الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

5/89

وزارة التعليم العالي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT : ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

Conception et Réalisation d'un Kit Pédagogique autour du MPU MC 68000

Proposé par : Mme Hamami **Etudié par : A. CHEBIRA** **Dirigé par : Mme Hamami**
A. FARAH

PROMOTION : JUIN 89

P.A.O ASTEIN

E.N.P. 10 Avenue Hacène Badi - EL HARRACH - ALGER

الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT : ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

Conception et Réalisation d'un Kit Pédagogique autour du MPU MC 68000

sé par : Mme Hamami

Etudié par : A. CHEBIRA
A. FARAH

Dirigé par : Mr Hamami
Mme Hamami

PROMOTION : JUIN 89

A Nos Meres

pour leur patience Exemplaire

A Nos Peres

pour tous leurs Sacrifices

A Toute notre Famille et nos Ami(e)s

Nous Dedions ce Memoire

A. CHEBIRA & A. FARAH

REMERCIEMENTS

Nous tenons tout d'abord à exprimer notre profonde reconnaissance à Monsieur HAMAMI, qui nous a permis de réaliser cette étude en nous accueillant à l'ENSI (Direction de la recherche et du développement, et en nous orientant tout le long de notre travail.

Nos remerciements vont à Madame HAMAMI; non seulement protocolaires, ils sont l'expression de notre sincère considération et notre affectueuse reconnaissance pour l'admirable enseignement qu'elle nous a prodigué tout le long de notre formation.

Au terme de notre stage qui s'est déroulé dans le département HARDWARE de l'ENSI, et que nous terminons très satisfait et comblé d'une expérience fort enrichissante, nous tenons à exprimer notre profonde gratitude à la sympathique équipe du laboratoire péri-informatique notamment, Messieurs TOUATI Zinnedine, EL MANSALI Farid, BOUZA Nasser, REZZOUG Nourredine, CHAOUCH Mansour et TAGHLIT Abdellah, qui malgré leurs innombrables préoccupations, nous ont fait le grand honneur de nous aider à réaliser ce travail et nous faire profiter de leur grande expérience par leur conseils précieux. Leur sens du travail, nous a été un exemple quotidien.

Nous remercions également MR SAADOUN , Madame BEDEKE, MR et Mademoiselle CHEBIRA AMOR et ASSOUANE pour leur soutien inestimable .

Nous tenons enfin à saluer toute l'équipe ASTEIN , notament Monsieur CHAOUCH Nabil pour l'aide apportée quant à la réalisation de cet ouvrage.

Nous n'oublierons pas celui, qui, de près ou de loin, par sa présence ou en pensée, moralement ou matériellement a contribué un tant soit peu à cet aboutissement.

"Je rendrai à leurs enfants l'instruction que j'ai reçue de leurs pères"

HIPPOCRATE

SOMMAIRE

Notation utilisées:

△ : Roncontrer dans les schémas représente un résisistance"PULL UP"

∗: Remplace la notation habituelle BARRE (ex: VPA*= $\overline{\text{VPA}}$)

$ et H : Indiquent qu'un nombre est en héxadécimal.

INTRODUCTION

Si une partie de l'initiation aux microprocesseurs peut se faire sur de gros systèmes informatiques munis de programmes adéquats, tels qu'assembleurs, compilateurs et simulateurs, la partie pratique de celle-ci ne se conçoit valablement que sur du matériel réel.

Pour répondre à ce besoin, le kit d'initiation au microprocesseur MC68000 de MOTOROLA le FC68, qui a fait l'objet de ce travail permet au laboratoire calculateur du département électronique de l'école nationale polytechnique de s'équiper à moindre frais, puisqu'il a été évaluer à environ 2500 DA.

Destiné principalement à l'enseignement, le kit FC68 offre des possibilités pédagogiques bien adaptées à sa vocation de formation. Il permet à l'élève électronicien ou autre, d'avoir une vue assez complète sur le fonctionnement et la programmation des systèmes à microprocesseurs 16 bits en général et tout particulièrement ceux bâtis autour du MPU 68000, l'un des microprocesseurs les plus répondu actuellement sur le marché de la micro-informatique, et que l'on pourrait classer volontiers pour ces performances dans la catégorie des UC des "super micro" voire mini-ordinateurs par exemple: Station graphique HP9000, Macintosh d'APPLE etc..

Un premier contact avec le kit et ses spécifications techniques est introduit dans le premier chapitre du présent mémoire.

Le chapitre 2 donne 68000 réponses aux 32 questions que l'on peut se poser sur l'unité centrale du kit FC68.

Le chapitre 3 est un voyage au coeur du kit , avec description détaillée module par module, circuit par circuit s'appuyant sur les schémas électriques de la carte, ce qui intéressera sans nul doute ceux qui souhaiterait réaliser des extensions particulières et savoir comment sont générés les signaux de contrôle.

Le chapitre 4 est une aide au diagnostic des différentes pannes qui peuvent survenir éventuellement au cours des manipulations.

Pour essayer des programmes avec la bonne veille du MON-68K, moniteur du kit FC68 , des exemples d'utilisation de ce dernier sont décrits dans le dernier chapitre.

CHAPITRE 1.

PRESENTATION DU KIT

1.1 Etude d'un projet :

La première tâche dans la conception d'un système logique est toujours l'identification des besoins et contraintes du système (définition du cahier des charges).

L'aboutissement à un système final opérationnel répondant aux résultats attendus passe par différentes phases comme résumé dans l'organigramme de la Figure 1.1.

1.1.1- Identification des besoins :

Le cahier des charges relatif à l'étude et la réalisation du kit d'initiation au 68000 est le suivant:

-Système pédagogique construit autour du microprocesseur MC68000 de MOTOROLA

-Communications avec:

-un terminal clavier-écran via un port sériel. Standard RS232C

-un calculateur hôte ou imprimante via un second port sériel. Standard RS232C

-Vitesse de transmission variable.

-Fréquences d'horloge configurables.

-Reset manuel.

-Pas à pas HARD.

-Un "BREAK" manuel.

-Utilisation de RAM Statique (µPD 2167 - 16Kx1bit).

-Possibilité d'implantation d'un moniteur.

1.1.2 Plan d'action:

Afin de faire face aux contraintes des échéances lors de l'accomplissement d'un projet, il s'avère très utile et parfois indispensable de planifier son travail suivant un calendrier journalier, relatif aux différentes tâches à exécuter, et ce, pour aboutir aux résultats finaux dans des délais raisonnables. A cet effet, un programme d'action a été suivi lors de la conception, la réalisation et la documentation de ce projet de fin d'étude et est résumer dans le tableau ci-après (figure 1.2).

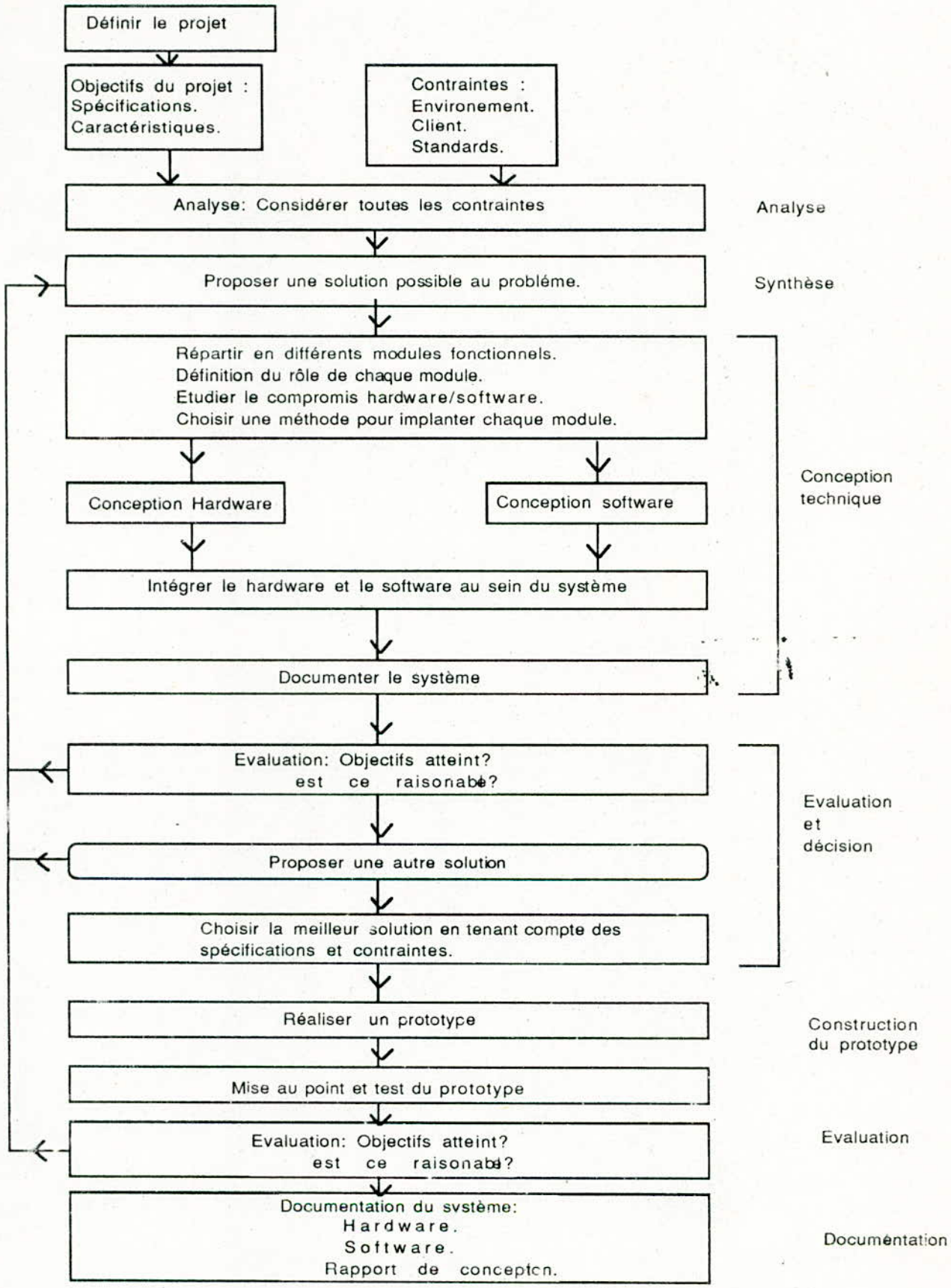


Figure 1.1 Organigramme de déroulement d'un projet

Tâche \date	Octobre	Novembre	Décembre	Janvier	Février	Mars	Avril	Mais	Juin
Recherche Bibliographique	**********	**********							
Etude théorique			**********	**********					
conception du sytème					**********	**********			
Réalisation et mise au point							**********	**********	
Etude et implantation du Moniteur							**********		
Documentation du système								**********	**********

Figure 1.2 Calendrier de travail

1.2 description générale du système:

Bâti autour du microprocesseur MC68000 de MOTOROLA, le système monocarte FC68 comprend entre autres, de la mémoire, 2 interfaces sériels de communication (compatible RS232C). Pour être opérationnel, il faut lui adjoindre une alimentation unique de 5 V(3 A maximum) et un terminal clavier-écran classique. Ce dernier peut être un terminal intelligent, relié à la carte par une liaison série RS232C à 9600 bauds maximum, cette configuration minimale permet la connexion d'une imprimante sérielle grâce au port 2.

De même, la connexion à un calculateur HOTE est intéressante pour le téléchargement des programmes et est prévue d'origine grâce à ce même port doté d'une RS232C supplémentaire (la vitesse de transfert est variable).

Cette carte permet donc de se constituer un environnement de développement et d'essai de programmes 68000/68010 sans aucune difficulté et sans avoir à rajouter des options non prévues.

Le matériel:

La carte supporte un microprocesseur 68000 qui travail à 8 MHz ce qui n'est pas la vitesse maximale permise par ce type de circuit mais qui n'est que peu d'importance dans ce cas particulier.

16 boîtiers de RAM sont montés sur la carte et offrent ainsi 32 KOctets à l'utilisateur ou 16 KMots.

2 boîtiers d'EPROM occupant 16 KOctets, contiennent le moniteur de la carte.

2 circuits d'interfaces ACIA MC6850 sont utilisés et permettent de disposer de 2 liaisons sérielles. Les niveaux RS232C sont des <<vrais>> et sont fabriqués sur la carte malgré l'utilisation d'une tension d'alimentation unique de 5 V grâce à des circuits MAX237 qui réalisent la conversion 5 V en +/-12 V.

Divers outils et artifices à caractères didactiques sont implantés afin de faciliter les manipulations sur la carte et le diagnostic des pannes.

Le logiciel:

Le moniteur qui équipe la carte FC68 permet quant à lui de manipuler la mémoire avec des commandes d'initialisation, de lecture par octet ou par bloc, de test, de modification.

Il permet en outre de tester des programmes avec un maximum de souplesse grâce à plusieurs modes d'exécution en pas à pas et la possibilité de pose de points d'arrêts. La visualisation ou la modification des registres du 68000 , la configuration de registres virtuels supplémentaires sont aussi possibles.

Un certain nombre d'autres commandes complètent ce tableau avec, notamment, la configuration du second port, la sauvegarde de programmes via ce port, la définition de macrocommandes etc...

La syntaxe de toutes ces commandes est classique et est analogue à celle que l'on rencontre sur les moniteurs des gros outils de développement. L'utilisateur du kit qui passe ensuite sur des machines <<réelles>> n'est donc pas dépaysé et peut très vite s'adapter.

Spécifications :

Processeur:	MC68000 ou MC68010
Horloge -CPU:	4 ou 8 MHz.
-Périphériques :	0.5, 1, 2 MHz
Mémoire:	8 pages mémoire de 32 KO adressables.
	16 ou 32 KO EPROM (paire de 2764 ou 27128).
	32 KO RAM statique (vecteur de 16x16Kbits.µPD2167)
Etats d'attente :	0 à 7 états d'attente, sélection par microSwitchs
Entrées/Sorties:	2 ports sériels compatibles RS232C avec baud rate réglable (9600,4800,2400,1200,600,300,150,110 bauds).
	Port1 terminal
	Port2 Host,MODEM,imprimante etc.
Interruptions autovectorisées:	Non masquable niveau 7.
	niveau 6, port2.
	niveau 5, port1
"Chien de garde":	Signalisation d'erreur bus après:
	5µs sous 8 MHz,10µs sous 4 MHz
LEDs témoins:	LED indiquant la mise sous tension.
	LED indiquant l'état HALT du processeur.
	LED d'auto-test.
	LED du pas à pas
Switchs:	Mise sous tension.
	Arrêt programme (interruption non masquable).
	Contrôle du pas à pas.
	Reset manuel.
	Sélection d'états d'attentes.
	Sélection de baud rate.
	Mode de contrôle imprimante sérielle
Cavaliers:	Activer/Inhiber le " chien de garde" .
	Validation de la fonction de démarrage.
	Reconfiguration de la carte mémoire.
	Inhiber le mode pas à pas matériel
Alimentation:	Monotension 5 V/3 A maximum
Température de fonctionnement:	0 à 50 °c
Coût:	Environ 2500 DA

CHAPITRE 2

ETUDE SOMMAIRE DU MC68000

Le MC68000 présente la troisième génération de microprocesseur chez MOTOROLA, une première génération a été marquée par le MC6800 le microprocesseur 8 bits, la deuxième par le MC6809 qui dispose de registres internes de 8 et 16 bits et offre quelques caractéristiques propres au microprocesseurs 16 bits.

Dans les années 80, les progrès de la technologie VLSI rendirent possible la fabrication de microprocesseurs 16 bits d'architecture avancée, dont le 68000, qui fait la synthèse entre le **NEC PLUS ULTRA** de la technologie, les techniques avancées de conception de circuits et les sciences informatiques .

Il est issu du projet **MACSS** (MOTOROLA ADVANCED COMPUTER SYSTEM ON SILICON), qui a été lancé dans le courant de l'année 1976, avec pour objectif d'aboutir à un microprocesseur monolithique dont les performances reposeraient sur deux principes qui sont:

- La simplicité

L'orthogonalité , i.e la banalisation des registres internes vis-à-vis des modes d'adressage et des instructions.

La technologie HMOS utilisée pour sa fabrication devrait permettre de réduire dans un rapport de 2 à 3 la surface d'une cellule élémentaire, de diviser le facteur qualité (produit de la consommation par la vitesse) entraînant de ce fait une consommation de 1 picojoule à 8 MHZ.

HARDWARE DU MC68000

2.1- Organisation interne

L'organisation interne du MC68000 est orientée comme le montre la figure 2.1, autour de 19 registres:

* 8 registres de données de 32 bits D0-D7.
* 7 registres d'adresses de 32 bits A0-A7.
* 1 pointeur de pile utilisateur de 32 bits A7 (User Stack Pointer USP)
* 1 pointeur de pile superviseur de 32 bits A7' (Supervisor Stack Pointer SSP).
* 1 compteur de programme de 24 bits (Program Counter PC).
* 1 registre d'état de 16 bits (Status Register SR).

2.1.1- Registre d'état:

Le registre d'état dont chaque bit a une signification bien particulière, constitue l'âme du processeur comme nous allons pouvoir en juger. Il se compose de 2 octets: un octet utilisateur et un octet superviseur (Figure 2.1)

Octet utilisateur: Seuls les 5 premiers bits de cet octet sont significatifs pour le programmeur. Existant sur la plus part des microprocesseurs 8 bits, ces indicateurs renseignent sur l'état du processeur après le traitement d'une instruction arithmétique ou logique.

superviseur "appelé sécurité système", S=0 mode utilisateur.

* Mode TRACE: bit 15, T=1 permet de "Tracer" un programme instruction par instruction. Fonction logicielle équivalente à l'opération pas à pas matériel.

* Masque d'interruption: Les bits I0, I1 et I2 (Interupt Priority Level). Le 68000

possède 7 niveaux d'interruptions programmables à travers les bits I0 I1 I2 (fig 2.2).

