Système minimum à base de 68000

Je tiens à remercier chaleureusement Monsieur **Philippe Meyne**, enseignant à l'IUT GEII d'Evry pour sa participation importante à l'élaboration de ce cours.

Objet : Etude de la réalisation d'un système minimum à base d'un µP de Motorola : le 68000.

Choix du processeur : Matériel existant mais les notions de base sont les mêmes quel que soit le processeur.

I.Rappels : structure d'un système informatique

I.1.Composition d'un système minimum

T7 : Les composants minimums d'un système informatique.

Microprocesseur : Coeur du système.

Timer : Evolution et cadencement.

ROM : Mémoire morte. Elle contient ce qui doit être résident : moniteur, boot…

RAM : Mémoire vive. Elle contient des données et du code volatils.

Entrées-Sorties : Communication avec l'extérieur.

T8 : Les relations entre ces composants : les bus.

Bus de données : Permet de faire circuler les données d'un composant à l'autre.

Bus d'adresses : Permet de dialoguer avec divers composants.

Bus de contrôle : Assure la synchronisation des échanges entre le processeur et les circuits périphériques.

T9 : On en déduit la structure d'un composant informatique.

Bus d'adresses, de données et de contrôle.

Remarque : Tous les composants utilisent les mêmes bus. Des conflits d'utilisation risquent de survenir. Cela implique la notion de sélection des circuits.

CS	R/W	Fonction
0	0	Ecriture
0	1	Lecture
1	X	Haute impédance

I.2.Réalisation d'un système minimum

T10 : Les composants que l'on vient de voir + les 3 bus.

Signaux de validation :

Assure la synchronisation des échanges entre le processeur et les circuits périphériques.

- Validation des adresses : Le µP indique que les adresses sont stables.
- Validation des données :
  - En écriture : Le µP indique que les données sont stables
  - En lecture : Le circuit périphérique indique que les données sont stables.
- Lecture/Ecriture : direction du bus de données.

Interruption : Prise en compte du temps utilisateur (temps réel).
Notion fondamentale de l'informatique industrielle.

Allocation des bus :

Utilisé dans le transfert rapide des données. Le µP s'exclut de l'échange et le périphérique écrit ou lit directement en mémoire : Direct Access Memory (DMA).

Attribution du bus : Demande de bus par le périphérique.

Bus alloué : Autorisation de prise du bus par le périphérique.

Horloge : Temps processeur.

Remise à zéro : Initialisation à la mise sous tension et déblocage en cas de "plantage" du système.

T11 : Le décodage d'adresses. 4 zones mémoire → Utilisation d'un démultiplexeur 2 vers $4=2^2$.

T12 : La sélection des boîtiers. Décodage + validation des données en écriture.

T13 : La validation des données. Validation des données en lecture.

II.Processeur 68000 de Motorola

TMotorola : 64 broches : bus d'adresses, bus de données, bus de contrôle…

Registres : SP, PC, SR.

Niveaux de tension : Vdd-Vss compris entre 3V et 7V.

Compatible TTL :
- Vih min = 2V
- Vih max = Vdd-Vss
- Vil min = Vss-0.3V
- Vil max = 0.8V
- Voh min = Vdd-0.75V
- Voh max = pas de limite en théorie

Fréquence d'utilisation : 4Mhz à 12.5Mhz selon la version.

Dissipation à 8mhz : 1.5W

II.1.Signaux du 68000

T14 : *Bus de données* : Bus bidirectionnel trois états de 16 bits.

Bus d'adresses : Bus unidirectionnel de 23 bits sans bit A0. Il peut donc adresser $2^3.2^{20}=8$Mm soit 16Mo.

Bus de contrôle : Echanges :

- $\overline{CLK}$: CLocK
- $\overline{AS}$: Adresse Strobe
- $\overline{UDS}, \overline{LDS}$: Upper/Lower Data Strobe
- R / $\overline{W}$: Read/Write
- asynchrone : $\overline{DTACK}$: DaTa ACKnoledge
- synchrone : $\overline{E}$: Enable (horloge)
- $\overline{VMA}$: Valid Memory Adress
- $\overline{VPA}$: Valid Peripheral Adress

Etat du processeur : Fc_0, Fc_1, Fc_2

Traitement des exceptions :

- $\overline{RES}$: RESet
- $\overline{HLT}$: HaLT. En entrée, stoppe le processeur. En sortie, indique un arrêt du processeur après une double erreur de bus.
- $\overline{BERR}$: Bus ERRor. Détection d'une erreur de bus.

$\overline{IPL1,2,3}$: Interrupt Pending Level. Interruptions externes.

Gestion d'attribution des bus :

$\overline{BR}$: Bus Request. Indique qu'un système extérieur désire prendre le contrôle des bus.

$\overline{BG}$: Bus Grand. Indique que le processeur va libérer les bus à la fin du cycle en cours.

$\overline{BGACK}$: Bus Grand ACKnoledge. Indique qu'un autre circuit a pris le contrôle des bus.

II.2.Organisation des données

Trois types de données :

octet :	8 bits
mot :	16 bits
mot long :	32 bits

Ces données sont gérées à l'aide de registres.

