sur cette entrée autorise le passage de la donnée dans le registre d'entrée du port, celui-ci sera « verrouillé » sur le front montant du signal STB (voir les figures 43 et 44).

Figure n° 43 : Mot de contrôle et schémas correspondant au mode « 1 » pour une entrée

IBF (Input Buffer Full) ou bascule de buffer d'entrée plein, c'est un signal de sortie qui indique que la donnée est prête en entrée sur le port. Cet indicateur est mis à « 1 » sur le front descendant du signal STB. Il sera remis à « 0 » sur le front montant de RD, c'est à dire lorsqu'on aura effectué une opération de lecture de la donnée entrée. En clair, c'est un accusé de réception ! (voir les figures 43 et 44).

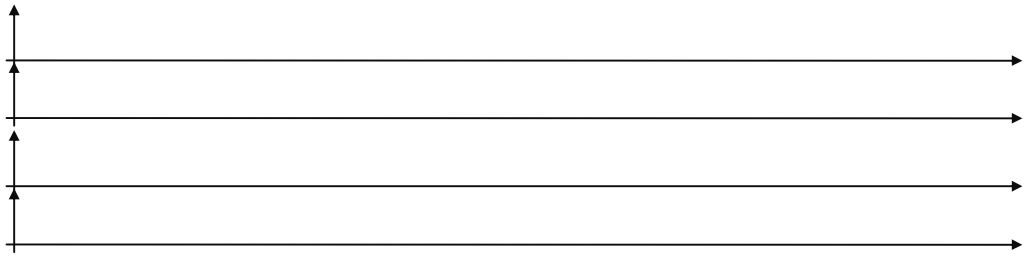


Figure n° 44 : Chronogrammes correspondant au mode « 1 » pour une entrée

INTR (INTerrupt Request) ou demande d'interruption. C'est un signal de sortie qui peut être utilisé pour effectuer une demande d'interruption au P. Il passe à « 1 » sur le front montant de STB si le masque qui lui est associé « INTE » est préalablement mis à « 1 » par une commande de positionnement du bit du port « C » (bas ou haut) correspondant au rang « INTE » dans le mot d'état. « INTE » est en fait une bascule interne dont l'état permet de masquer ou non la demande d'interruption ! INTR est remis à « 0 » sur le front descendant de RD : ça correspond à une lecture de l'information (voir les figures voir les figures 43 et 44). Notons à l'occasion que « INTE A » est contrôlée par le bit set/reset du bit « c4 » du port « C » et « INTE B » est contrôlée par le bit set/reset du bit « c2 ».

Figure n° 45 : Mot de contrôle et schémas correspondant au mode « 1 » pour une sortie

→ Lorsque le port associé du groupe est en sortie (figure n° 45),

$\overline{\text{OBF}}$ (Output Buffer Full), c'est un signal de sortie qui nous indique que le buffer de sortie est plein ou que la donnée est prête en sortie du port (voir les chronogrammes de la figure n° 46). OBF passe à « 0 » sur le front montant de WR : c'est à dire lors d'une opération de sortie (écriture) sur le port associé. Elle est remise à « 1 » sur le front descendant de ACK (c'est la réponse du périphérique comme quoi il a bien reçu la donnée).

$\overline{\text{ACK}}$ (ACKnowledge) ou acquittement, c'est une entrée d'accusé de réception. Un niveau « 0 » sur cette entrée signale au « 8255 » que la donnée a été prise en compte (voir les chronogrammes de la figure n° 46).

INTR (INTerrupt Request) ou demande d'interruption. C'est un signal de sortie qui peut être utilisé pour effectuer une demande d'interruption au P et ceci lorsque le périphérique a reçu la donnée. Il indique aussi que celui-ci peut recevoir une autre. Cette passe à « 1 » sur le front montant de ACK si le masque qui lui est associé « INTE » est préalablement mis à « 1 ». INTR est remis à « 0 » sur le front descendant de WR.

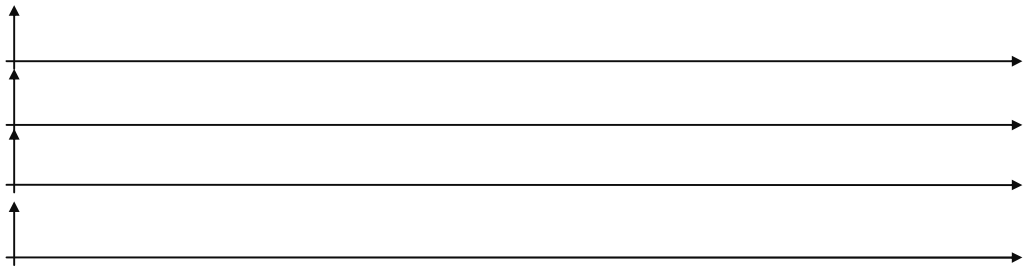


Figure n° 46 : Chronogrammes correspondant au mode « 1 » pour une sortie

Figure n° 47 : Différentes combinaisons correspondant au mode « 1 »

En effet, les types de transferts possibles dans le mode « 1 » sont :

1°) Transferts caractère par caractère par interruption. Les signaux INTR, indiquant qu'une donnée est entrée sur le bornier ou qu'une donnée a été reçue par le périphérique, peuvent être utilisée comme signaux de demande d'interruption vers le P.