I0	I1	I2	NIVEAU D'INTERRUPTION
1	1	1	7 NON MASQUABLE
1	1	0	6
1	0	1	5
1	0	0	4
0	1	1	3
0	1	0	2
0	0	1	1
0	0	0	0 PAS D'INTERUPTION

FIG 2.2 Niveaux d'interruption

2.1.2- **Registres d'adresses A0-A6:**

Les 7 registres traitent des opérandes de taille égale à 16 bits (MOT) ou de 32 bits (LONG MOT) et peuvent être utilisés indifféremment comme pointeurs, pointeurs de piles, registres de bases ou registres d'indexe.

2.1.3- **Pointeurs de pile SP (Stack Pointer):**

Le registre d'adresse A7 tient un rôle particulier, en effet il est utilisé comme pointeur de pile. Par une indirection dépendant du bit S, il peut se dédoubler en 2 registres A7 (pointeur de pile utilisateur S=0) et A7' (pointeur de pile superviseur S=1).

2.1.4- **Registres de données:**

Ils acceptent des opérandes de taille égale à 8 , 16 ou 32 bits .

2.1.5- **Compteur de programmes:**

Ce registre contient 24 bits qui permettent au processeur d'adresser 16 MEGA octets.

2.2- Organisation matérielle du MC68000:

Le MC68000 est disponible dans un boîtier DIP (Dual in Line Package) de 64 broches sous 5 versions (4, 6, 8, 10, 12.5 MHZ). La figure 2.3 indique son brochage.

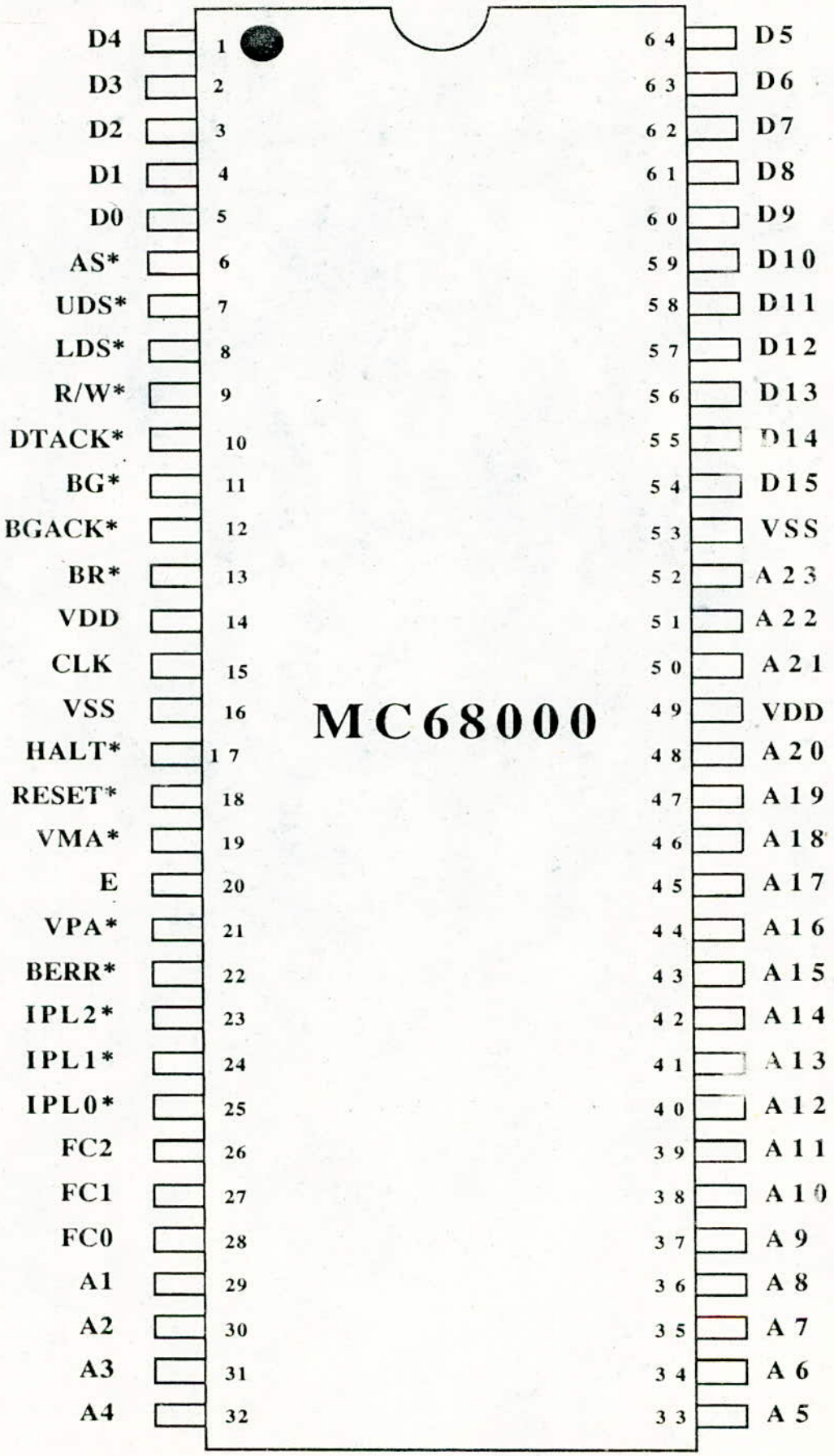


Fig 2.3 Brochage du MC68000

Les signaux du MC68000 peuvent par ailleurs être organisé en plusieurs groupes :

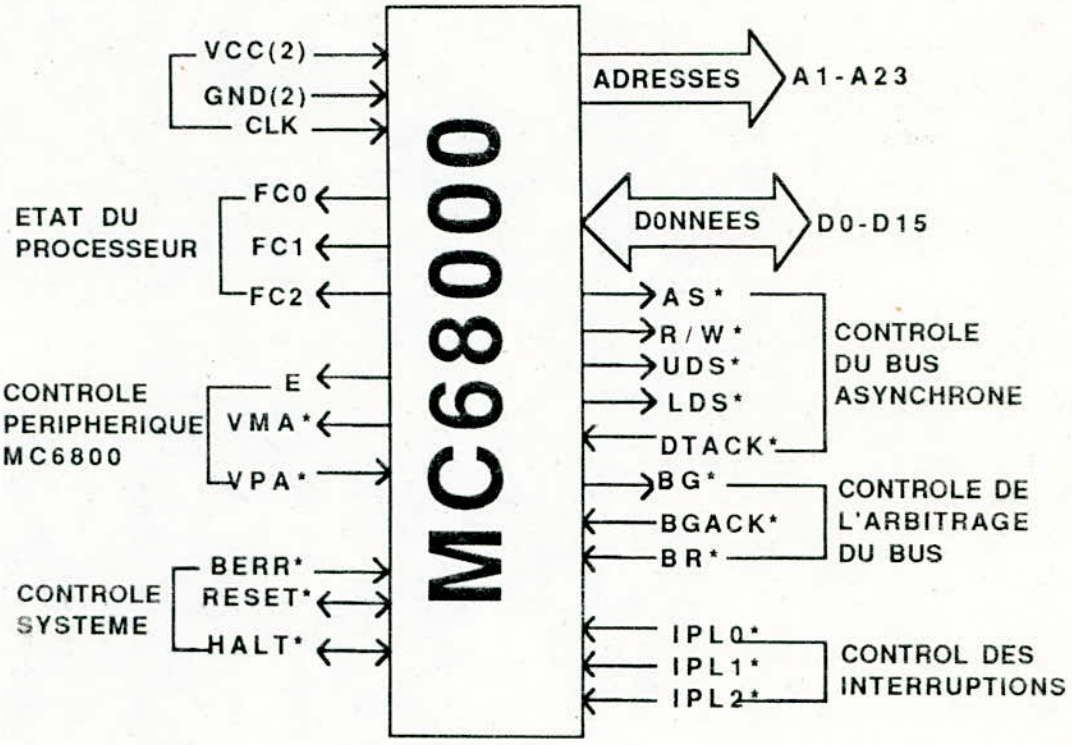


Fig 2.4 Les Fonctions du MC68000

2.2.1- **Bus de données:** (D0-D15)

Logique 3 états. C'est un bus de 16 lignes bidirectionnelles non multipléxé, qui peut transférer des donnés sur des longueurs de 16 et 8 bits afin d'assurer une compatibilité matérielle avec la famille des circuits 6800.

2.2.2- **Bus d'adresses:** (A1-A23)

Logique 3 états. Ce bus de 23 lignes unidirectionnelles est capable d'adresser 16 millions d'octets. Notons que le 68000 ne dispose pas de ligne d'adresse A0, qui est remplacé par les deux signaux UDS* et LDS*, signaux générés par le 68000 en fonction du type des données adressées (bit A0 est interne).

2.2.3- **Commandes du bus asynchrone:** AS*, R/W*, UDS*, LDS* ET DTACK*

Les transferts asynchrones de données sont réalisés en utilisant les signaux de commandes suivants:

a) Echantillonnage d'adresses AS* (Address Strobe): Logique 3 états. Ce signal indique qu'il y a une adresse valide sur le bus d'adresses.

b) Lecture/écriture R/W* (Read/Write): Logique 3 états. Ce signal définit le type de transfert sur le bus de données comme étant une lecture (R/W*=1) ou une écriture (R/W*=0)

c) Echantillonnage de données haut et bas UDS* et LDS*(Uper & Lower Data Strobe):

Logique 3 états. Ces deux signaux en combinaison avec R/W* indiquent la lecture ou l'écriture d'un mot ou d'un octet. Voir tableau(figure 2.5).

UDS*	LDS*	R/W*	D8-D15	D0-D7	OPERATION
H	H	-	non valides	non valides	-
L	L	H	valides	valides	lecture d'un mot
H	L	H	non valides	valides	lecture d'un octet impair
L	H	H	valides	non valides	lecture d'un octet pair
L	L	L	valides	valides	écriture d'un mot
H	L	L	recopie bits 0-7	valides	écriture d'un octet impair
L	H	L	valides	recopie bits 8-15	écriture d'un octet pai r

Fig 2.5 Tableau des combinaisons de R/W*,UDS*,LDS*

d) Reconnaissance de transfert de données DTACK*(Data Transfert ACKnoledge): Logique 3 états. Ce signal apporte au microprocesseur l'information suivante:

**En lecture:* Les données à lire sont disponibles.

**En écriture:* Les données ont été prises en compte par le circuit externe.

2.2.4- **Contrôle d'arbitrage du bus** : BR*, BG* et BGACK*

Ces trois signaux assurent l'arbitrage du bus.

a) Demande du bus BR* (Bus Request): Cette entrée à l'état bas, informe le processeur qu'un dispositif externe réclame le bus.

b) Cession du bus BG* (Bus Grant): Ce signal indique au circuit demandant le bus que le 68000 à pris en compte sa requête et alerte son entourage qu'il cédera son bus à la fin du cycle en cours.

c) Reconnaissance d'allocation du bus BGACK* (Bus Grant ACKnoledge): Confirmation de la part du circuit demandeur de la prise du bus et son contrôle. Cette ligne ne peut être validée que si les conditions suivantes sont réunies:

1. BG* affirmé (BG*=0).
2. AS*=1, DTACK*=1, BGACK*=1 (infirmés).

2.2.5- **Etats du processeur**: FC0,FC1,FC2 (Fonction Code)

Logique 3 états. Ces 3 sorties indiquent l'état du processeur, c'est à dire s'il est en mode superviseur ou en mode utilisateur et quel est le type de cycle en cours d'exécution (voir figure 2.6). Ces lignes présentent donc une sécurité pour le système tout en offrant la possibilité d'accroître la capacité d'adressage de 16 à 64 Mega octets.

FC2	FC1	FC0	TYPE DU CYCLE
L	L	L	réservé
L	L	H	données utilisateur
L	H	L	programme utilisateur
L	H	H	réservé
H	L	L	réservé
H	L	H	données superviseur
H	H	L	programme superviseur
H	H	H	reconnaisance d'interruption

Fig 2.6 Tableau des codes de fonction

2.2.6- **Demande d'interruption** : IPL0-IPL2(Interupt Priority Level)

Ces trois lignes permettent au microprocesseur de savoir qu'un périphérique demande une interruption dont le niveau est indiqué par une valeur codée et placée sur ses entrées. Le niveau 7 a la plus haute priorité, alors que le niveau 0 indique qu'aucune interruption n'est en attente.

2.2.7- **Contrôle du système** : RESET*, HALT* et BERR*

Ces lignes sont utilisées pour initialiser, arrêter le processeur ou indiquer que des erreurs sont survenues au cours d'un échange. Ces signaux sont:

a) BERR* (Bus ERRor): Cette entrée peut être affirmée par une logique externe pour informer le processeur que des problèmes sont survenus dans le cycle de bus en cours d'exécution. Par exemple:

-Cas d'un périphérique ne répondant pas au processeur.

-Absence de DTACK* au cours d'opération de lecture ou écriture en mémoire après un délai fixé par le concepteur.

b) HALT*: Ce signal est bidirectionnel

en entrée:

-En combinaison avec le signal RESET*, il assure l'initialisation du 68000.

-Seul, lorsqu'il est affirmé par un circuit externe arrête le MPU à la fin du cycle bus en cours.

en sortie: Il informe la périphérie que le processeur est à l'arrêt. Seul une action sur la broche RESET* permet de quitter l'état HALT.

c) RESET*: Ce signal est également bidirectionnel.

en entrée: Il y a initialisation du processeur.

en sortie: On effectue une initialisation de tous les circuits externes grâce à l'exécution de l'instruction RESET. Dans ce cas l'état du processeur n'est pas affecté.

2.2.8- **Commande des périphériques 6800** : VMA*, VPA*, E

Ces signaux de commande permettent l'interfaçage des circuits synchrones de la famille 6800 avec le circuit asynchrone 68000.

a) E (Enable): Ce signal périodique généré à partir d'une horloge interne flottante représente le signal standard de validation des circuits 6800. La période de celui-ci est égale à

10 périodes de l'horloge d'entrée du MC68000 avec un rapport cyclique de 60/40.

b) Validation d'adresse périphérique VPA* (Valid Peripheral Address): Cette entrée indique que le transfert de données sera synchronisé sur E car un périphérique désire converser avec le 68000 de manière synchrone. Elle indique également que le processeur sera en mode autovectorisé en cas d'interruption.

c) Validation adresse mémoire VMA* (Valid Memory Address): Dés la réception du signal VPA* le processeur se synchronise avant de confirmer l'échange de dialogue avec le périphérique 6800 par la mise à zéro de la sortie VMA* lorsque l'horloge E passe à l'état bas.

2.2.9- **Les alimentations: (2) VCC et (2) GND:**

Le MC68000 nécessite une alimentation unique de 5V. On note que les broches VCC et GND sont au nombre de deux, cela afin de subvenir aux alimentations des boîtiers externes, sans pertes importantes.

2.2.10 **L'horloge CLK:**

L'entrée CLK doit être un signal compatible TTL stable, symétrique et doit suivre les caractéristiques données par le constructeur (voir fig 2.7) . La forme d'onde de l'horloge est représentée dans la Figure 2.8.

CARACTERISTIQUES	SYMBOLE	MC68000 L10		UNITES
		MIN	MAX	
Fréquence	F	2	10	MHz
Temps de cycle	tcyc	100	500	ns
Temps de monté	tm	-	10	ns
temps de descente	td	-	10	ns
Largeur d'impultion				
etat haut	tch	45	250	ns
etat bas	tcb	45	250	ns

Fig 2.7 Caractéristiques de l'horloge du 68000

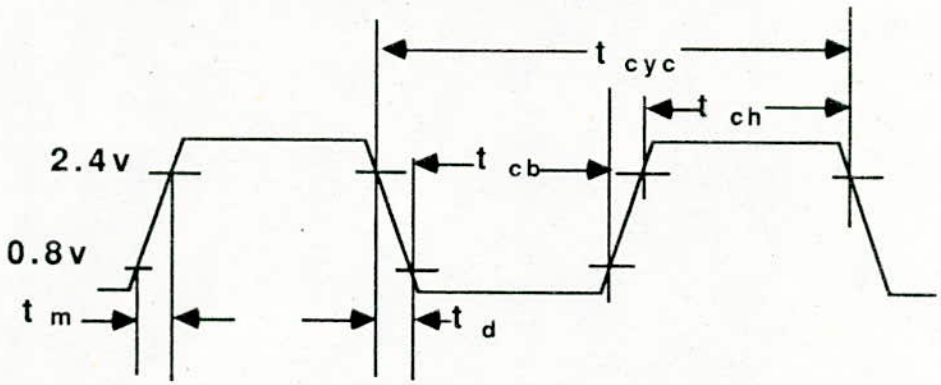


Fig 2.8 Forme d'onde de l'horloge

2.3 Fonctionnement du bus:

Les paragraphes suivants visent à éclairer le concepteur d'application à base du MC68000 sur le fonctionnement de son bus pendant les opérations de transfert asynchrone de données et le

dialogue avec les circuits synchrones.

2.3.1- Echanges asynchrones:

Pour minimiser les problèmes dus aux différents temps d'accès des circuits, le MC68000 dialogue avec les boîtiers périphériques selon le mode d'échange asynchrone. Cela permet de détecter les erreurs en cours d'échange et de n'imposer aucune contrainte de temps durant le cycle bus.

a) Cycles de lecture: Pendant un cycle de lecture, le processeur reçoit des données de la mémoire ou d'un périphérique. Il lit des octets de données dans tous les cas et utilise pour cela le bit interne A0 pour préciser la voie que doit emprunter la donnée. Si A0 est égal à zéro alors UDS*=0 et LDS*=1(octet pair), si A0=1 alors UDS*=1 et LDS*=0(octet impair).

Si l'instruction spécifie une opération sur un mot, le processeur lit les 2 octets placés sur le bus, dans ce cas UDS*=0 et LDS*=0. Un organigramme et un chronogramme du cycle de lecture d'un mot sont données en figure 2.9 et 2.10.

Notons que si le signal DTACK* n'arrive pas avant l'état S4 (au set up prés) le processeur insère des états d'attente (WAIT STATES) dans le cycle du bus (fig 2.11), les états d'attente sont introduits par paires . Si le signal DTACK* n'intervient pas du tout il est utile de prévoir une logique externe (chien de garde) permettant de générer le signal approprié BERR*.

Les circuits mémoires et périphérique ne sont pas tous conçus pour générer de manière automatique le signal DTACK*, pour palier à ce probième, on utilisera une électronique externe qui va conter les tops d'une horloge à partir du moment où les signaux AS*,UDS* et LDS* sont affirmés , ce pendant une durée qui est fonction du temps d'accès du circuit.