Registres de données : 8 registres de 32 bits.
Opérations sur octets, mots, mots longs.

ex :	move.b	#$01,D0	avant	FFFF FFFF
				FFFF FF01
	move.w	#$01,D0	avant	FFFF FFFF
				FFFF 0001
	move.l	#$01,D0	avant	FFFF FFFF
				0000 0001

Registres d'adresses : 7 registres de 32 bits
1 pointeur de pile utilisateur (USP)
1 pointeur de pile système (SSP)

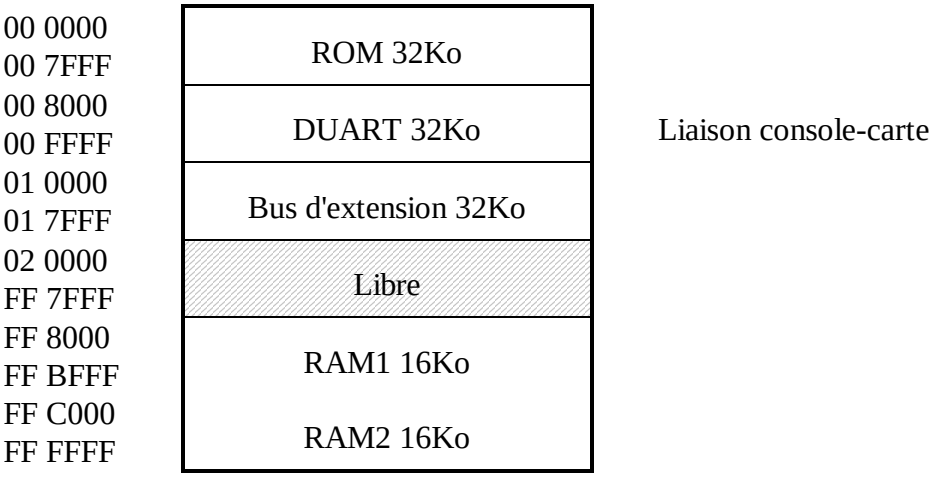
 Extension de signe : lea #$8000,A0 A0=FFFF8000

III.Décodage d'adresses

Principe : Réglementer l'accès au bus de données en n'autorisant les composants périphériques à accéder au bus que s'ils sont sélectionnés.

Réalisation : Associer à chaque composant une portion de l'espace d'adressage.

Espace d'adressage : Capacité d'adressage d'un processeur. Elle est donnée par la taille du bus d'adresses.
ex : 23 bits d'adresses soit 2^{23} mots = 16Mo.
Le découpage de l'espace d'adressage est le plan mémoire.



III.1.Conception du décodage d'adresses

Découpage de l'espace d'adressage en 4 parties de même taille.

$32Ko = 2^{15} \rightarrow A_{15}$.

4 parties : $2^2 \rightarrow$ 2 bits d'adresses d'où A_{15} et A_{16}.

soit :

A16	A15	$\overline{CSROM}$	$\overline{CSDUART}$	$\overline{CSIO}$	$\overline{CSRAM}$
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

T15 : *Composant* : Décodeur démultiplexeur.

- Circuit : 74LS139 — A,B : Entrées de sélection. 0…3 : Sorties de décodage. $\overline{G}$: Entrée de validation (active bas).
- RAM : Si on n'adresse ni la ROM, ni le DUART, ni le bus d'extension, on adresse la RAM. La différenciation des 2 boîtiers s'effectue grâce au bit A_{14}.

$A^{14} \rightarrow 2^4.2^{10} = 16Ko$.

T16 :

- Validité : Il ne doit être valide que lorsque les adresses sont valides sur le bus d'adresses, soit $\overline{AS}$ relié à $\overline{G}$ du premier décodeur.
- Accès : Uniquement pour des données ou des instructions. Information donnée par les broches Fc_0, Fc_1 et Fc_2 donnant l'état du processeur. Elles permettent de différencier les données utilisateur ou superviseur, les instructions utilisateur ou superviseur ainsi que les interruptions.

T17 : Schéma complet de décodage.

On peut simplifier le schéma en supprimant la différence entre superviseur et utilisateur.

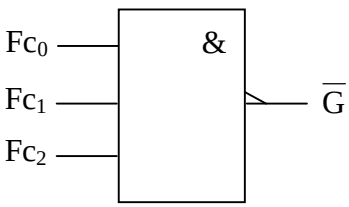


Figure III.1 : *Schéma simplifié de la sélection du décodage d'adresses.*

III.2.Sélection des boîtiers

Pb : Un boîtier doit être sélectionné sous deux conditions :

- L'adresse est comprise dans l'espace d'adressage du composant. C'est le rôle du décodage d'adresses.
- Les données sont disponibles sur le bus de données.

La sélection d'un boîtier fait donc intervenir :

- $\overline{CSxxx}$: Chip Select issu du décodage d'adresses.
- Validation des données :
  - $\overline{UDS}, \overline{LDS}$: Upper/Lower Data Strobe. Poids fort : poids faible du bus de données.
  - $R / \overline{W}$: Read/Write.

ex : cas du DUART, composant 8 bits.