2°) Transferts caractère par caractère par test d'état. Par une simple opération de lecture du bornier « C », le P peut connaître le « mot d'état » du « 8255 ».

Les différentes combinaisons possibles sont résumées sur le tableau de la figure n° 47. Les signaux de dialogue étant au nombre de 6, il reste deux fils du bornier « C » qui peuvent être utilisés en entrée ou sortie parallèle : pour des commandes particulières de périphériques par exemple.

2.3.3 Le mode « 2 »

Ce mode permet d'utiliser le bornier « A » en bus bidirectionnel, les informations de dialogue transitent sur 5 fils du bornier « C ». Les entrées ou sorties d'information sont mémorisées. Le bornier « B » et les trois fils restant du bornier « C » peuvent être utilisées dans les modes « 0 » ou « 1 » (voir les figures suivantes).

Figure n° 48 : Schéma d'interfaçage du « 8255 » en mode « 2 »

Comme dans le mode « 1 », les transferts caractère par caractère par interruption ou par test d'état, sont possibles.

INTR (INTerrupt Request) ou demande d'interruption. Un « 1 » sur cette sortie peut être utilisé pour une interruption du P pour des opérations d'entrées et de sorties.

→ Lorsqu'il s'agit des opérations d'écriture,
OBF (Output Buffer Full), c'est un signal de sortie. Le passage de cette ligne à l'état bas indique que le P a écrit des données dans le port « A ».

Figure n° 49 : Mot de contrôle et schéma correspondant au mode « 2 »

ACK (ACKnowledge) ou acquittement, c'est une entrée d'accusé de réception. Un niveau « 0 » sur cette entrée valide la sortie trois états du port « A » pour l'émission de données. Autrement, la sortie du buffer est en état de haute impédance.

INT 1, bascule de « INTE » associée à « OBF », est contrôlée par le bit set/reset du bit « c6 ».

Figure n° 50 : Chronogrammes correspondant au mode « 2 »

→ Lorsqu'il s'agit des opérations des lectures,

$\overline{\text{STB}}$ (Strobe input), c'est une entrée de synchronisation. Un « 0 » sur cette entrée charge les données dans le verrou d'entrée.

IBF (Input Buffer Full) ou bascule de buffer d'entrée plein. Un niveau « 1 » sur cette sortie indique que les données ont été chargées dans le verrou d'entrée.

INT 2, bascule de « INTE » associée à « IBF », est contrôlée par le bit set/reset du bit « c4 ».

Le tableau suivant résume les différents mots de commande du mode « 2 » ainsi que les mots d'état correspondant.

Figure n° 51 : Mots de commande et mots d'état correspondant au mode « 2 »

Les schémas correspondant aux combinaisons mode « 2 » - mode « 0 » et mode « 2 » - mode « 1 » sont donnés par les figures suivantes.

Figure n° 52 : Combinaisons possibles correspondant au mode « 2 »

Figure n° 52 : Combinaisons possibles correspondant au mode « 2 » (suite)

2.3.4 Fonctions de contrôle des interruptions

Quand le circuit « 8255 » est programmé en mode « 1 » ou « 2 », des signaux de contrôles sont fournis et il est possible de les utiliser comme des demandes d'interruption pour le P. Les signaux de demande d'interruption générés par le port « C » peuvent être validés ou invalidés en positionnant ou en remettant à zéro la bascule « INTE » qui leur est associée, par l'intermédiaire du bit set/reset relatif au port « C ». Cette fonctionnalité permet au programmeur d'autoriser ou d'interdire à un périphérique d'E/S donné de créer une interruption P sans que cela n'affecte aucun autre périphérique. La bascule « INTE » peut être définie comme suit (voir la figure n° 53) :

(bit-set) → *INTE est positionné* → *l'interruption est autorisée ;*
(bit-reset) → *INTE est remis à 0* → *l'interruption est interdite.*

Notons que toutes les bascules du masque d'interruption sont remises à zéro lors d'une sélection de mode ou du reset du « 8255 ».

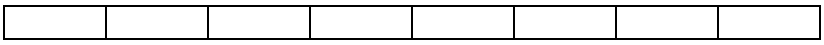


Figure n° 53 : Format du bit set/reset

2.4 Exemple de mise en application

Après avoir vu comment se connectait le circuit « 8255 » aux différents bus (du P et/ou des périphériques) et comment il se programme, il est logique de nous intéresser maintenant à l'exploitation d'un tel composant. D'après le schéma ci-dessous, le « 8255 » peut gérer à la fois un clavier entièrement décodé et un afficheur intelligent.

Figure n° 54 : Interfaçage de clavier et de dispositif d'affichage