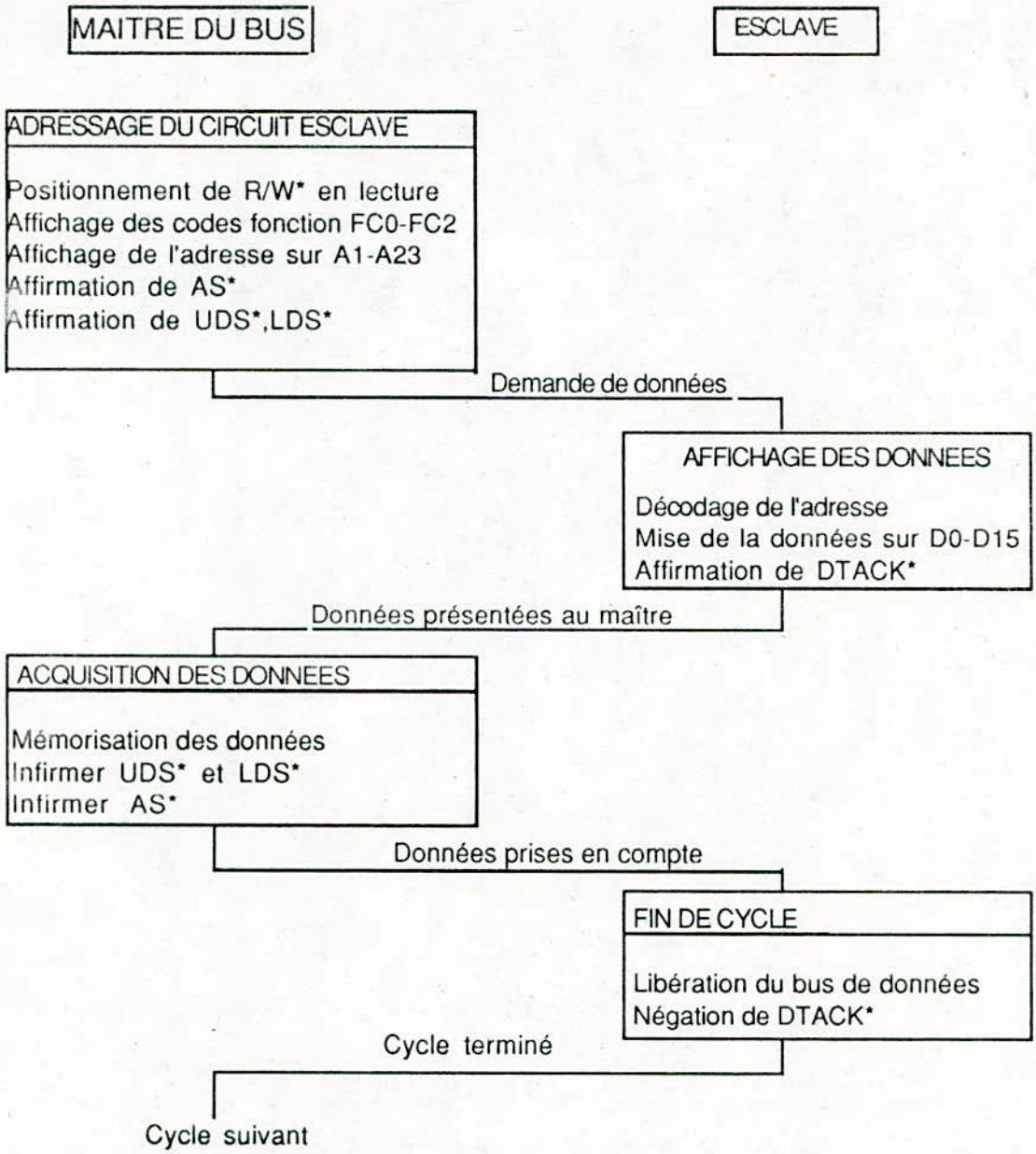


Fig 2.9 Organigramme du cycle de lecture d'un mot

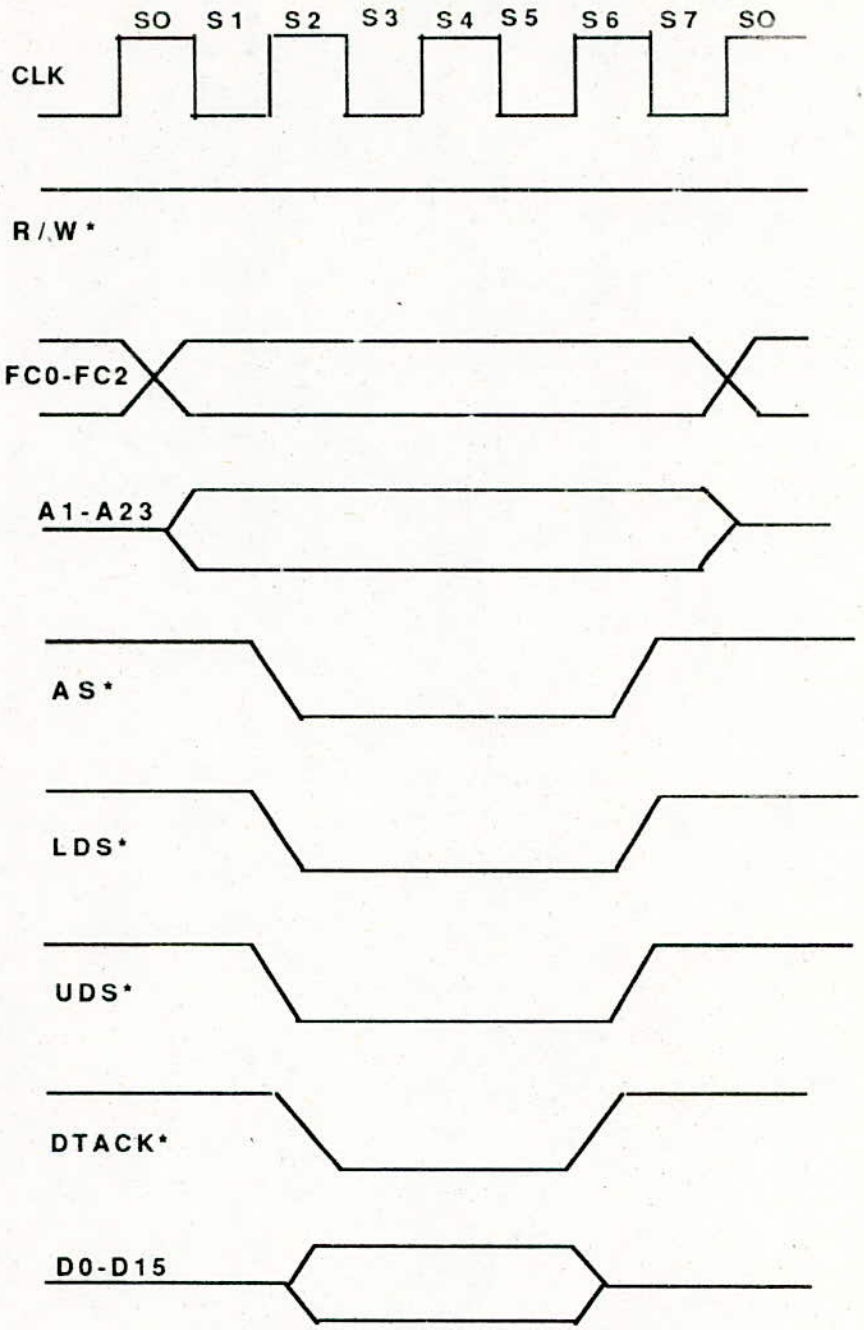


fig 2.10 Chronogramme de lecture d'un mot

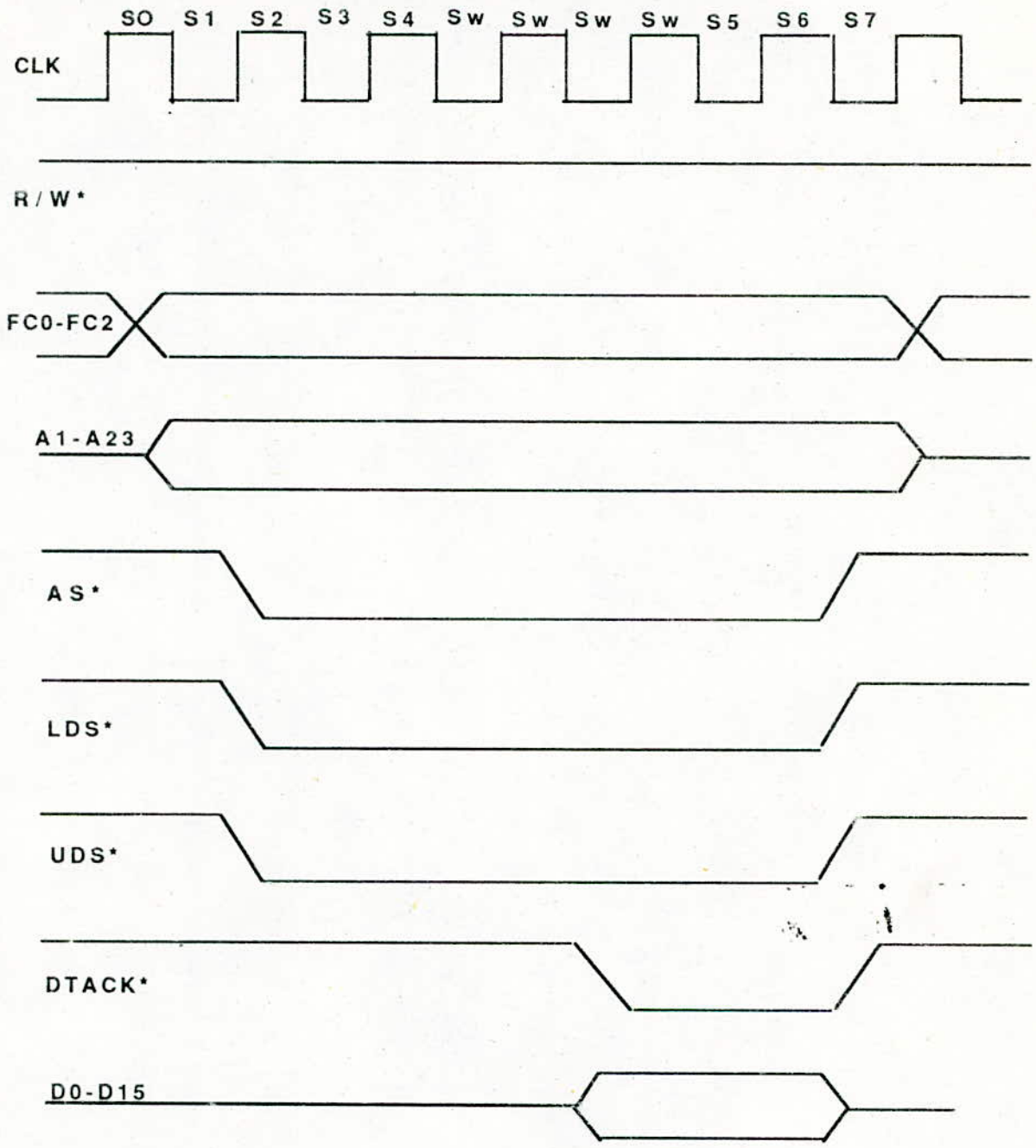


Fig 2.11 Chronogramme de lecture avec insertion d'états d'attente

b) Cycle d'écriture: Lors d'un cycle d'écriture, le processeur envoie des données à une mémoire ou à un périphérique. Comme pour l'opération de lecture, le MC68000 écrit des octets, bien sûr si l'opération spécifie un opérande de type mot ou long mot, il écrit respectivement 2 et 4 octets.

La structure d'un cycle d'écriture est décrite à l'aide de l'organigramme et du chronogramme des figures 2.12 et 2.13.

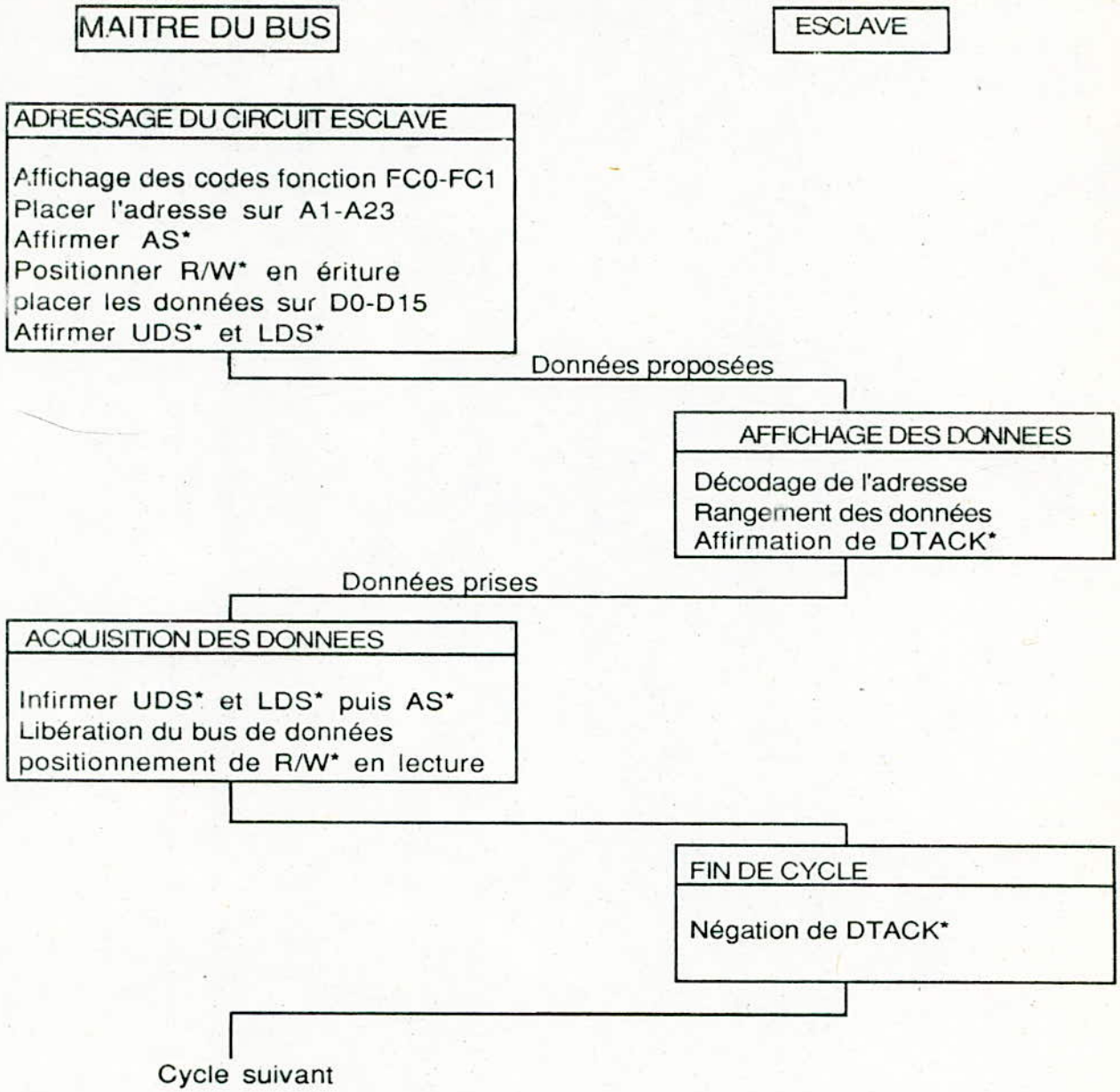


Fig 2.12 Organigramme d'écriture d'un mot

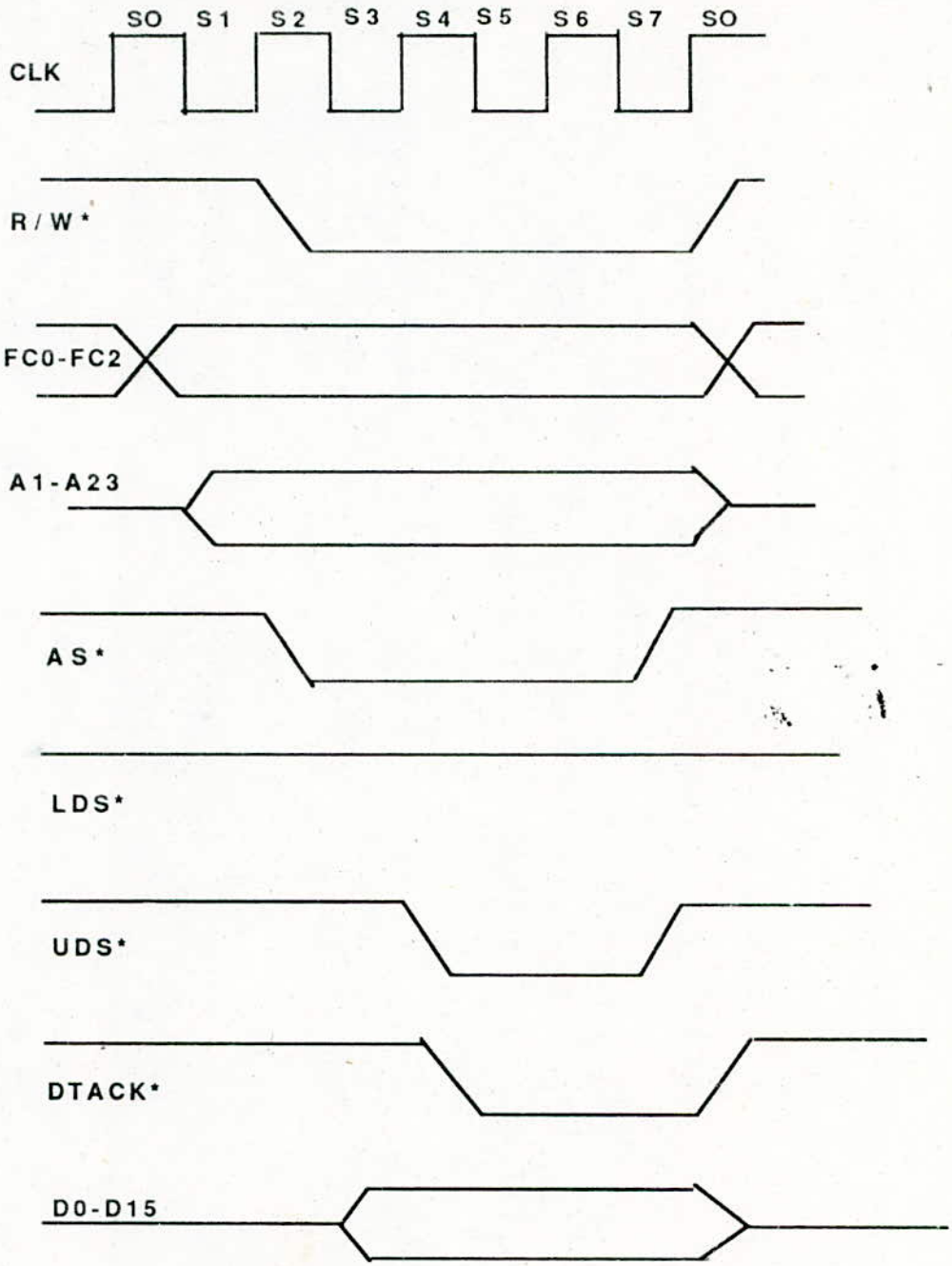


fig 2.13 Chronogramme d'écriture d'un octet

2.3.2- Echanges synchrones:

Contrairement à l'échange asynchrone, l'échange synchrone entre le circuit maître et le circuit esclave est piloté par l'horloge E générée par le MC68000; tout signal doit survenir obligatoirement dans une plage de temps bien définie par ce dernier, ce dialogue est schématisé par l'organigramme de la figure 2.14.