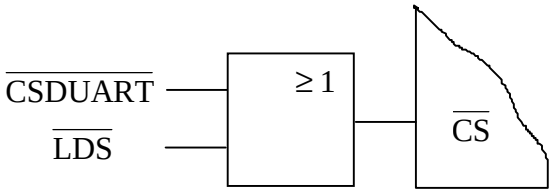


Figure III.2 : *Sélection du DUART.*

IV.Mise en oeuvre de la mémoire

Mémoire séparée en deux :

- Zone ROM (mémoire morte). Elle contient tout ce qui doit être résident :
  - Moniteur de la carte CAIM : éditeur, assembleur, déboggeur.
  - "Boot Loader" : chargeur du système d'exploitation.
- Zone RAM (mémoire vive). Elle contient les applications de l'utilisateur et les variables fugitives.
  - Zone système : utilisée par le système pour la gestion de la machine. Sur la carte CAIM, FF8000 à FF9000.
  - Zone utilisateur : réservée pour les applications. Sur la carte CAIM, FF9000 à FFFFFF.

IV.1.Le problème du bus d'adresses

T18 : Le 68000 est un processeur 16 bits, mais les circuits mémoires sont des circuits 8 bits. Donc, le 68000 gère la mémoire comme une mémoire 8 bits mais il adresse des mots de 16 bits. Il n'y a donc pas de bit A0 sur le bus d'adresses. Il est remplacé par les signaux $\overline{UDS}$ et $\overline{LDS}$.

Vu de l'intérieur

D_7 … D_0	
Octet1	$A_0^*=0$
Octet2	$A_0^*=1$

Adresse : $A_{23}…A_0^*$.

Vu de l'extérieur

D_{15} … D_8	D_7 … D_0
Octet1 $A_0^*=0$	Octet2 $A_0^*=1$
$\overline{LDS}=1$ $\overline{UDS}=0$	$\overline{LDS}=0$ $\overline{UDS}=1$

Bus d'adresses : $A_1…A_{23}$.

Lecture octet par octet avec $\overline{LDS}$ et $\overline{UDS}$.

Lecture d'un mot avec $\overline{LDS}=0$ et $\overline{UDS}=0$.

T19 : Adressage de mots ou de mots longs en mémoire.

IV.2.Mise en oeuvre de la mémoire morte

T21 : *Choix possibles* :

PROM :	Programmation chez le constructeur (pas d'évolution).
EPROM	Programmation sur site
EEPROM	Evolution possible car effaçable (UV, électriquement)
	Durée de vie plus courte.

Choix : EPROM 27128

14 bits d'adresses 8 bits de données	16 Koctets → 2*27128 pour avoir 32 Koctets=16Kmots.

Réalisation :

- On ne lit que des mots, donc pas d'utilisation de $\overline{LDS}$ et $\overline{UDS}$. Le bus d'adresses est commun au deux mémoires.
- Le bus de données est séparé en deux.
- On sélectionne la mémoire par les entrées Enable.
- On interdit les écritures.
- L'utilisation de $\overline{AS}$ n'est pas vraiment nécessaire (voir décodage d'adresses).

IV.3.Mise en oeuvre de la mémoire vive

T20 : *Choix possibles* :

Dynamique :	Grande capacité.
	Rapide.
	Rafraîchissement.
Statique :	Capacité moyenne.
	Moins rapide que la mémoire dynamique.
	Pas de rafraîchissement.

Choix : Mémoire statique 5564. Pour les faibles capacités on utilise souvent des mémoires statiques, plus simples à mettre en oeuvre.

13 bits d'adresses 8 bits de données	8 Koctets → 2*2*5564 pour avoir 2*16 Koctets.

Réalisation :

- Bus d'adresses commun à toutes les mémoires → On adresse a priori des mots.
- La mise en oeuvre nécessite les signaux $\overline{LDS}$ et $\overline{UDS}$ pour l'écriture d'octets.
  - ☞Mémoire paire : poids forts des mots → adressée par $\overline{UDS}$.
  - ☞Mémoire impaire : poids faibles des mots → adressée par $\overline{LDS}$.
- Le bus de données est séparé en deux.
- Sélection à l'aide de 3 signaux : $\overline{OE}$, $\overline{CE1}$ et $\overline{CE2}$ pilotés pas $\overline{CSRAM}$ $\overline{LDS}$ et $\overline{UDS}$.
- Connexion de R / $\overline{W}$.
- Deux blocs de 16 Koctets.
  - 1ier bloc de FF8000 à FFBFFF.
  - 2nd bloc de FFC000 à FFFFFF.

V.Gestion des échanges

Nécessité de la gestion des échanges : Les données sont des variables ou des instructions. Si leur intégrité n'est pas assurée, le système risque de planter. Il existe deux manières d'assurer l'intégrité des échanges :

Synchrone : L'intégrité de la transmission est assurée en synchronisme (cohérence) avec une horloge.

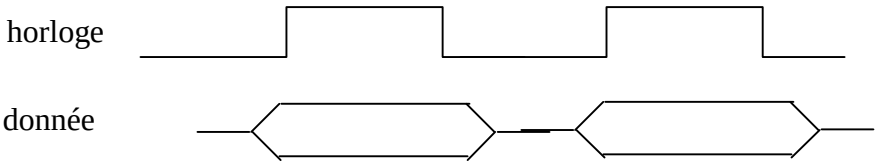


Figure V.1 : *Echange synchrone.*

Intérêt : Le récepteur est synchronisé sur l'émetteur des données.