Trois lignes du MC68000 assurent l'interfaçage avec les circuits périphériques de la famille 6800:

-L'horloge E

-Le signal de validation d'adresse périphérique: VPA*

-Le signal de validation d'une position mémoire: VMA*

Le signal de validation E correspond à l'horloge ø2 des systèmes 6800. (E=1/10 de la fréquence injectée sur l'entrée clock du 68000).

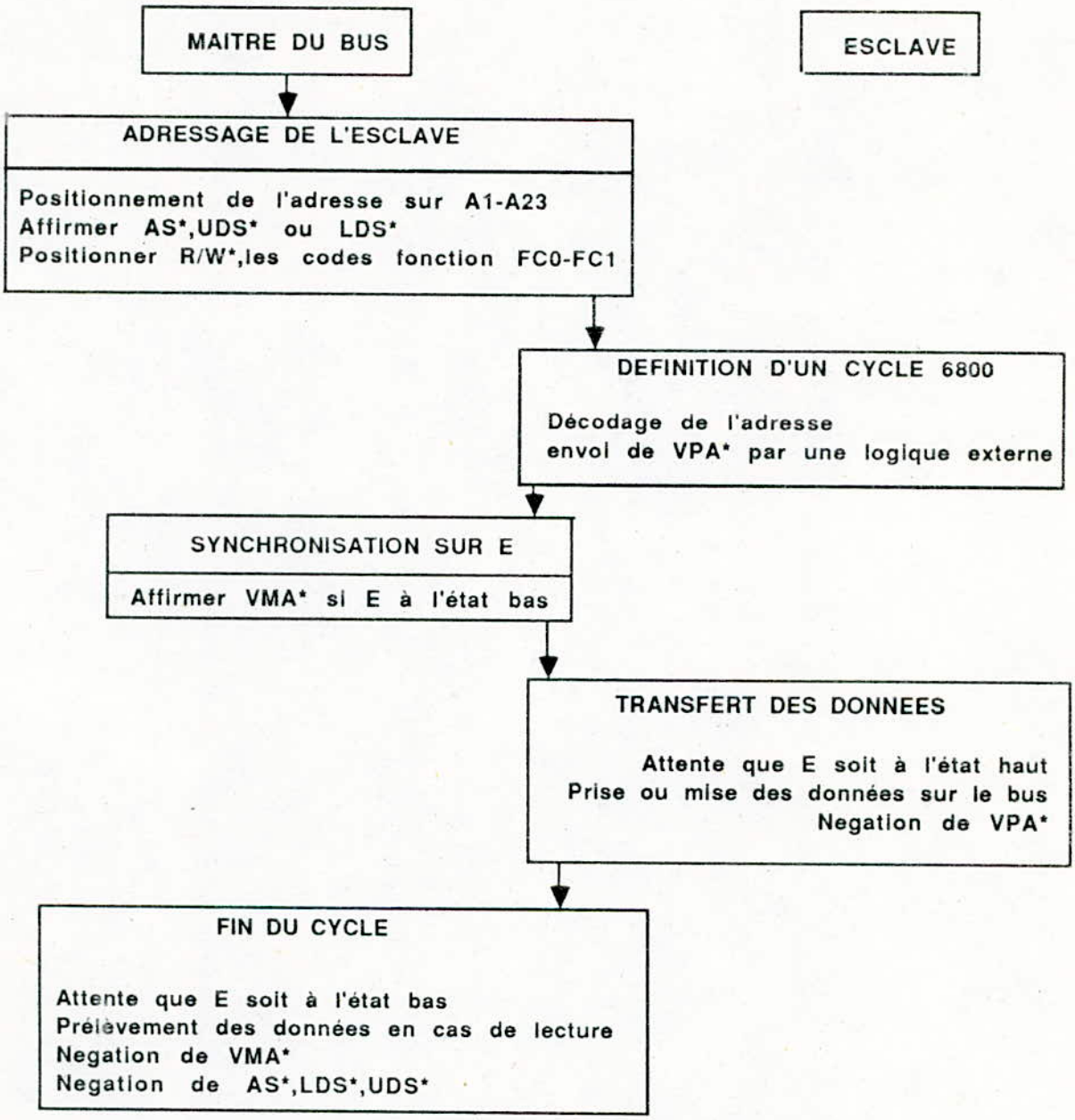


Fig 2.14 Organigramme de l'échange synchrone

2.4-Les exceptions:

On appelle exception le changement de tâche du microprocesseur due le plus souvent à des événements internes (logiciels) ou externe (matériels). Les exceptions générées de façon externe sont les interruptions, les demandes d'initialisation et les erreurs bus, celles générées de façon interne proviennent d'instructions spécialisées , d'erreurs d'adresse ou du mode trace. Les instructions trappe (TRAP,TRAPV), test registre aux limites (CHK) et division (DIV) peuvent

toutes générer des exceptions comme faisant partie de leurs exécution. A chaque exception correspond un numéro de vecteur qui est un nombre de 8 bits. Ce dernier, multiplié par 4 donne l'adresse du vecteur d'exception.

La table d'exceptions contient 255 vecteurs de 32 bits, elle est située en mémoire basse , de l'adresse 0 à 3FFH et est représentée sur la figure 2.15.

ADRESSE DEC---HEX	MUMERO DU VECTEUR	EXEPTION CONCERNEE
0-------0	0	RESET:INITIALISATION DE SSP
4-----004		RESET:INITIALISATION DE PC
8-----008	2	ERREUR BUS
12----00C	3	ERREUR ADRESSE
16----010	4	INSTRUCTION ILLEGALE
20----014	5	DIVISION PAR ZERO
24----018	6	INSTRUCTION CHK
28----01C	7	INSTRUCTION TRAPV
32----020	8	VIOLATION DE PRIVILEGE
36----024	9	TRACE
40----028	10	EMULATEUR LIGNE 1010
44----02C	11	EMULATEUR LIGNE 1111
48----030	12-14	RESERVE
59----03B		-
60----03C	15	INTERRUPTION NON INITIALISEE
64----040	16-23	RESERVE
95----05F		-
96----060	24	INTERRUPTION PARASITE
100---064	25	AUTO-VECTEUR NIVEAU 1
104---068	26	AUTO-VECTEUR NIVEAU
108---06C	27	AUTO-VECTEUR NIVEAU 3
112---070	28	AUTO-VECTEUR NIVEAU 4
116---074	29	AUTO-VECTEUR NIVEAU 5
120---078	30	AUTO-VECTEUR NIVEAU 6
124---07C	31	AUTO-VECTEUR NIVEAU 7
128---080	32-47	VECTEUR INSTRUCTION TRAP
191---0BF		-
192---0C0	48-63	RESERVE
255---0FF		-
256---100		VECTEUR INTERRUPTION UTILISATEUR
1023--3FF		-

Fig 2.15 Table d'exception

Le traitement d'une exception par le 68000 s'effectue en 4 étapes comme explicité ci-après:

Première étape:

*Une copie du registre d'état en cours est faite dans un registre interne au processeur.

*Le bit S est ensuite affirmé (passage au mode superviseur).

*Le bit T est infirmé (exécution de l'interruption se fait en mode normal).

*Si l'exception est de type interruption ou RESET, le masque de priorité du registre d'état est mis au niveau de l'interruption en cours (7 pour le RESET).

Deuxième étape: Détermination du numéro du vecteur d'exception.

*Pour les interruptions, le numéro est fourni au processeur au cours de la phase de reconnaissance de l'interruption.

*Pour les autres types d'exception, dont les interruption vectorisées, la logique interne du processeur délivre le numéro de vecteur.

Troisième étape: Il y a sauvegarde dans la pile superviseur du compteur de programme et la copie sauvegardée du registre d'état. La valeur du compteur de programme contient généralement l'adresse de l'instruction suivante qui aurait dû être exécutée sans l'exception sauf dans le cas d'une erreur bus ou d'adresse dans quel cas la valeur du PC mise dans la pile est imprévisible. Toutefois, pour les erreurs de bus et d'adresse des informations supplémentaires sont empilées dans la pile superviseur.

Quatrième étape: Le contenu du vecteur d'exception est chargé dans le compteur de programme. Le processeur reprend alors l'exécution des instructions adressées par le PC.

Les différents types d'exceptions qui peuvent survenir lors du fonctionnement du système sont:

a- Les interruptions.

b- Le RESET(initialisation).

c- instructions de type TRAP.

d- Instructions illégales ou non implantées.

e- Violation de privilège.

f- Le mode trace.

g- L'erreur bus.

h- l'erreur d'adresse.

i- L'interruption non initialisée.

Dans ce qui va suivre on va étudier trois types d'exception qui sont: RESET, interruption, erreur bus.

2.4.1-Le RESET:

A la mise sous tension on obtient une mise à zéro générale du système en affirmant RESET* et HALT* pendant au moins 100 ms.

Le 68000 lit alors la table des vecteurs à l'adresse $00000000 correspondant au numéro de vecteur 0, son contenu est chargé dans le pointeur de pile superviseur. Le processeur lit, ensuite le contenu de l'adresse $000004, le charge dans le compteur de programme et commence alors à exécuter la première instruction.

Si le processeur était déjà sous tension, la séquence d'exception RESET est générée si les

signaux RESET* et HALT* sont affirmés pendant 10 cycles d'horloge.

Durant la phase d'initialisation, l'exécution des instructions se fait en mode superviseur, le bit T est forcé à zéro pour mettre hors fonction le mode TRACE. Si une erreur bus intervient pendant cette phase le processeur s'arrête et tout traitement cesse.

2.4.2-**L'erreur bus:**

Une erreur de bus est détectée par une logique externe qui en informe le processeur en validant l'entrée BERR*.

La réception de BERR* fait rompre le cycle bus en cours, le contexte à sauvegarder est plus complexe. La séquence du traitement de l'erreur bus est la suivante:

*Copie du registre d'état.

*positionnement des bit S à 1 et T à 0.

*Génération interne du numéro de vecteur.

*Sauvegarde du compteur programme et registre d'état. la valeur sauvegardée du PC est augmentée de 2 à 10 octets (phénomène d'anticipation).

*Sauvegarde du registre d'instruction.

*Sauvegarde de l'adresse pour laquelle s'est produite l'anomalie.

Sauvegarde de l'état de R/W.

*Sauvegarde des codes de fonctions.

A l'aide de toutes ces informations il est possible d'établir un diagnostic logiciel. Si une autre erreur survient pendant le traitement de l'exception il y a double faute et le processeur s'arrête. Seul un RESET peut le redémarrer.

2.4.3-**Les interruptions:**

Un circuit externe ou périphérique peut interrompre le processeur par l'intermédiaire des lignes IPL0-IPL2. La demande d'interruption est donc faite en codant le niveau de la requête sur les lignes IPL0-IPL2, suivant 7 niveaux de priorité. Le niveau 7 a la plus haute priorité, il est sensible aux fronts du signal appliqué, et est non masquable, il est muni d'un réarmement interne après le début du programme d'exception; ce qui signifie que toute interruption de niveau 7 peut interrompre une interruption de niveau 7.

Les niveaux 1 à 6 sont masquables et sensibles aux niveaux. Lorsqu'une interruption est perçue, son niveau est comparé au masque du registre d'état, si le niveau est **inférieur ou égal** au masque, l'interruption est inhibée.

Si la priorité de l'interruption en attente est supérieure à celle en cours, la séquence de traitement est la suivante:

*Copie du registre d'état.

*Le bit S est mis à 1 et le bit T est forcé à 0.

*Le masque du registre d'état est mis au niveau de l'interruption prise en compte.

*Le 68000 recherche alors le numéro de vecteur, entame le cycle de

reconnaissance de l'interruption et affiche le niveau d'interruption sur les lignes d'adresses A1-A3. Deux cas peuvent se présenté alors:

a) Vectorisation automatique:

Une logique externe demande une vectorisation automatique par l'affirmation du signal VPA*. Le 68000 génère alors le numéro de vecteur de manière interne en fonction du niveau d'interruption.

b) Vectorisation non automatique:

Une logique externe doit fournir dans ce cas le numéro du vecteur , le placer sur le bus de donnees D0-D7 et terminer son transfert en affirmant DTACK*.

Dans les deux cas si une logique externe affirme BERR*, cela signifie qu'il y a une interruption parasite. Le processeur génèrera alors le numéro de vecteur de l'interruption parasite.

Récapitulons le processus de traitement d'une exception par l'organigramme de la figure 2.16.

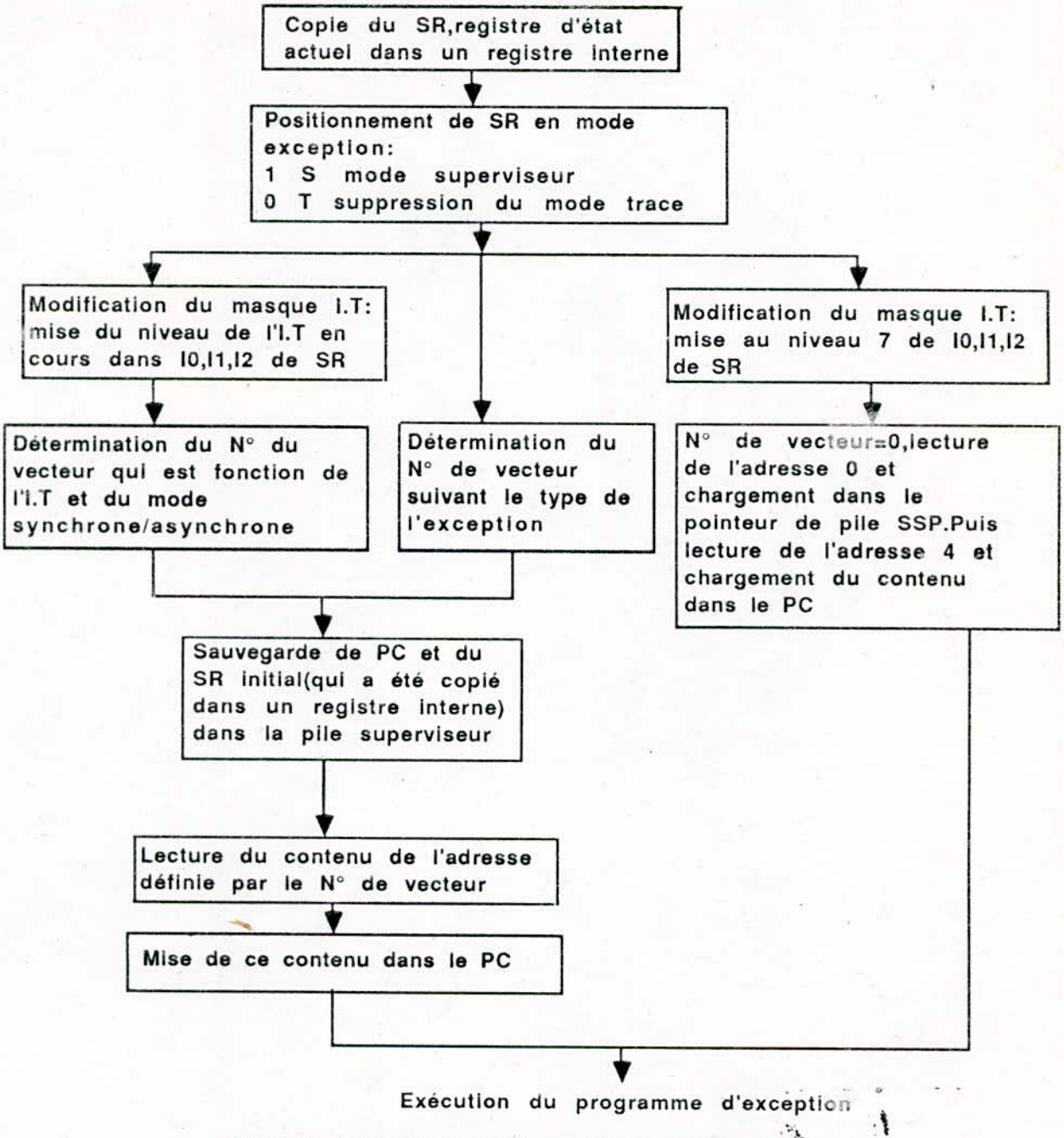


Fig 2.16 Principe général du traitement des exceptions

SOFTWARE DU MC68000

2.5- Modes d'adressage:

Définition: On appel modes d'adressage, les différents moyens que possède une instruction pour accéder à une adresse, afin d'y effectuer un traitement.