Inconvénient : Il n'y a pas de retour ; l'émetteur ne sait pas si la donnée a bien été transmise.

Asynchrone : L'intégrité de la transmission est assurée par des signaux de validation.

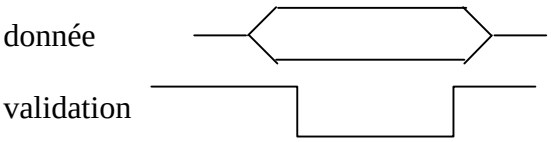


Figure V.2 : *Echange asynchrone.*

Ces deux types d'échange existent sur le 68000.

V.1.Echange asynchrone

C'est le mode d'échange par défaut du 68000. Il met en oeuvre les signaux de contrôle suivants :

$\overline{AS}$:	Adresse Strobe
$\overline{UDS}, \overline{LDS}$:	Upper/Lower Data Strobe
$R / \overline{W}$:	Read/Write
Fc_0, Fc_1, Fc_2 :	Etat du processeur (code fonction)
$\overline{DTACK}$:	DaTa ACKnoledge

V.1.1.Différents cycles de lecture et écriture

V.1.1.1.Cycle d'écriture d'un mot

Processeur	Esclave
Place l'adresse	
Place les codes fonction Fc_0, Fc_1, Fc_2	
Affirme l'échantillonnage d'adresses $\overline{AS}$	
Etablit $R / \overline{W}$ en écriture	
Place les données	
Affirme l'échantillonnage des données $\overline{UDS}, \overline{LDS}$	
	Décode les adresses
	Lit les données
	Affirme la reconnaissance de transfert des données $\overline{DTACK}$
Infirme $\overline{UDS}, \overline{LDS}$	
Infirme $\overline{AS}$	
Rétablit $R / \overline{W}$ en lecture (état haut)	
Fin de cycle	
	Infirme $\overline{DTACK}$

Pour l'écriture d'un octet, seuls $\overline{UDS}, \overline{LDS}$ changent suivant qu'il s'agit d'un octet de poids fort ou de poids faible.

V.1.1.2.Cycle de lecture d'un mot

Processeur	Esclave
Etablit $R/\overline{W}$ en écriture	
Place l'adresse	
Place les codes fonction Fc_0, Fc_1, Fc_2	
Affirme l'échantillonnage d'adresses $\overline{AS}$	
Affirme l'échantillonnage des données $\overline{UDS}, \overline{LDS}$	
	Décode les adresses
	Ecrit les données
	Affirme la reconnaissance de transfert des données $\overline{DTACK}$
Lit les données	
Infirme $\overline{UDS}, \overline{LDS}$	
Infirme $\overline{AS}$	
	Infirme $\overline{DTACK}$
Fin de cycle	

Pour la lecture d'un octet, seuls $\overline{UDS}, \overline{LDS}$ changent suivant qu'il s'agit d'un octet de poids fort ou de poids faible.

T22 : Visualisation des signaux décrits ci-dessus.

V.1.1.3.Cycle de lecture-modification-écriture

T23 : Ce cycle est utilisé dans le cas d'une ressource partagée, par exemple une imprimante. Un jeton (octet) indique si l'imprimante est libre. Pour qu'une unité centrale puisse utiliser cette ressource, il faut qu'elle lise la valeur du jeton et qu'elle la modifie si l'imprimante est libre. Ce cycle doit être indivisible pour que deux unités centrales ne puissent pas réserver la ressource en même temps.

V.1.2.Génération du $\overline{DTACK}$

Les composants Motorola, conçus pour fonctionner avec le 68000, ont une génération interne du $\overline{DTACK}$. Ce n'est pas le cas pour les mémoires par exemple. Une logique externe doit donc être réalisée pour établir les échanges. Or, un échange est synonyme de $\overline{UDS}, \overline{LDS}$ et $R/\overline{W}$ affirmés.

Dans le cas d'une ROM par exemple avec une horloge à 7.5Mhz (soit Th#133ns), on doit avoir le chronogramme suivant :

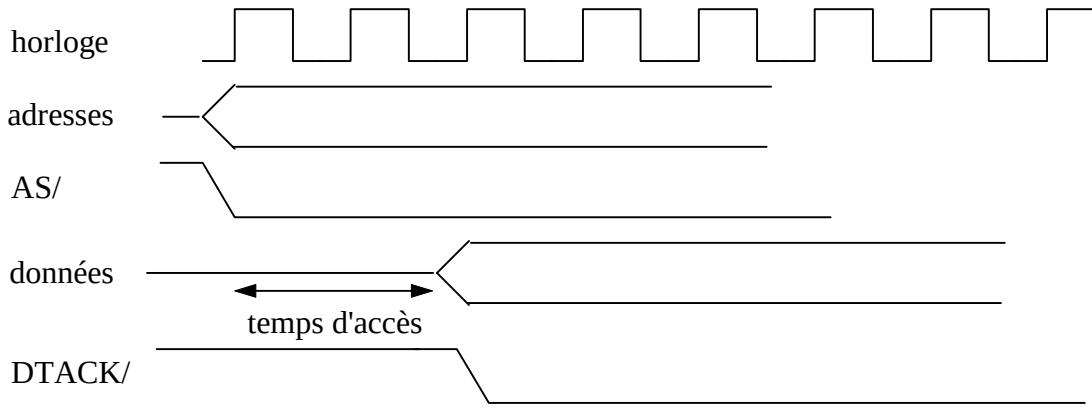


Figure V.3 : *Chronogramme du DTACK/ pour la ROM.*

Le $\overline{\text{DTACK}}$ doit être retardé de 250ms par rapport au début de l'échange (vitesse de la ROM).