Le MC68000 possède 14 modes d'adressage différents qui permettent, malgré le nombre limité (56) d'instructions, de le rendre extrêmement puissant. On peut regrouper les 14 modes

d'adressage en six catégories:

* Adressage registre direct
* Adressage registre indirect
* Adressage absolu
* Adressage immédiat
* Adressage relatif
* Adressage implicite

2.5.1- **Adressage registre direct:**

Dans ce mode, l'adresse effective de l'opérande est un des registres de données. Pour illustrer, on prendra l'exemple suivant:

CLR.B D1

CLR: instruction clear (effacement).

B : Désigne la taille de l'opérande(BYTE: Octet).

D1 : Opérande (Registre de données D1).

Cette instruction initialise les huit bits de poids faible du registre de données D1, elle peut traiter des opérandes de taille 8,16 ou 32 bits. Alors que les instructions CLR.W D1 et CLR.L D1 initialisent respectivement les 16 et 32 bits du registre D1.

Remarque: W désigne un opérande de taille 16 bits (WORD:MOT).

L désigne un opérande de taille 32 bits (LONG WORD: LONG MOT).

2.5.2- **Adressage direct (registre d'adresse):**

Dans ce mode l'adresse effective de l'opérande est le registre d'adresse. Exemple:

MOVEA.W A0,$1000

Cette instruction transfert les 16 bits de poids faible du registre d'adresse A0 vers l'emplacement mémoire 1000H . La taille de l'opérande ne peut être que le mot ou le long mot. On pourra écrire l'instruction MOVEA.W $1000, A0 qui fait le transfert en sens inverse.

2.5.3- **Adressage registre indirect:**

Ce mode rassemble 5 types de sous modes qui sont:

a) Adressage registre indirect: L'adresse effective de l'opérande est le contenu du registre d'adresse désigné. Exemple:

CLR.B (A5) ( ):contenu de, supposons que (A5)=$F010.

Cette opération initialise les huit bits dont l'adresse est $F010. Si la taille désignée est le mot ou le long mot, l'adresse contenue dans le registre d'adresse doit être paire.

b) Adressage registre indirect avec post-incrémentation: L'adresse de l'opérande est le contenu du registre d'adresse désigné qui est ultérieurement incrémenté de 1,2 ou 4 suivant que la taille de l'opérande est l'octet le mot ou le long mot. Exemple:

CLR.L (A0)+ ; A0=$AABC

Cette instruction met à zéro les octets d'adresse $AABC, $AABD, $AABE et $AABF (32

bits) après quoi le contenu de A0 sera $AABC+4=$AAC0. De même si la taille précisée est le mot ou le long mot le contenu du registre d'adresse doit être paire.

c) Adressage registre indirect avec prédécrémentation: l'adresse effective de l'opérande est contenue dans le un registre d'adresse après avoir été décrémenté de 1,2 ou 4. Si on reprend l'exemple précédant modifié:

CLR.L -(A0)

L'adresse est décrémentée de 4 { (A0)=$ABBC-4=$AAB8 } avant d'initialiser les octets d'adresses $AAB8, AAB9, AABA et AABB.

Remarque: pour ces deux derniers modes, si le registre d'indirection est le pointeur de pile et si la taille de l'opérande est l'octet, ce registre est décrémenté de 2 et non de 1, cela permet de ramener le taille au mot.

d) Adressage indirect avec déplacement: L'adresse effective est le contenu du registre d'adresse désigné, augmenté d'un déplacement signé sur 16 bits. Exemple:

CLR.W $600,(A0)

Cette instruction met à zéro le mot dont l'adresse effective est (A0)+$600.

e) Adressage registre indirect avec déplacement et indexe: L'adresse effective est la somme du contenu du registre d'adresse, de la valeur du déplacement et du contenu d'un registre utilisé comme index avec extension sur 32 bits si la taille du registre d'adresse est le mot ou le long mot. Exemple:

CLR.B $FF(A1,D1.B); $FF:déplacement,A1:registre d'indirection,D1:registre d'index.

L'adresse effective est (A1)+(D1.B)+$FF.

2.5.4 **Adressage absolu:**

L'adresse effective est donnée dans l'instruction et peut être sur 16 ou 32 bits, ce qui définit l'adressage absolu court et l'adressage absolu long. Exemple:

CLR.L $2000; adressage absolu court.

CLR.L $1 F000; adressage absolu long.

2.5.5 **Adressage immédiat:**

L'opérande est donné dans l'instruction. Exemple:

ADD.W #$1A,D1

Cette instruction additionne le contenu de D1 avec la valeur $1A et range le résultat dans le registre D1. L'opérande ne peut être seul dans l'instruction, il doit être accompagné soit d'un registre d'adresse, registre de donnée ou d'une adresse de position mémoire. Ce qui permet une classification de ce mode en sous modes suivant que l'opérande destination est un:

-Registre d'adresse.

-Registre de donnée

-Une position mémoire.

2.5.5- **Adressage relatif compteur programme (PC):**

Dans ce mode le processeur calcule l'adresse effective par rapport à l'adresse pointée par le conteur programme, il existe deux sous modes à savoir:

a) Compteur programme avec déplacement: L'adresse est égale à l'adresse courante du PC sommée à un déplacement signé sur 16 bits.

b) Compteur programme avec index: Dans ce sous mode on ajoute encore la valeur contenue dans le registre utilisé comme index, qui peut être un registre d'adresse ou de données sur 16 ou 32 bits.

2.5.6- **Adressage implicite:**

Certaines instructions font allusion à un registre sans pour cela qu'il soit spécifié dans l'instruction. On fournit ci-après une liste d'instructions qui désignent implicitement certains registres.

Bcc(PC), BRA(PC) , BSR(PC , USP), CHK(SSP, SR)

2.6 Jeu d'instruction du MC68000:

Le jeu d'instructions du MC68000 comporte 56 instructions, cela pourrait nous amener à penser que ce dernier est un microprocesseur moins performant que le MC6800 par exemple, dont le jeu d'instruction comporte 192 instructions. La réalité est tout autre, les instructions du MC68000 peuvent opérer sur des opérandes de différentes tailles pouvant être le bit, l'octet, le mot ou le long mot sans oublier l'existence de 14 modes d'adressage qui font en réalité qu'il possède plus de 1000 instructions.

Les instructions du MC68000 se répartissent en 7 groupes qui sont:

*Manipulation des données.

*Manipulations au niveau de bit.

*Opérations arithmétiques.

*Opérations logiques.

*Instructions de rotation et de décalage.

*Saut systématiques ou conditionnels.

*Commandes et contrôle de l'unité centrale.

*Instructions privilégiées.

2.6.1- Instructions de manipulation des données:

C'est l'ensemble des instructions qui permettent l'échange et le transfert de données. Cet ensemble d'instructions est résumé dans le tableau de la figure 2.17.

Instruction	Taille de l'opérande	Explication
MOVE	8,16,32	Transfert
MOVE→CCR	16	Chargement du registre de conditions
MOVE→SR	16	Chargement du registre d'état
MOVE←SR	16	Transfert du contenu du reg d'état en mémoire
MOVE USP	32	Transfert du pointeur de pile dans registre d'adresse et vis-versa
MOVEA	16,32	Chargement du registre d'adresse
MOVEM	16,32	Transfert des contenus de plusieurs registres dans la mémoire
MOVEM	16,32	Transfert des contenus de positions mémoire consécutives dans les registres
MOVEP	16,32	Tranfert d'une données périphérique
MOVEQ	32	Chargement d'une quantité de 8 bits dans un registre de données Dn
LEA	32	Chargement de l'adresse effective dans un registre d'adresse
PEA	32	Chargement de l'adresse effective dans la pile
EXG	32	Echange des contenus de deux registres
SWAP	16 16	Echange des poids forts et des poids faibles
LINK		Appel d'un nouveau contexte
UNLK		Retour d'un contexte

Fig 2.17 Instructions de transfert

On remarque en analysant ce tableau que l'instruction MOVE a dix versions, qui servent toutes au transfert de données entre les registres du microprocesseur et la mémoire. Les instructions les plus remarquables sont:

MOVEQ: Effectue un transfert rapide d'une donnée de 8 bits dans un registre de donnés.

MOVEM: Transfert de plusieurs registres dans des emplacement consécutifs dans la mémoire ou inversement, cette instruction augmente considérablement le temps d'exécution en cas d'appel de sous programmes.

MOVEP: Cette instruction est particulièrement utile lors du transfert de données entre la mémoire et les coupleurs de la famille MC6800.

Ce qui est utile de remarquer, c'est l'existance des instructions LINK et UNLK qui permettent de maintenir dans la pille une liste de données et de zones de paramètres servant au appels de sous programmes emboîtés.

2.6.2- **Instructions au niveau du bit:**

C'est un ensemble d'instructions permettant de tester ou de positionner à 0 ou à 1 un bit d'un opérande de 8 ou 32 bit en spécifiant son rang.

Instruction	Taille de l'opérande	Explication
BTST	8,32	Sélection d'un bit dans un mot et transfert dans Z
BCHG	8,32	Même opération que BTST avec complément à un du bit concerné
BSET	8,32	Même opération que BTST avec mise à un du bit concerné
BCLR	8,32	Même opération que BTST avec mise a 0 du bit concerné

Fig 2.18 Instructions au niveau du bit

2.6.3- **Opération arithmétiques:**

A part les instructions courantes, c'est à dire celles qu'on rencontre dans le jeu d'instructions des microprocesseurs 8 bits, on remarque l'existance de variantes pour les opérations. Telles que l'addition avec retenue, sans retenue, immédiate, rapide, sur des quantités binaires ou en BCD (binaire codé décimal). L'absence d'opérations d'incrémentation et de décrémentation est compensé par les opérations d'addition et de soustraction rapides, l'existance d'opération de multiplications et de divisions rend le processeur très puissant.

Instruction	Taille de l'opérande	Explication
ADD	8,16,32	Addition sans retenue
ADDA	16,32	Addition sans retenue
ADDX	8,16,32	Addition avec retenue
SUB	8,16,32	Soustraction sans retenue
SUBA	16,32	Soustraction sans retenue
SUBX	8,16,32	Soustraction avec retenue
MULU	16x16:32	Multiplication non signée
MULS	16x16:32	Multiplication signée
DIV U	32/16	Division non signée
DIV S	32/16	Division signée
ADDI	8,16,32	Addition immédiate
SUBI	8,16,32	Soustraction immédiate
ADDQ	8,16,32	Addition immédiate rapide
SUBQ	8,16,32	Soustraction immédiate rapide
ABCD	8	Addition décimale avec retenue
SBCD	8	Soustraction décimale avec retenue
EXT	8:16,16:32	Extension du bit de signe du registre de données
CLR	8,16,32	Remise à zéro
NEG	8,16,32	Complément à 2 en binaire
NEGX	8,16,32	Complément à 2 avec retenue en binaire
NBCD	8	Complément à 10 en BCD
TST	8,16,32	Test avec zéro
CMPM	8,16,32	Comparaison de 2 positions mémoire
CMP	8,16,32	Comparaison de 2 opérandes
CMPA	16,32	Comparaison de 2 opérandes
CMPI	8,16,32	Comparaison immédiate

Fig 2.19 Instructions arithmétiques

2.6.4- Instructions logiques:

mis à part les instructions classiques, il y a une instruction particulière TAS (Test avaible source) qui test et positionne à un le bits du poids fort d'un octet, le tout s'effectuant en un seul cycle indivisible. Cette instruction a été implanté par MOTOROLA pour facilité l'utilisation du MC68000 en milieu multiprocesseurs.

Instruction	Taille de l'opérande	Explication
AND	8,16,32	ET logique
EOR	8,16,32	OU exclusif
OR	8,16,32	OU inclusif
NOT	8,16,32	Complément à 1
ANDI	8,16,32	ET logique immédiat
EORI	8,16,32	OU exclusif immédiat
ORI	8,16,32	OU inclusif immédiat
TAS	8	Teste de disponibilité d'une ressourse

Fig 2.20 Instructions logiques

2.6.5- Instructions de rotations et de décalages:

Comme leurs noms l'indiquent, ces instructions effectuent des décalage ou des rotations dans

les registres de données ou en mémoire.

Particularités:

-On peut effectuer des décalages et rotations sur plusieurs bits si l'opérande est un registre de données.

-Les instructions de décalages et rotations dans la mémoire ne peuvent se faire que sur des opérandes mots(16 bits).

Instruction	Taille de l'opérande	Explication
ASL	8,16,32	Décalage arithmétique à gauche
ASR	8,16,32	Décalage arithmétique à droite
LSL	8,16,32	Décalage logique à gauche
LSR	8,16,32	Décalage arithmétique à droite
ROL	8,16,32	Rotation à gauche
ROL	8,16,32	Rotation à droite
ROXL	8,16,32	Rotation à gauche avec X
ROXR	8,16,32	Rotation à droite avec X

Fig 2.21 Instructions de rotation et décalages

2.6.6- Instructions de sauts:

Ces instructions comprennent les instructions de saut systématiques tel que BRA et JMP, de saut conditionnel tel que Bcc qui teste une condition, décrémente et fait un saut en relatif sur 16 bits. Cette instruction peut être considérée comme une instruction d'un langage évolué.

Instruction	Explication
BRA	Saut sytématique en relatif avec déplacement sur 8 ou 16 bits, c'est-à-dire +128 0u +32768
JMP	Saut systématique en absolu
Bcc	Saut conditionnel en relatif avec déplacement signé sur 8 16 bits
BSR	Appel d'un sous programme en relatif avec déplacement signé sur 8 ou 16 bits
JSR	Appel d'un sous programme en absolu
RTS	Retour au programme principal
RTR	Retour au programme principal et restitution du registre d'état
DBcc	Test d'une condition,décrémentation et saut en relatif sur 16 bits

Fig 2.22 Instructions de sauts

Remarque: L'indice cc désigne le code condition qui est explicité sur la figure 2.23

CC 0100-retenue à zéro	LS 0011-bas ou identique
CS 0101-retenue à 1	LT 1101-inférieur
EQ 0111-égal	MI 1011-moins
F 0001-jamais vrai	NE 0110-différent
GE 1100-supérieur ou égal	PL 1010-plus
GT 1110-supérieur	T 0000-toujours vrai
HI 0010-haut	VC 1000-pas de débordement
LE 1111-inférieur ou égal	VS 1001-débordement

Fig 2.23 Tableau des codes condition

2.6.7- **Instructions de commande et contrôle de l'unité centrale:**

C'est un ensemble d'instructions qui permettent de commander l'unité centrale (interruptions systématiques TRAP ou conditionnelle (TRAPV, CHK), les circuits externes (RESET), ou le contrôle des programmes (RTR, RTS, RTE). Voir Figure 3.24.

Instruction	Explication
TRAP	Instruction d'interruption systématique
TRAPV	Instruction d'interruption sur dépassement
CHK	Instruction d'interruption si (Dn)<0 ou(Dn)>limite
STOP	Arrêt du processeur jusqu'à la prochaine interruption ou remise à l'état initial
RESET	Remise à Zéro des circuits externes
NOP	Sans opération
Scc	Positionnement conditionnel à 1 de tous les bits d'un octet
RTS	Retour au programme principal après traitement d'un sous-programme
RTE	retour d'une trappe
RTR	Retour avec restauration des codes conditions

Fig 2.24 Instructions de contrôle

2.6.7- **Instructions privilégiées:**

Comme on l'a vu précédemment le MC68000 peut travailler suivant deux modes: mode superviseur et utilisateur. Il existe des instructions qui ne peuvent s'exécuter qu'en mode superviseur, cela à pour effet d'accroître la sécurité du système et facilité la tâche de l'utilisateur. l'exécution d'une instruction privilégiée en mode utilisateur entraîne systématiquement une exception. Il est nécessaire de remarquer que toute tentative de passage du mode utilisateur au mode superviseur ne peut se faire qu'à l'aide des instructions de contrôle de l'unité centrale (TRAP,TRAPV.CHK etc..).

Instruction	Explication
RESET	Remise à l'état initial des circuits périphériques
RTE	Retour d'exception
STOP	Arrêt de l'exécution d'un programme
ORI SUR SR	<<OU IMMEDIAT>> sur le registre d'état
MOVE USP	Transfert du pointeur de pile dans registre et vice versa
ANDI SUR SR	<<ET IMMEDIAT>> sur le registre d'état
EORI SUR SR	<<OU EXCLUSIF>> sur le registre d'état
MOVE EA SR	Chargement du registre d'état

Fig 2.25 Instructions privilégiées

CHAPITRE 3

CONCEPTION ET REALISATION DU KIT FC68

La plus part des systèmes logiques gérés par un microprocesseur comprennent principalement de la mémoire et des interfaces d'entrées/sorties. La figure 3.1 représente le schéma synoptique de tels systèmes:

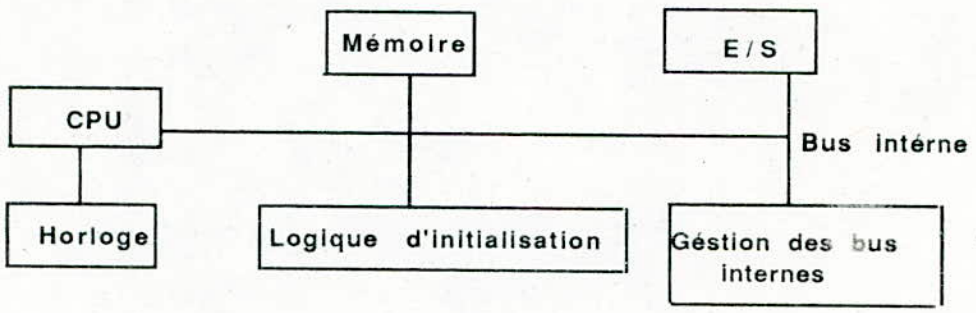


Figure 3.1 Schéma bloc d'un système à microprocesseur

Le microprocesseur MC68000 de MOTOROLA qui constitue le coeur du kit pédagogique FC68 possède une structure externe riche et un jeu de signaux complet, tel qu'il est possible de répartir en différents modules indépendants, toutes les fonctions nécessaires au développement de la carte. Un tel procédé permet de faciliter considérablement la conception et la mise au point du système (tests et détection d'erreurs).

La figure 3.2 représente le schéma bloc du système minimal FC68.

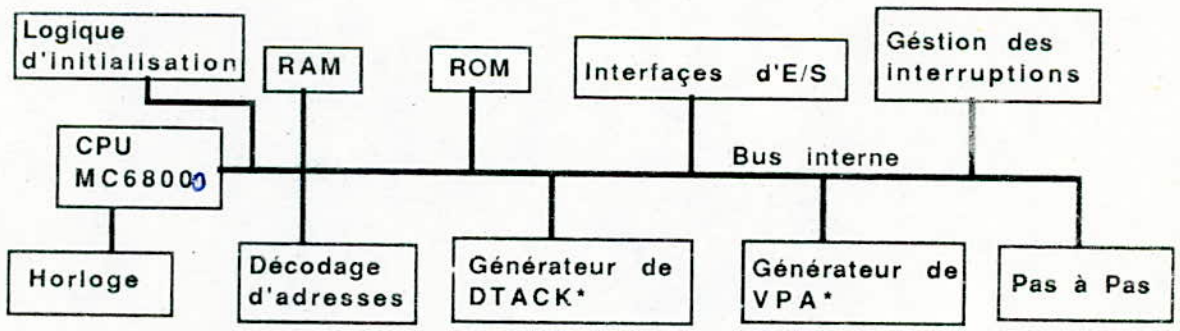


Figure 3.2 Schéma synoptique du système minimal

Le fonctionnement de chaque module va être détaillé dans les paragraphes suivants :

3.1- Le circuit d'horloge:

C'est à partir du signal d'horloge injecté sur l'entrée clock du MC68000 que celui-ci <<produit>> les différents signaux nécessaire à son fonctionnement (exemple : l'horloge E).

Le signal compatible TTL doit être parfaitement stable et suivre les caractéristiques du

constructeur. Le circuit d'horloge du MC68000 a été conçu d'une manière simple grâce à quelques résistances , inverseurs (74LS04) et un cristal 8MHz comme le montre la figure 3.3.

Les inverseurs sont pourvus de réactions positves locale et peuvent être assimilés à des amplificateurs opérationnels dans la partie linéaire de leurs caractéristiques.

Le quartz 8MHz , contrôle la réinjection du signal donc permet de fournir et de stabiliser la fréquence du signal, un diviseur de fréquence (74LS93) pourvoie le système d'un jeu de signaux d'horloge de différentes fréquences (8, 4, 2, 1, 0.5 MHz) pouvant servir à la mise au point et les tests des circuits.

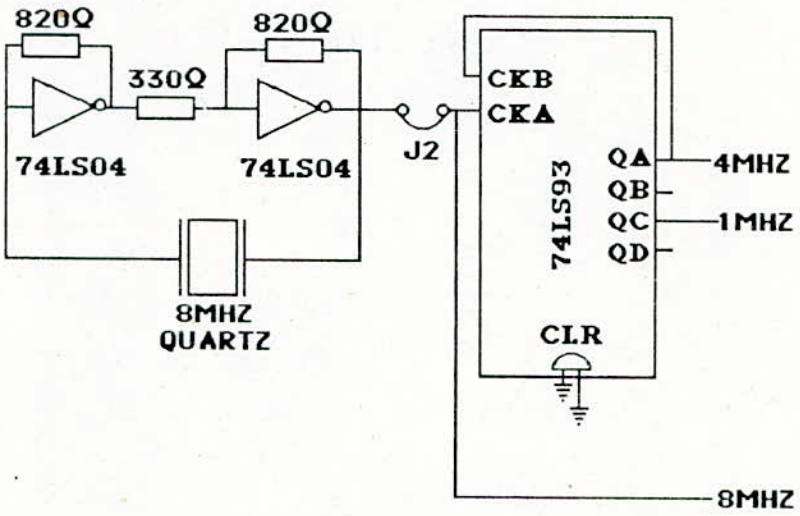


Fig 3.3 Circuit d'horloge

Les fréquences d'horloges utilisées sont respectivement de 4 et 8 MHz. Notons que lors de la communication du 68000 avec des périphériques relativement lents, l'utilisation du signal d'horloge de 4 MHz s'avère d'une égale rapidité que l'utilisation d'une fréquence double avec l'insertion de 4 ou plusieurs états d'attente.

3.2- Logique d'initialisation:

La séquence d'initialisation du MC68000 se résume à un niveau bas d'au moins 100 ms sur les entrées HALT* et RESET*. Pour ce faire et fournir la durée nécessaire, on utilise un circuit R-C générant un signal exponentiel qui est mis en forme par deux inverseurs de types trigger de SCHMITT (voir figure 3.4). En effet les portes à trigger de SCHMITT procurent un niveau de basculement élevé (1.7V) et donc une bonne immunité au bruit permettant ainsi de nous dispenser de l'utilisation d'une bascule anti-rebonds (RS).

Un bouton poussoir est prévu pour permette au manipulateur de générer une séquence d'initialisation manuelle, l'appui sur ce bouton décharge la capacité et un autre cycle de charge

donc d'initialisation est entamée.

La LED permet d'indiquer l'état HALT du processeur, quant une erreur système grave survient (double erreur bus) , le processeur cesse tout traitement et affirme le signal HALT* en sortie. Le signal DEMR* est prévu pour une utilisation dans le module de décodage d'adresses.

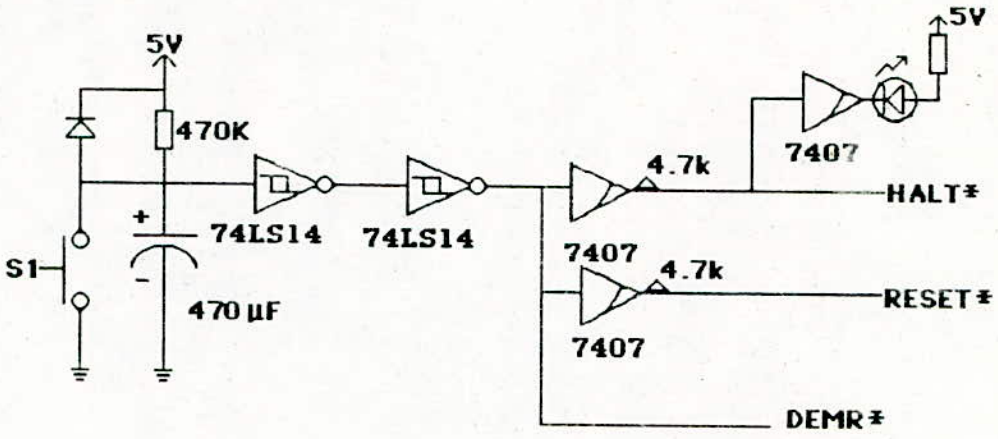


Figure 3.4 Logique d'initialisation

3.3- Décodage d'adresses:

La carte mémoire du système minimal FC68 est représentée sur la figure 3.5.

champ Mémoire		Boîtiers	Adresse	signal de validation
RAM système	Tables d'interruption	EPROM	000000-000007H	CSROM*(1)
		RAM	000000-0003FFH	CSRAM*
		RAM	000400-0008FFH	CSRAM*
RAM utilisateur		RAM	000900-007FFFH	CSRAM*
Moniteur EPROM		EPROM	008000-00BFFFH	CSROM*
utilisateur		EPROM	00C000-00FFFFH	CSROM*(2)
Non utilisée			010000-01003FH	(3)
ACIA1	terminal	MC6850	010040 et 010042H	I/O SELECT*
ACIA2	HOST	MC6850	010041 et 010043H	I/O SELECT*
Non utilisée			010044-07FFFFH	(3)
Non utilisée			020000-07FFFFH	(4)
Non utilisée			080000-0FFFFFFH	(5)

(1) Seulement durant la séquence de démarrage.
(2) Non implanté, extension.
(3) prévu pour extension, périphériques synchrones.
(4) prévu pour extension.
(5) utilisable si une logique est ajoutée.

Figure 3.5 Carte mémoire du KIT FC68

L'adresse de la mémoire RAM se trouve au bas de la carte mémoire, sauf durant la séquence de démarrage où le moniteur est décodé à l'adresse 000000H.

Une partie de cette carte mémoire découle de l'utilisation du moniteur "TUTOR FIRMWARE" qui impose la localisation de la mémoire RAM système et le moniteur. A partir de cette table est déduit le logigramme représenté sur la figure 3.6.

Un décodeur (74LS138) est utilisé pour segmenter la zone adressable en 8 pages mémoire de taille 32 KOctets chacune. Ce décodeur est validé par le signal NORA ainsi que la ligne

d'adresse A18.

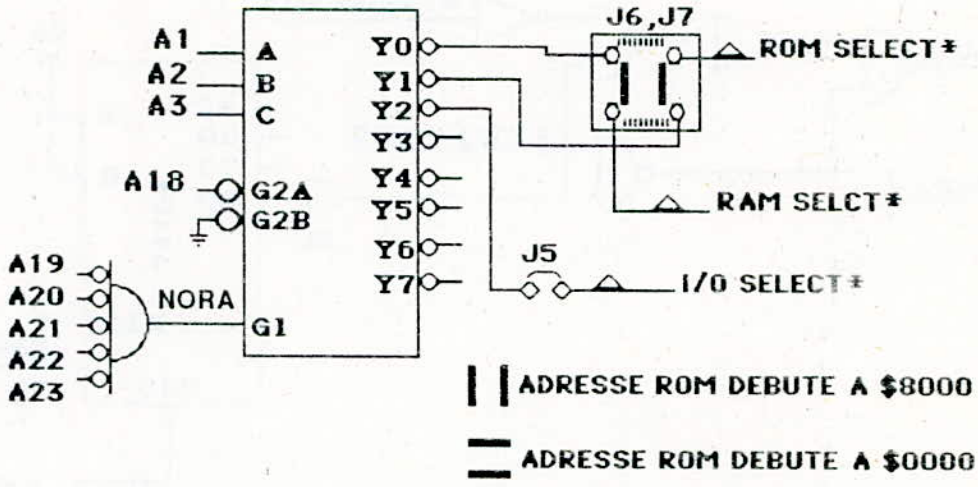


Fig 3.6 Décodage d'adresses

Pour permettre une plus grande souplesse d'utilisation du KIT, les cavaliers J6 et J7 ont été prévus pour permuter les adresses des zones mémoires RAM et ROM comme indiquer sur la figure 3.6.

Au RESET, le 68000 lit aux adresses 0 jusqu'à 7H les valeurs du pointeur de pile système et du compteur programme. Ces deux valeurs se trouvant dans le moniteur, il est nécessaire de pouvoir le décoder à l'adresse 0H pendant les 4 premiers cycles accès mémoire. Le circuit représenté sur la figure 3.7 permet de réaliser cette fonction appelée fonction démarrage. Un compteur (74164) indique que 4 cycles accès mémoire ont été effectués à partir du moment où la séquence d'initialisation est terminée. Cette information se trouvant sur la sortie Q_D du compteur (l'horloge pilotant ce dernier est le signal AS*) est mise en équation avec les signaux RAM SELECT* et ROM SELECT* pour forcer CSRAM* à l'état HAUT, CSROM* à l'état BAS et ce durant la séquence de démarrage (les 4 premiers cycles accès mémoire) donc d'accéder à la zone mémoire EPROM.

Le signal DEMR* permet d'inhiber le compteur (74164) et ce durant la phase d'initialisation.

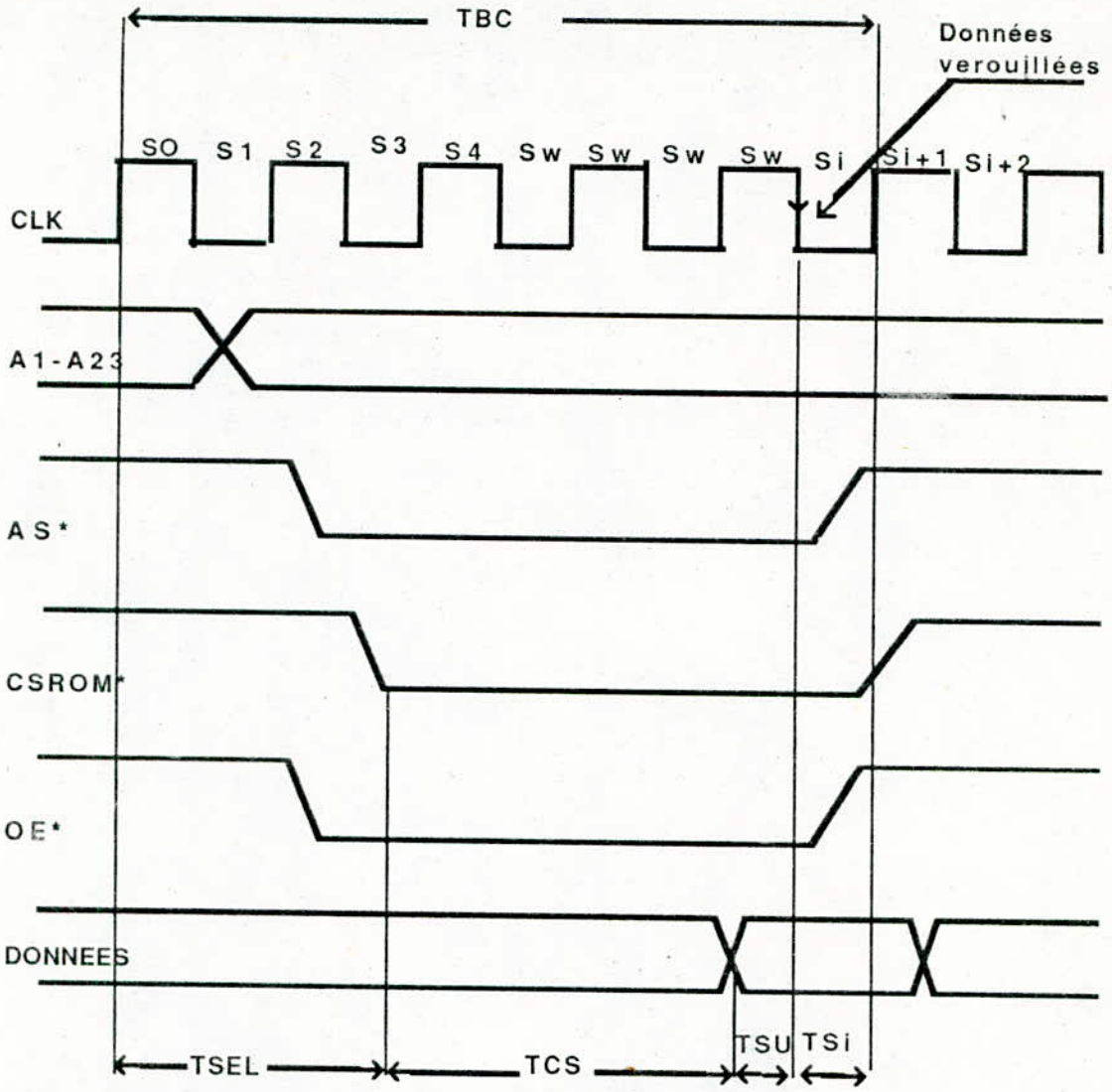


Fig 3.8 Chronogramme de dialogue avec l'EPROM

légendes:

TBC: Temps du cycle bus complet.

TCS: Temps qui s'écoule entre le moment où on valide l'EPROM et le moment où les données se sont stabilisées sur le bus.

TSU: (set up) le temps qui s'écoule entre le moment où les données sont valides et le front descendant de l'horloge.

TSEL: Temps qui s'écoule entre le début du cycle et le passage de CSROM* à l'état bas.

TSi: durée d'une demi-période de l'horloge système (CLK68).

Si: Un état du cycle correspondant à une demi-période.

T68000 : Période de l'horloge système (CLK68).

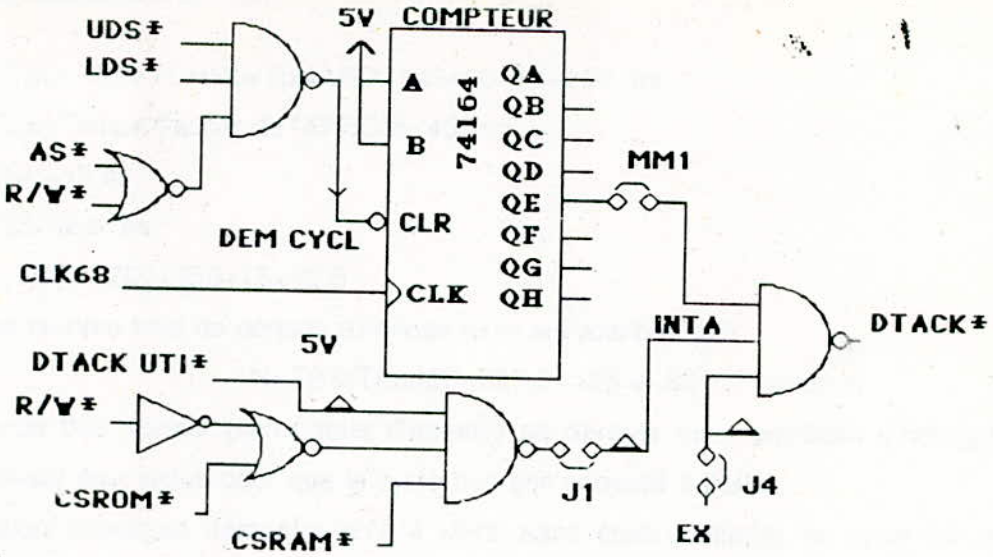


Fig 3.9 Circuit générateur du signal DTACK*

Le signal DTACK* est affirmé par une logique externe une fois le temps d'accès du circuit adresser est écoulé. Si le signal DTACK* n'arrive pas avant S4 (au set up près), le processeur substitue des états d'attente aux états S5 et S6. Ces états sont introduits par paires.

Le circuit représenté sur la figure 3.9 permet de réaliser une telle fonction. Un compteur, validé par le signal DEM CYCL (Démarrage de cycle) permet de compter les TOPS de l'horloge système (CLK68). Suivant la sortie choisie, ce dernier génère une temporisation de 1,2,...,7 fois la période de l'horloge système. Un jeu de micro-switch permet le choix de la temporisation appropriée. (Voir Annexe C)

La génération d'un seul signal DTACK* commun à tout les dispositifs asynchrones réduit considérablement le coût du système.