Conclusion : Le $\overline{\text{DTACK}}$ dépend des signaux $\overline{\text{UDS}}$, $\overline{\text{LDS}}$ et R / $\overline{\text{W}}$ mais aussi du retard dû au temps d'accès au périphérique. On utilise donc un registre à décalage.

T27 : Par exemple, le 74LS164 : registre 8 bits permettant de décaler une donnée série vers la droite à chaque top d'horloge.

Hypothèses : ROM temps d'accès 250ns
RAM temps d'accès 550ns
horloge à 7.5 Mhz soit Th=133ns
d'où temps d'accès ROM#2Th
RAM#4Th

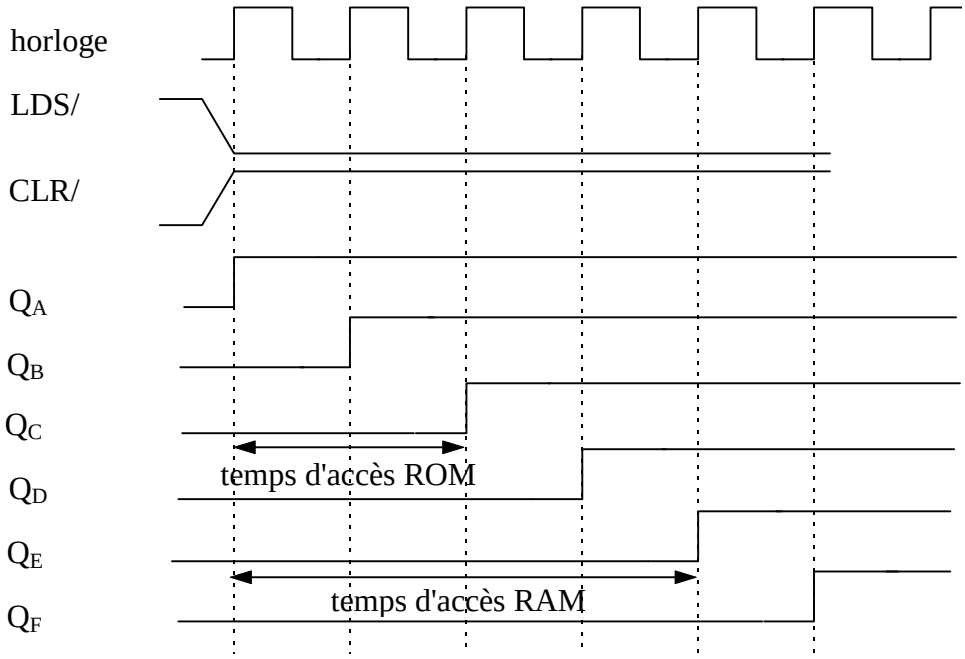


Figure V.4 : *Chronogramme de génération du retard des DTACK/ pour la ROM et la RAM.*

V.1.3.Génération de $\overline{\text{BERR}}$

Pour qu'un échange asynchrone puisse s'effectuer, il faut que le $\overline{\text{DTACK}}$ passe à zéro. En attendant, le processeur génère des états d'attente.

Si le $\overline{\text{DTACK}}$ ne passe jamais à zéro, le processeur reste indéfiniment en attente.

L'entrée $\overline{\text{BERR}}$ sert à interrompre cet état en limitant le nombre d'états d'attente du processeur. Cette limitation est réalisée à l'aide d'une logique externe.

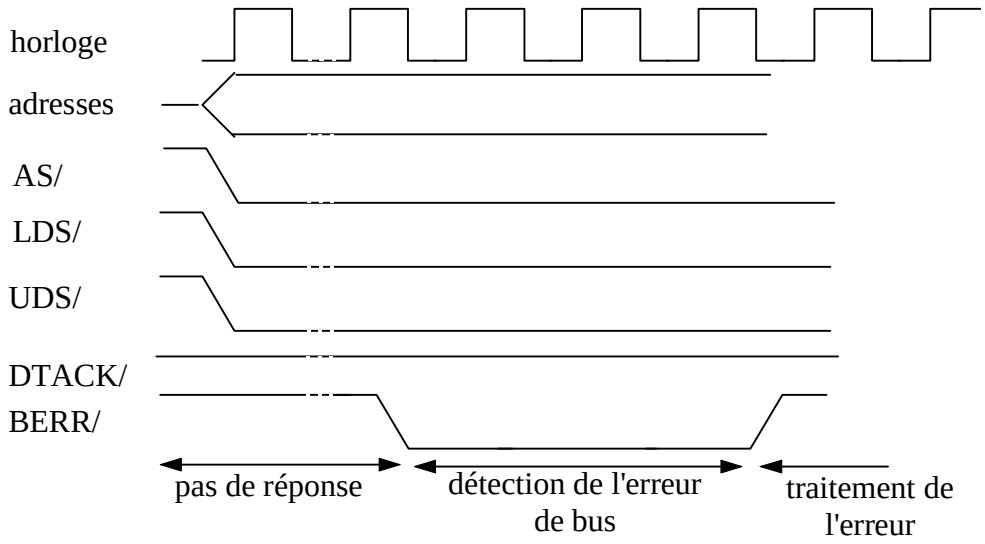


Figure V.6 : *Chronogramme de BERR/.*

La détection d'erreur revient à compter le nombre de coups d'horloge maximum pour un échange, par exemple, 16 coups. Le début du cycle de comptage correspond à $\overline{UDS}, \overline{LDS}$ ou $\overline{AS}$ actif.

Choix du compteur : 74LS193, compteur/décompteur 4 bits prépositionnables. La génération de $\overline{BERR}$ utilise la retenue.

CLR est actif au niveau haut.

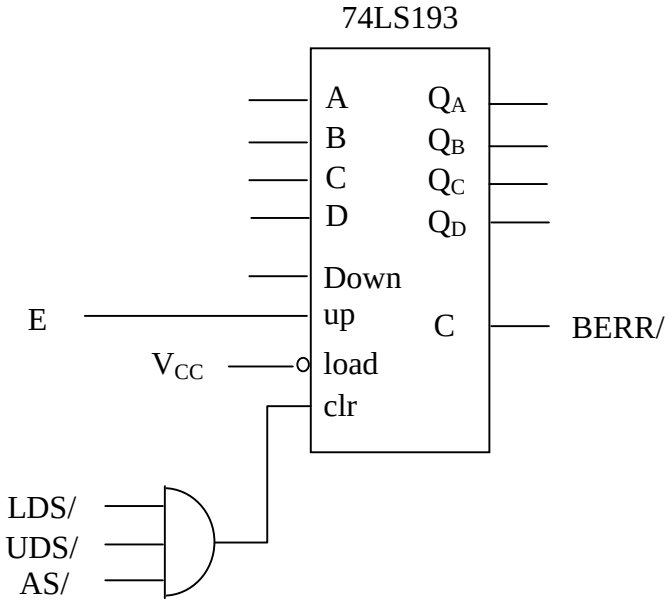


Figure V.7 : *Génération de BERR/.*

E : horloge de transfert synchrone, de fréquence $F_{CLK}/10$. Si on choisit CLK, l'erreur de bus intervient au bout de 15 coups d'horloge système. Si on choisit E, l'erreur de bus intervient au bout de 150 coup d'horloge système.

V.2.Echange synchrone

Les échanges synchrones existent sur le 68000 pour le rendre compatible avec les circuits de la famille 6800 :

6821 :	Port parallèle d'E/S
6840 :	Timer
6850 :	ACIA (liaison série)

6843 : Contrôleur de disque souple
6845 : Contrôleur de visualisation sur écran

V.2.1.Différents cycles de lecture et écriture

Processeur	Esclave
Démarrage d'un cycle classique	
	Décode les adresses Affirme la validation d'adresse périphérique $\overline{VPA}$
Contrôle de E jusqu'à l'état bas Affirme la validation d'adresse mémoire $\overline{VMA}$	
	Attente que E soit actif pour transférer les données.
Attend que E passe à l'état bas Infirme $\overline{VMA}$ Infirme $\overline{UDS}$, $\overline{LDS}$ et $\overline{AS}$	

Remarque : Les composants de la famille 6800 peuvent tourner jusqu'à 1Mhz et le 68000 jusqu'à 10Mhz. Pour que les temps de transferts soient compatibles, E est 10 fois moins rapide que CLK.

V.2.2.Génération du $\overline{VPA}$

Comme pour certains composants asynchrone, on peut être amené à générer le signal $\overline{VPA}$ à l'aide d'une logique externe.

VI.Gestion des données en pile

La gestion des données s'effectue à l'aide de :

variables : Une donnée correspond à une adresse.

Tableaux : On utilise un pointeur d'adresses A_n associé à un index D_m. On peut même rajouter un déplacement.
Ex : move.l $(A_n,D_m),D_p$.

Pile : Les données ne sont pas stockées à une adresse fixe mais selon l'ordre dans lequel elles sont utilisées.
La gestion est logicielle.
Deux méthodes sont utilisées : FIFO et LIFO.

VI.1.Principe

Une zone mémoire est réservée. Dans le cas du 68000, la pile utilisée étant de type LIFO, le pointeur d'adresses A_7, appelé Stack Pointeur (pointeur de pile), permet d'accéder aux données. Il pointe sur la dernière donnée écrtie ou lue.

T28 : Ex : Pile LIFO.

	9	←A7	9	
FD	2		2	
FE	1		1	
FF	0		0	
00 0000		Prédécrémentation		←A7

VI.2.Pile et sous-programme

La pile sert lors de l'appel à une sous programme et lors du retour au programme principal.

Appel à sous programme :

BSR <étiquette> Le compteur de programme (PC) est sauvegardé en pile avant d'être initialisé à l'adresse du sous programme.

Retour d'un sous programme :

RTS : Dépilement de PC pour le retour au programme principal.