L'une des sorties du compteur est mise en équation avec les signaux EX et INTA, afin de permettre:

* Interdiction de l'accès à l'EPROM en mode écriture.
* Interdiction de l'accès à la zone mémoire non utilisable.
* Validation du mode pas à pas.
* Forcer le signal DTACK* à 1, lors d'un cycle d'échange synchrone, le signal VPA* prenant la relève dans ce cas.

Les dispositifs utilisés en mode de dialogue asynchrone sont les RAM statiques µPD 2167 (temps d'accès 45 ns) et les EPROM 2764 (temps d'accès 450 ns). Utilisant une fréquence d'horloge de 8 MHz, il est impératif d'introduire des états d'attente lors de la communication avec l'EPROM qui un dispositif relativement lent. Le calcul du nombre d'états d'attente à introduire dans le cycle se fait ainsi:

Le temps du cycle bus minimum est

TBC > TSEL+TCS+TSU+TSi avec

TSEL=Temps entre le début du cycle bus et CSROM* validé

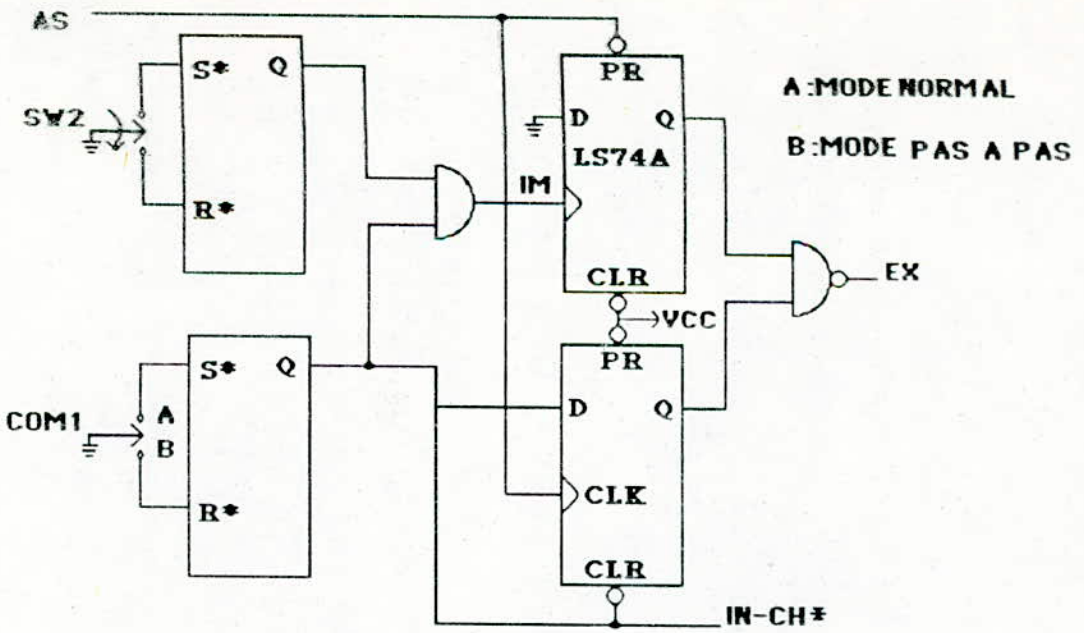


Fig 3.10 Module pas à pas

*Commutateur COM1 en position B: EX=1(mode pas à pas inhibé)

COM1 en position A: EX=0, l'appui sur SW2 permet d'envoyer une impulsion positive qui sert d'horloge à la bascule D, un niveau 0 est transmis à sa sortie entraînant le passage du signal EX à l'état haut (affirmant DTACK) . Ce dernier est remis automatiquement à l'état initial à la fin du cycle bus, lorsque AS* est infirmé.

Remarque: Il est possible de contrôler le mode pas à pas à l'aide du signal HALT*, seulement le passage dans ce cas des bus adresses et données à l'état haute impédance fait perdre son utilité au mode pas à pas. L'utilisation du signal DTACK* est toutefois déconseillée dans le cas où des mémoires RAM dynamiques sont implantées dans le système (nécessité de rafraîchissement).

3.6- Le "chien de garde":

Le circuit représenté sur la figure 3.11 valide l'entrée BERR* du 68000 pour lui indiqué qu'une erreur dans le cycle bus est survenue. Cette erreur arrive dans le cas ou l'on adresse une zone mémoire interdite (écriture dans l'EPROM) ou non implantée.

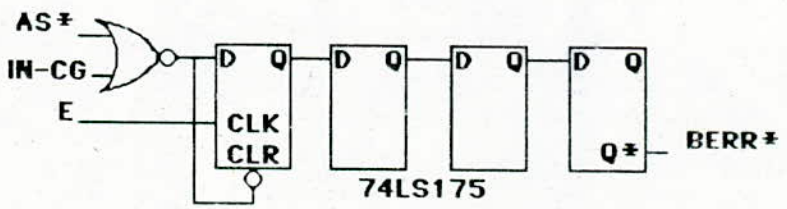


Fig 3.11 Le "chien de garde"

Si le cycle bus ne se termine pas au bout de 4 périodes de l'horloge E générée par le 68000 le "chien de garde" valide BERR*. Ce signal est inhibé par le signal IN-CG quant on passe au

mode pas à pas matériel.

3.7- Module mémoire:

La conception des modules mémoire RAM et EPROM ne pose pas de problèmes particuliers. Les boîtiers mémoires s'interfacent au microprocesseur à l'aide des lignes d'adresses, la ligne lecture/écriture et les signaux de validations (CSRAM* et CSROM*). La RAM est validée par le signal CSRAM*, l'EPROM par CSROM* (voir schéma numéro 5 annexe B).

Le protocole de communication est pris en main par une autre logique à savoir le module DTACK*.

Le système minimal dispose de 32 KOctets de mémoire RAM statique disposée en 16Kx1bit (16 boîtiers de 16 Kx1bit, pour obtenir un bus de données de 16 bits). Les boîtiers de RAM utilisés sont les µPD 2167 de NEC, ils sont compatibles avec les 51C67 d'INTEL. Leur utilisation est imposée par le cahier des charges.

Avantages: Rapidité; temps d'accès de 45ns , ne nécessitent pas de rafraîchissement.
inconvénients: Consommation élevée de puissance, encombrement de la carte UC.

16 KOctets de mémoire morte (reprogrammable) EPROM contiennent le moniteur qui gère le kit et sont organisés en 8KOctets x 2. Les boîtiers utilisés sont des 2764.

Le simple remplacement de ces derniers par des boîtiers 27128, sans modifier le hardware, permet d'avoir une capacité double de mémoire EPROM (16KOctets x2 soit 32 KOctets).

3.8- Circuit d'interruption:

Sur les 7 niveaux disponibles, 3 seulement sont utilisés et sont de type autovectorisé, les 4 autres sont prévus pour une éventuelle extension, comme indiquer dans le tableau ci-après.

NIVEAU	Interruption
7	NMI *
6	ACIA2 *
5	ACIA1 *
4	Non utilisé **
3	Non utilisé **
2	Non utilisé **
1	Non utilisé **

Fig 3.12 Table des niveau d'interruption

* Interruption autovectorisée.

* Non implanté, prévu pour extension.

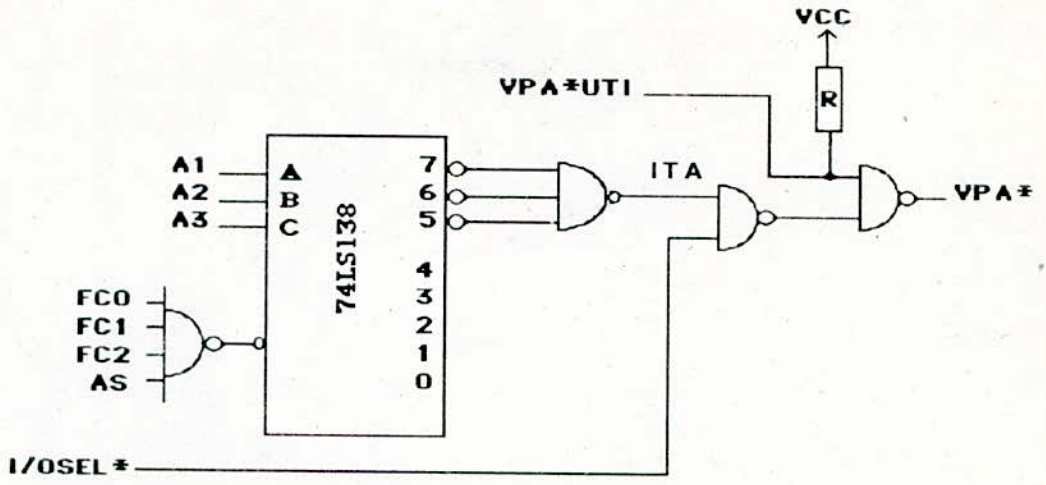


Fig 3.14 Génération du signal VPA*

De même que le signal DTACK UTI*, le signal VPA* UTI (utilisateur) est mis à la disposition de l'utilisateur pour permettre l'extension du système.

3.10- Interfaces d'entrées/sorties:

Le kit FC68 dispose de deux interfaces d'entrées/sorties sériels compatibles RS232C.

Les deux ports ont été configuré en DCE (data communication equipment: transmission sur la broche 3 et réception sur la 2 du connecteur V24).

Deux ACIA MC6850 sont utilisés à cet effet. Le MC6850 est un circuit d'interface entre le microprocesseur et un périphérique travaillant en mode série synchrone. Il réalise la mise au format des données et le contrôle de le transmission. Ce dernier est relier au système par des entrées de sélection, d'horloge, la ligne lecture/écriture, une ligne d'interruption et un bus de données 8 bits bidirectionnel.

comme indiquer sur la figure 3.15, la ligne A1 du CPU étant reliée à l'entrée RS (register select: sélection de registre) du MC6850, permetant, en conjonction avec la ligne R/W* d'accéder à l'un des 4 registres internes de l'ACIA. Ce dernier est sélectionné par I/O select*, VMA* et A6. Les lignes UDS* et LDS* permettent l'aiguillage vers l'un des deux ACIA. Sur les 25 signaux spécifiés par la norme RS232C, seuls 4 signaux sont utilisés pour la communication avec le terminal (port1):TxDATA, RxDATA, DTR, GND et 3 pour le port2 (host) TxDATA, RxDATA, GND.

TxDATA : Ligne de transmission des données.

RxDATA : Ligne de réception des données.

DTR : Terminal prêt à recevoir ou à émettre.

GND : La masse.

Le passage des niveaux TTL (0, 5 V) aux niveaux RS232C (+12 et -12 V) est réalisé par le circuit intégré MAX 237 CNG, qui a pour avantage d'être alimenter sous 5 V, il en découle une configuration monotension pour la carte FC68 (5 V).

Vue que le moniteur ne supporte pas la fonction impression de données, un artifice

matériel a été implanter au sein de la carte et permet de relier les deux ports simultanément comme indiquer sur la figure 3.16.

L'imprimante connectée sur le port 2 serra vue par le kit comme étant une console.

Une série de microswitchs est prévue pour la sélection de la cadence de transmission (BAUD RATE). Les fréquences utilisables sont de 9600, 4800, 2400, 1200, 600, 300, 150, 110. (Voir annexe C).

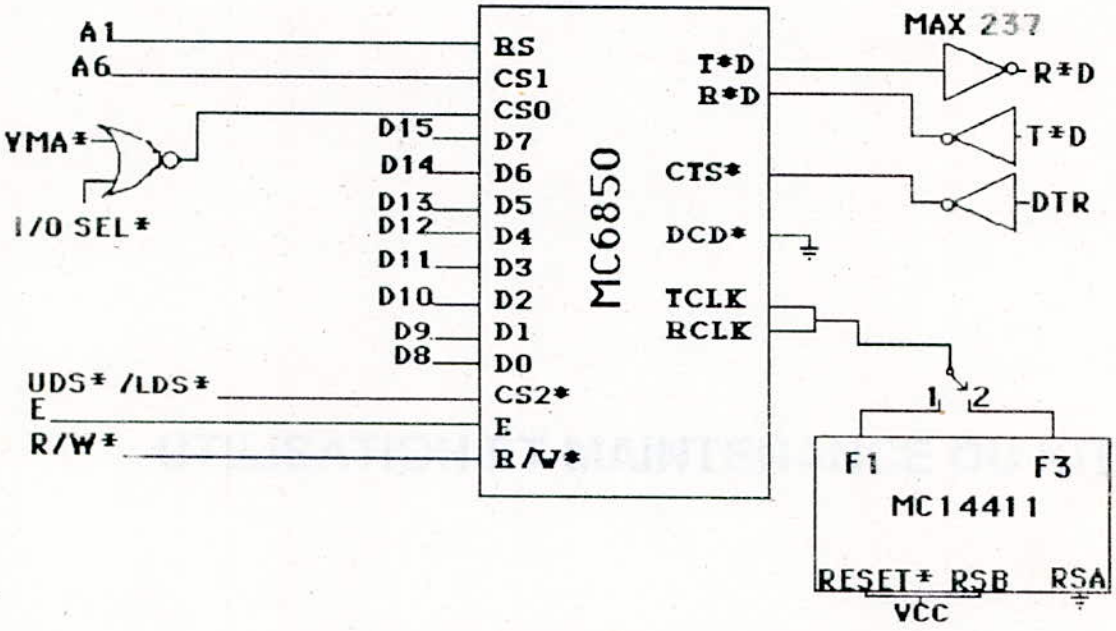


Fig 3.15 Interfaces d'entrées/sorties

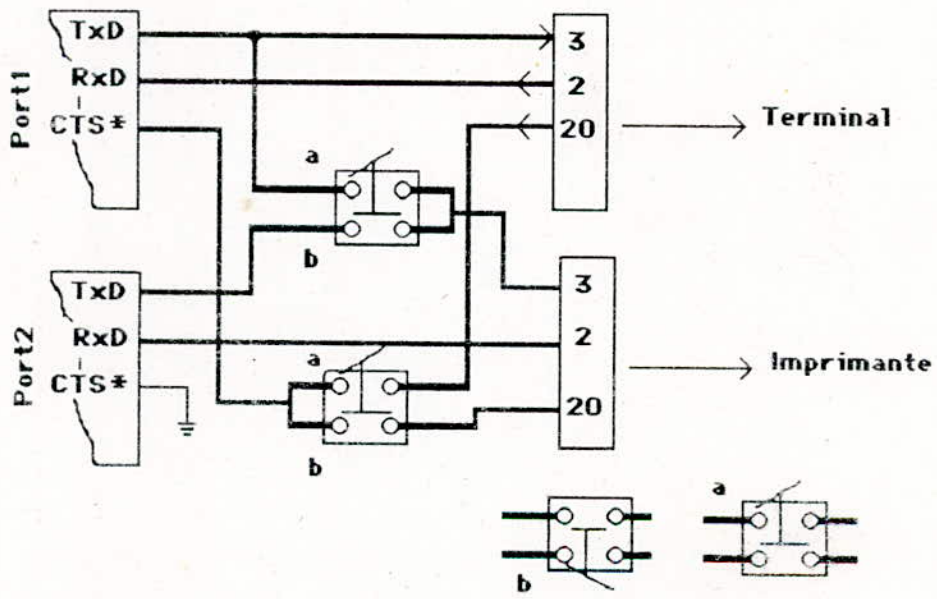


Fig 3.16 connexion des deux ports sériels

4.1 Utilisation du kit FC68:

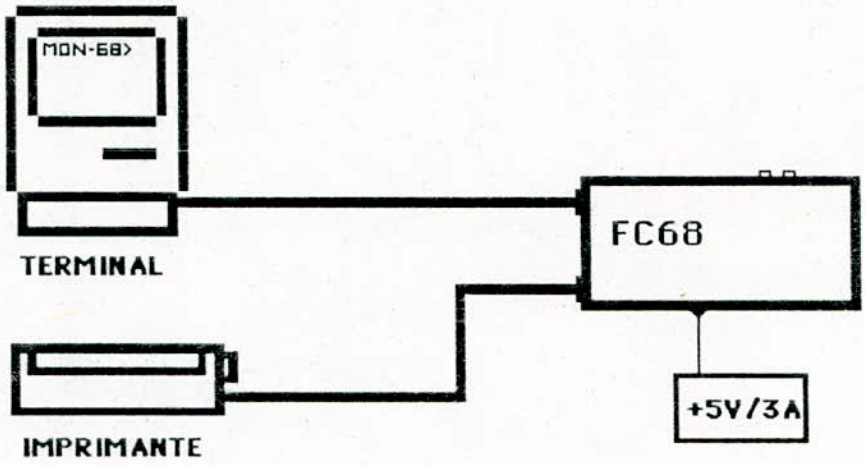


Fig 4.1 installation du système minimal

La mise en marche du kit FC68 se fait comme suit:

+ Avant d'alimenter le système
  - Vérifier que les cavaliers J1, J2, J3, J5 et J8 sont en place. (Annexe C)
  - Vérifier que les cavaliers J6 et J7 sont placés de façon à ce que l'EPROM soit décodée à l'adresse 8000H. (Annexe C)
  - Se positionner en mode normal. (Annexe C)
  - Connecter le port1 à un terminal grâce à un câble RS232C.
  - Choisir à l'aide des microswitchs MM2 la vitesse de transmission suivant le terminal utilisé. (Annexe C)
  - La carte étant déconnectée, régler le niveau de tension de l'alimentation utilisée à 5V +/- 5%. (3A Max)
+ Alimenter la carte
  - Mettre sous tension le terminal.
  - Le bon fonctionnement de la carte est manifesté par l'affichage du prompt **MON-68 1.0>** sur l'écran du terminal.

+ Pour l'impression de données
  - Couper l'alimentation.
  - Mettre hors service l'imprimante.
  - Sélectionner la vitesse de transmission requise à l'aide des microswitchs MM3 suivant l'imprimante utilisée. (Annexe C)
  - Alimenter l'imprimante puis la carte.
  - Mettre les microswitchs MM3 en position impression.
  - Taper une commande terminée par un retour chariot.

4.2 Autotest:

Le kit pédagogique est muni de certains artifices permettant:

- dépannage rapide en cas de panne.
- Indiquer l'état HALT du processeur.
- Fonctionnement en mode pas à pas matériel.
- Permutation des adresses de la RAM et l'EPROM et inhibition de la fonction démarrage.
- Génération d'une interruption non masquable pour arrêter l'exécution d'un programme.
- Contrôle d'une imprimante sérielle émulation "console".