RTR : Dépilement de PC pour le retour au programme principal et restitution de registre code condition..

```
Ex1 :   org     $FF9000              * début de programme
        lea     $FF9000,A7           * initialisation du pointeur de pile
        move.l  #$AABBCCDD,-(A7)     * empilement de données
        bsr     RIEN                     * appel au sous programme RIEN
        os9     F$Exit               * fin du programme principal
RIEN:                                * sous programme vide
        rts
```

Adresse	contenu	
F8	00	← après l'appel au sous programme
F9	FF	
FA	90	
FB	10	
FC	AA	← avant l'appel au sous programme
		↖ après le retour au programme principal
FD	BB	
FE	CC	
FF 8FFF	DD	
FF 9000		← initialisait de la pile

T29 : 2nd exemple.

F8	PC MSB	← SP après l'appel à SP1
F9	PC MSB	
FA	PC LSB	
FB	PC LSB	
FC	D6 MSB	← SP avant l'appel à SP1
		↖ SP après le retour de SP1
FD	D6 LSB	
FE	D5 MSB	
FF	D5 LSB	
FF 8000		← initialisation de SP

F6	SR MSB	← SP après la sauvegarde de SR
F7	SR LSB	
F8	PC MSB	← SP après l'appel à SP2
F9	PC MSB	
FA	PC LSB	
FB	PC LSB	
FC	D6 MSB	← SP avant l'appel à SP2
		↖ SP après le retour de SP2
FD	D6 LSB	
FE	D5 MSB	
FF	D5 LSB	
FF 8000		← initialisation de SP

VII.Mise en oeuvre des interruptions

VII.1.Principe

Une interruption est un événement **asynchrone** qui permet d'interrompre de déroulement normal d'un programme pour aller exécuter une tâche particulière.

Comparaison avec la vie courante :

Lecture = programme principal.

Interruptions possibles : quelqu'un sonne à la porte
la cuisson du gratin est terminée et le four sonne
le téléphone sonne
…

Gestion :	1- On termine la phrase en cours.	fin instruction
	2- On marque la page.	empilement du PC
	3- On va à la porte, au téléphone…	PC → ss-pgm d'IT
	4- On revient à sa lecture, à la bonne page.	retour au pgm princ

≠ avec un sous programme classique : L'action associée à un sous programme classique intervient à un **instant connu du déroulement du programme principal**. On peut donc appeler le sous programme. Par exemple, à la fin du 3^{ième} chapitre, je prévois de m'arrêter pour partir faire du vélo.

→ Problème particulier : Comment savoir le sous programme d'interruption à exécuter ? Il faut identifier l'interruption (four, téléphone…).

VII.2.Table des vecteurs d'interruption

Lorsqu'une interruption intervient (on dit aussi exception), un numéro de vecteur est généré.

En interne : Par exemple lors d'une erreur de bus ou une division par 0.
En externe : Dans le cas d'une interruption utilisateur.

T30 : A partir du numéro de vecteur, le processeur détermine l'adresse du sous programme d'interruption (traitement d'exception).

Détermination de l'adresse :

Cas du 68000 : Le processeur va chercher l'adresse correspondant au numéro de vecteur.

Ex : Erreur de bus .

Le processeur va chercher l'adresse du traitement à l'adresse $000008.

Cas du 68010 : Le processeur va chercher l'adresse correspondant au numéro de vecteur + le contenu du registre VBR (sur la carte CIAM).

Ex : Erreur de bus.

VBR = $FF8000

Le processeur va chercher l'adresse du traitement à l'adresse $FF8008.

T31 : Table des vecteurs d'interruption.

VII.3.Traitement d'une interruption

T32 : Ordre de traitement des interruptions :*

Reset
Interruption matérielle
Autre (logicielle)

VII.3.1.Reset

Activée soit :
- En externe par action sur la broche reset (mise à 0).
- En interne avec l'instruction reset. La broche reset est alors mise à 0 pour initialiser les autres composants.

Actions :
- Toutes les autres interruptions sont interdites. Le reset est l'interruption de plus haut niveau.
- SSP est initialisé à l'adresse définie par le vecteur 0 soit le contenu pointé par VBR. En cas de plantage du système, la pile peut être modifiée et gêner le fonctionnement de sous programme.
- PC est initialisé à l'adresse définie par le vecteur 1 vois VBR+4 ($FF8004 sur la carte CAIM).

VII.3.2.Interruption matérielle

Action sur les broches $\overline{IPL0}$, $\overline{IPL1}$ et $\overline{IPL2}$. Les vecteurs correspondants sont les n°25 à 31 et 64 à 255.

- Broches unidirectionnelles sur lesquelles sont placées le niveau de l'interruption.
  - 0 : Pas d'interruption. $\overline{IPL2}$ $\overline{IPL1}$ $\overline{IPL0}$ = 111.
  - 7 : Interruption de plus haut niveau. $\overline{IPL2}$ $\overline{IPL1}$ $\overline{IPL0}$ = 000.
- Interdiction des interruptions de niveau inférieur.
- Le niveau détermine le n° de vecteur (autovectorisé).
- Sauvegarde de PC et SR en pile.
- Initialisation de PC à l'adresse définie par le vecteur d'interruption.

  Ex : vecteur n°28.

  VBR + 70 $FF8070

VII.3.3.Autre

Idem que précédemment mais sans interdiction d'interruption.

VII.4.Interruptions matérielles autovectorisées

T33 :
- Elles sont provoquées par des composants en mode synchrone.
- Elles concernent les vecteurs 25 à 31 selon le niveau de l'interruption :

  Niveau 1 : Vecteur n°25.
  Niveau 7 : Vecteur n°31.
  Niveau 0 : Pas d'interruption.

- Priorités selon le niveau de l'interruption :
  Niveau 7 : Le plus prioritaire. $\overline{IPL2}$ $\overline{IPL1}$ $\overline{IPL0}$ = 000.
  Niveau 0 : Le moins prioritaire. $\overline{IPL2}$ $\overline{IPL1}$ $\overline{IPL0}$ = 111.
- Lors de la reconnaissance d'interruption, FC_0 FC_1 FC_2 = 111.
- Déroulement du traitement d'une exception :

Processeur	Périphérique demandeur
	➢ Demande d'interruption ($\overline{IRQ}$ à 0).
➢ Compare le niveau de l'IT avec le niveau défini dans le registre SR. Si le niveau de l'IT est supérieur à celui défini dans le registre SR, l'IT est traitée, sinon elle est ignorée. ➢ Attend la fin de l'exécution de l'instruction en cours. ➢ Place sur A1, A2, A3 le niveau de l'interruption en cours. ➢ Etablit R / $\overline{W}$ en lecture. ➢ Valide FC_0, FC_1 et FC_2 = 1. ➢ Valide $\overline{AS}$ à 0. ➢ Valide $\overline{LDS}$ et $\overline{UDS}$ à 0.	
➢ Génère le n° de vecteur ➢ Infirme $\overline{LDS}$, $\overline{UDS}$ et $\overline{AS}$. ➢ Affirme $\overline{VMA}$. ➢ Lance le traitement de l'exception (sauvegarde de PC…)	➢ Fournit un $\overline{VPA}$ pour lancer le mode autovectorisé.
	➢ Interruption acquittée : $\overline{IRQ}$ repasse à 1.

- Cas où deux interruptions arrivent simultanément. C'est la plus prioritaire qui est traitée. Dans notre cas, le composant 2 est plus prioritaire que le composant 1.
- Si aucune interruption n'est présente, le programme principal continue normalement.
- Rôle des résistances de rappel : Les sorties $\overline{IRQ}$ sont généralement à drain ouvert. Cela permet un ou câblé entre plusieurs composants générant une interruption de même niveau.

VII.5.Interruptions matérielles vectorisées

T34 :
- Dans le mode autovectorisé, le numéro de vecteur est généré par le processeur. Ici, c'est le composant demandant l'interruption qui fournit, sur le bus de données le numéro de vecteur.
- Ce type d'interruption concerne les vecteurs n°6 à 255.
- Déroulement du traitement d'une exception :

Processeur	
➢ Compare le niveau de l'IT avec le niveau défini dans le registre SR. Si le niveau de l'IT est supérieur à celui défini dans le registre SR, l'IT est traitée, sinon elle est ignorée. ➢ Attend la fin de l'exécution de l'instruction en cours. ➢ Place sur A1, A2, A3 le niveau de l'interruption en cours.	

- Etablit R / $\overline{W}$ en lecture.
- Valide FC_0, FC_1 et FC_2 = 1.
- Valide $\overline{AS}$ à 0.
- Valide $\overline{LDS}$ et $\overline{UDS}$ à 0.

- Lit le numéro de vecteur
- Infirme $\overline{LDS}$ et $\overline{AS}$.

- Lance le traitement de l'exception (sauvegarde de PC…)

Périphérique demandeur

- Demande d'interruption ($\overline{IRQ}$ à 0).

- Place le n° de vecteur sur D_0-D_7.
- Affirme $\overline{DTACK}$.

- Infirme $\overline{DTACK}$.

- Interruption acquittée : $\overline{IRQ}$ repasse à 1.

VII.6.Exemple de traitement d'une interruption

T35 : Qu'il s'agisse du mode vectorisé ou autovectorisé, la différence est transparente du point de vue du logiciel.

Mise en oeuvre :

- Initialisation des interruptions
  - Initialisation de VBR.
  - IT est l'étiquette donnant l'adresse du sous programme d'interruption. Cette adresse est écrite en $FF8070, correspondant à l'interruption autovectorisée de niveau 4.
  - Autorisation des interruptions. Les bits I_0, I_1 et I_2 de SR sont mis à 0 pour autoriser tous les niveaux d'interruption.
- Tâche de fond : boucle sans fin.

Que se passe-t-il en cas d'interruption de niveau 4 ?

1- PC et SR sont sauvegardés en pile pour le retour.

2- PC est initialisé à l'adresse définie par le vecteur n°28 soit PC=IT. Le programme est ainsi dérouté vers le sous programme d'interruption.

3- Interdiction des interruptions par masquage. Comme SR est sauvegardé, on aurait pu utiliser move.w #$2700,SR et ne pas le restituer.

4- Sauvegarde du contexte. Une interruption intervenant à n'importe quel moment, il faut absolument que le sous programme ne modifie pas les registres internes du µP.

5- Traitement de l'exception.

6- Restauration du contexte.

7- Autorisation des interruptions. Pas vraiment nécessaire.

8- Retour = restauration de SR et PC.