4.2.1- Auto-test du kit:

La méthode utilisée lors des différentes phases de mise au point du kit est la suivante:

- Câblage d'un module.
- Test individuel du module.
- Intégration du module au sein du système, vérification de son bon fonctionnement à l'aide de l'auto-test.
- Passage à un autre module.

Le bon fonctionnement d'un module n'indique pas nécessairement son bon fonctionnement une fois intégré dans le système. Comme indiquer ci-dessus, pour s'assurer de la bonne démarche de la réalisation, il est quasi impératif d'utiliser le MC68000 en fonctionnement dit en mode "libre".

Le fonctionnement libre veut dire que le microprocesseur est capable d'exécuter l'instruction " **ne rien faire**" (**NRF**) continuellement. Ceci est accompli par la rupture de la liaison qui existe entre le 68000 et tous ces périphérique (RAM, EPROM etc..). Au lieu de charger les programmes de la mémoire, une instruction **NRF** , codée sur 16 bits est mise sur le bus de données. Le 68000 lit l'instruction **NRF** , l'exécute, incrémante le PC et lit la prochaine instruction **NRF**. Ce cycle se répète le long des 16 Mega-Octets adressables; quant le MPU atteint la fin des 16 MOctets, il reprend simplement à zéro pour refaire la même chose.

L'instruction **NRF** n'est autre que l'instruction **ORI #0, D0** ayant pour code 0000H. Ceci est fait en mettant tout le bus de données (D0-D15) à zéro c'est à dire à la masse comme indiqué sur la figure 4.1.

Seuls le module d'horloge et le circuit d'initialisation sont nécessaires pour réaliser le fonctionnement libre du CPU. Tous les modules peuvent être testés en utilisant ce mode à part le bus de données lui même ou plus précisément le transfert de données entre le CPU et un périphérique quelconque.

Le temps de balayage de tout le champs mémoire peut être facilement calculé. Chaque

On conclue par là que l'auto-test est un outil commode dans le dépannage de la carte et la compréhension du son fonctionnement (du point de vue HARDWARE).

Sous l'auto-test le passage au mode pas à pas matériel permet de contrôler l'allumage de la LED 2 à l'aide du bouton poussoir SW2.

4.2.2- **Maintenance du KIT FC68 :**

Le dépannage du système minimal a été facilité grâce à la constitution très modulaire de ce dernier. Parmi les méthodes qui peuvent être utilisées on peut citer:

- Auto-test : Fonctionnernent en mode libre du microprocesseur.
- Utilisation d'un émulateur.

Premiers test à faire:

Avant d'utiliser les grands moyens tels que l'émulateur faire les vérifications suivantes:

1- Est-ce-que le commutateur COM1 a été basculer accidentellement validant ainsi le mode pas à pas.(Annexe C)

2- Est-ce-que la tension est de 5 V +/- 5%.

3- Vérifier que les paramètres de la transmission avec le terminal (utiliser le SET UP du terminal) sont:

- Transmission de 8 bits.
- 1 bit de stop.
- Pas de bits de parité.
- Même Vitesse de transmission (BAUD RATE).
- Le terminal devrait procurer le signal DTR (DATA TERMINAL READY).

4- Voir si la LED1 (indiquant le mode HALT) est allumée? Si non appuyer sur le bouton RESET (SW1) : La LED devrait être allumée tant que le bouton est activé. Si la LED reste toujours allumée, désactiver le chien de garde (enlever le cavalier J4), est ce qu'elle s'éteint?

5- Observer la LED3. Elle doit être allumée partiellement, elle ne peut être éteinte ou complètement allumée que si le 68000 est à l'arrêt quelque part.

6- Se positionner en mode pas à pas, initialiser la carte, exécuter des pas (en appuyant sur SW2) . La LED3 devrait s'éteindre et s'allumer alternativement. Prendre du bus de données, les valeurs lues par le 68000 durant la séquence d'initialisation. Respectivement, on devrait trouver 0000H, 0444H (valeur du pointeur de pile superviseur) , 0000H et 8146H (PC).

Si on n'arrive pas à un diagnostic en suivant les phases citées précédemment, utiliser l'auto-test :

1- Enlever les deux boîtiers d'EPROM, placer à leur place des **Sockets** (court-circuit

du bus de données).

2- Enlever les cavaliers J1, J4, J5, J6 et J7.

3- Alimenter le système.

4- Observer la LED 2, elle devrait clignoter, si non vérifier le circuit d'horloge et le circuit d'initialisation à l'aide d'un oscilloscope.

5- Vérifier à l'aide d'une sonde logique si les signaux suivants basculent entre l'état haut et bas. se référer au schémas (annexe B).

6- Si le signal DTACK* est tout le temps à l'état haut, vérifier le module DTACK*. Est-ce-que le signal EX est à 1.

7- Utilisant un oscilloscope double trace, relier une de ses voies au signal AS* (broche 6 du 68000).

a- Le déclenchement se fait sur AS*.

b- Avec l'autre voie, relever les chronogrammes relatifs au signaux DTACK*, UDS*, LDS*. Sont-ils conforment au chronogrammes trouvés dans l'annexe E.

c- Augmenter le nombre d'états d'attente (MM1, annexe C). La LED 2 devrait clignoter plus lentement à chaque augmentation du nombre d'états d'attente.

Si le diagnostic n'est pas encore établit, utiliser un émulateur .

1- Tester les différents bus, vérifier qu'il n'y a pas de court circuit.

2- Test de la RAM : Tester les possibilités de lecture et d'écriture , Chaque bit à chaque adresse. Voir les bits court-circuités, Les erreurs de décodage d'adresses.

3- Test de la ROM.

CHAPITRE 5

DESCRIPTION ET UTILISATION DU MONITEUR/DEBUGGER

MON-68K

5.1 Présentation du *MON-68K*

Le kit FC68 est doté d'un programme résident de 16 k octets (contenu dans une paire d'EPROM 8K x 8 de type 2764), appelé MONITEUR 68000 abrégé: *"MON-68K"*.

Ce moniteur qui permet la communication entre l'utilisateur et le 68000 constitue le " *CHEF D'ORCHESTRE* " de tout le système, il offre en plus des fonctions classiques d'un moniteur/debugger, un moyen de programmation très souple grâce à son ASSEMBLEUR/DESASSEMBLEUR ainsi qu'un outil de développement puissant, le tout dans un environnement de travail autonome.

Le MON-68K est une version modifiée et adaptée du TUTOR FIRMWARE de MOTOROLA , lequel mis à la disposition des utilisateurs du microprocesseur MC68000 permet la mise en oeuvre rapide et le développement de systèmes autour de ce microprocesseur.

Les modifications apportées sur les terminologies des commandes et des messages de la version 1.3 du TUTOR, permettra une meilleure présentation du kit ainsi qu'une commodité d'usage.

Le kit FC68 ainsi muni de ce moniteur, procure à l'utilisateur un moyen de mise au point et de développement non négligeables à la fois "software" et "hardware". Utilisant ces atouts la carte CPU peut prendre en charge son éventuelle extension.

La mise au point du programme se fait simplement, grâce à l'ASSEMBLEUR-EDITEUR ligne par ligne, puis à l'aide du moniteur qui permet le lancement du programme et le contrôle de son déroulement (éventuellement instruction par instruction: c'est le mode 'TRACE'),et cela jusqu'a ce que la conformité avec les résultats désirés soit satisfaite. A ce moment, si cet ensemble n'est destiné qu'à couvrir cette seule application, le programme mis au point peut être enregistré sur EPROM qui viendront remplacer celles qui contiennent le *MON-68K*, devenu désormais inutile.

La communication avec le kit se fait à travers un jeu de commandes prédéfinies introduites via le clavier d'un terminal, ces commandes sont classées en 4 catégories :

a. Commandes de Visualisation ou Modification de mémoire.
b. Commandes de Visualisation ou Modification des registres du 68000.
c. Commandes permettant à l'utilisateur d'exécuter un programme sous différents niveaux de contrôle.
d. Commandes de contrôle d'accès aux ports d'Entrées/Sorties du système.

Le *MON-68K* possède en plus de ces commandes une fonction de prise en main appelée TRAP 14 permettant l'accès à différentes routines sous contrôle du moniteur, simplifiant ainsi

considérablement la programmation .

L'organigramme du *MON-68K* est représenté par la figure 5.1.

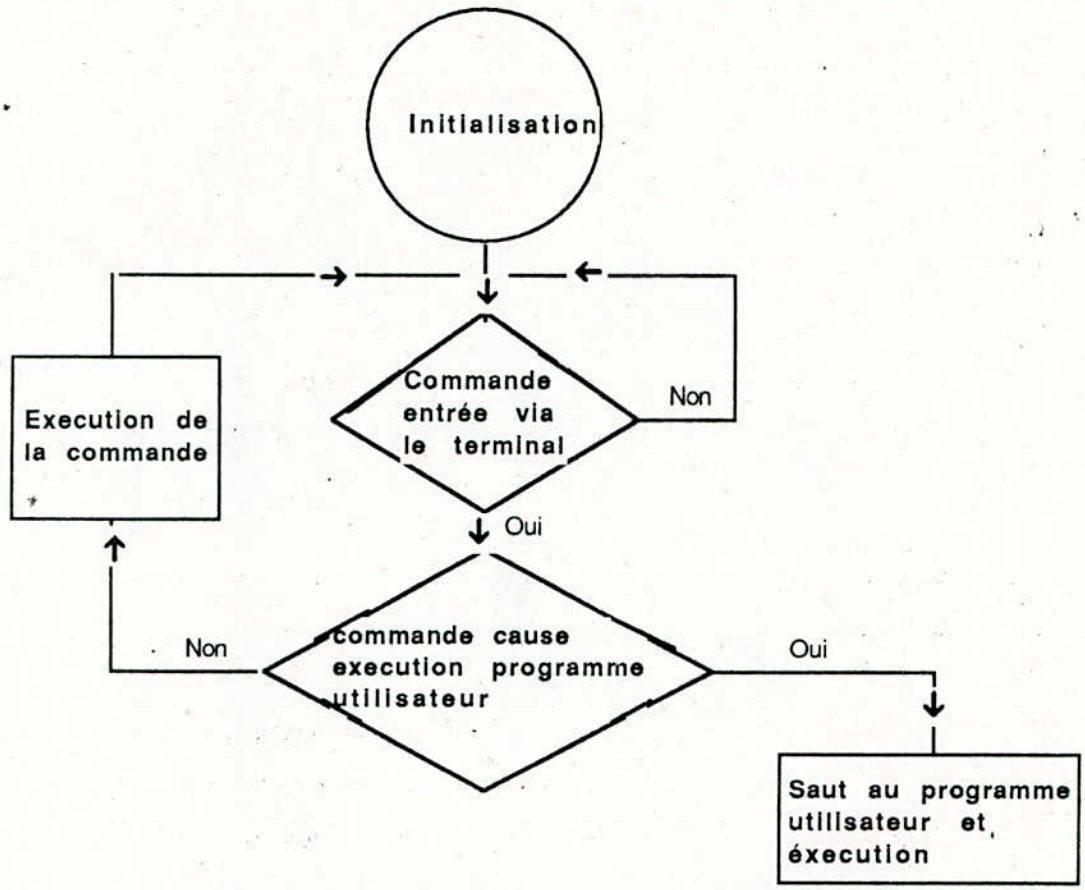


FIG 5.1 Organigramme de fonctionnement du *MON-68K*

Après initialisation, le système se met en mode attente, jusqu'a ce qu'une commande appropriée soit introduite à travers le terminal. Le traitement de la commande peut se faire dans l'un des 2 modes principaux d'exécution:

En mode utilisateur:

Si la commande fait appel à un programme utilisateur, le *MON-68K* peut ou non redonner la main à l'utilisateur (en fonction du contenu du programme et de la "discrétion " du programmeur).

En mode superviseur:

Dans le cas contraire (c.a.d si la commande fait appel à un programme superviseur), la commande est exécutée sous contrôle du moniteur qui repasse automatiquement après en mode attente.

5.2 Utilisation du *MON-68K*

5.2.1- Contrôle de l'Editeur:

Diverses touches clavier sont utilisées pour l'édition des lignes de commandes et leur contrôle.

a. Touche d'effacement (Delete) ou CTRL H : Efface le caractère à gauche du curseur .

b. CTRL X : Annule toute la ligne de commande .

c. CTRL D: Réaffiche la ligne de commande précédente .

d. Retour (Retour chariot) : Fait entrer la commande pour exécution .

e. CTRL W : Arrêt de défilement .l'appui sur n'importe qu'elle touche permet de reprendre le défilement.

f. Break : Inhibe l'exécution de la commande en cours .

5.2.2 **Format d'une commande:**

Le format d'une commande est la suivante:

MON-68K 1.0> [IN] < COMMANDE > [< PARAMETRES >] [; < OPTIONS >]

où :

MON-68K > : Est le prompt généré par le moniteur.

IN : La négation de la commande primitive.

COMMANDE : La commande primitive.

PARAMETRE : Séparés par des espaces et peuvent être sous forme d'une < EXPRESSION > ou < ADRESSE >

OPTIONS : Diverses options peuvent être choisies .

Une commande élémentaire comprend au moins un champ de commande primitive et un champ de paramètres, néanmoins, certaines commandes ne nécessitent pas de paramètres ex : VR (Visualisation de Registre du 68000).

Si une option est rajoutée à la commande, elle doit nécessairement être précédée d'un point virgule (;), différentes possibilités d'extensions sont offertes .

Les paramètres :

a.< Expression > : peut être une ou plusieurs valeurs numériques séparées par un opérateur arithmétique (+), (-). Tout nombres apparaissant dans une commande sont considérés comme héxadécimaux sauf ceux précédés du symbole (&) qui spécifie des nombres décimaux. En assembleur tout nombre non précédé du symbole ($) est pris par défaut comme étant décimal.

b. < Adresse >: Plusieurs commandes utilisent une < Adresse > comme paramètre, ces commandes ont la même syntaxe que celle utilisée en assembleur à une différence prés : Le mode indirect mémoire, où l'on utilise 8 registres d'offset (ou de décalage) R0-R7 ,

notons que ces registres sont purement software ou virtuels , ils permettent toutefois d'élargir les possibilités d'adressage déjà existantes.

Format d'adresse :

Format	Exemple	Description
Expression	140	Adresse absolue (pas d'offset).
Expression+Offset	130+R5 *	Adresse absolue plus un offset.
(A@)	(A5)	Adressage registre indirect .
(A@,D@)	(A6,D4)	Adressage registre indirect avec index
Expression(A@)	120(A3)	Adressage Registre indirect avec déplacement.
Expression(A@,D@)	110(A2,D1)	Adressage registre indirect avec
Expression(A@,A@)		déplacement et index .
[Expression]	[100]*	Adressage mémoire indirect .

* syntaxe non-utilisée en Assembleur .

NOTE : Le format des lignes de commandes est définies en utilisant des symboles spéciaux qui ont la signification syntaxique suivante :

[ ] : Délimite un champ optionnel.

< > : Délimite une variable syntaxique.

Ces symboles ne doivent pas être introduits par l'utilisateur ils sont utilisées uniquement à titre explicatif.

5.2.3 - **Registres d'Offsets:**

8 registres virtuels sont utilisés pour modifier les adresses contenues dans les commandes du *MON-68K*; les 7 premiers registres (R0-R6) sont utilisés exclusivement pour le décalage d'adresse , le 8^{iem} registre R7 est toujours forcé à zéro, leurs contenus peut être visualisés en utilisant la commande '.RO', et '.Rx' pour modifier le contenu de chacun d'eux .

Ces 8 registres sont mis à zéro lors de la séquence d'initialisation du système (soit à la sous tension ou lorsque le bouton Reset est activé).

Toute commande comprenant un paramètre d'adresse utilise implicitement R0 comme Offset, sauf si un autre Offset est introduit; ainsi si R0=1000 les commandes suivantes ont le même effet :

```
PA 10
PA 10+R0
```

L'adresse physique pour chacune des commandes est 1010.

L'Offset R0 est également ajouté aux autres registres chaque fois que ceux-ci sont modifiés sauf si un autre registre est spécifié, c'est le cas du registre R7 (toujours nul) qui est souvent ajouté aux autres registres afin de les initialiser.

Exemple:

.R1 8	R1=9	Offset R0 est nul,l'Offset R1=9 .
.R0 100	R0=400	
.R2 200	R2=200+400=300	l'Offset R0 est ajouté .
.R0 0+R7	R0=0	R0 est remis à zéro .

5.3 Les commandes du *MON-68K* :

5.3.1- SOMMAIRE DES COMMANDES:

Le sommaire des commandes du moniteur est représenté dans la table de la Figure 5.2 .

Mnémonique de la commande	description de la commande
VM	Visualisation mémoire .
MM, M	Modification mémoire .
CM	Changement mémoire .
.A0-.A7	Visualisation/Modification Registres d'Adresse
.D0-.D7	Visualisation/Modification Registres Données
.PC	Visualisation/Modification PC
.SR	Visualisation/modification Registre d'état
.SS	Visualisation/Modification Pointeur de Pile Superviseur
.US	Visualisation/Modification Pointeur de Pile Utilisateur
VR	Visualisation des Registres du 68000
RO	Visualisation des Registres d'Offset
.R0-.R6	Visualisation/Modification des Registres d'Offset
CB	Chargement d'un Bloc de Mémoire
DB	Déplacement d'un Bloc de Mémoire
TB	Test d'un Bloc de Mémoire
RB	Recherche d'un Bloc de Mémoire
PA	Points d'Arrêt
INPA	Initialisation des Points d'Arrêt
EX, E	Exécuter un programme
ET	Exécuter jusqu'aux Points d'Arrêt
ED	Exécution directe
TR, T	Trace
TT	Trace Temporaire
CD	Conversion de données
IC *	Imprimante Connectée
INIC *	Initialisation de l'Imprimante Connectée
FP	Format de Port
MT *	Mode transparent
*	Envoie de messages au port 2
AI	Aide
VB	Vidage d'un Bloc de Mémoire
CH	Chargement
VE	vérifications

* Commandes non utilisées avec la configuration du FC68.

Fig 5.2 Commandes du Moniteur

5.3.2- DESCRIPTION DES COMMANDES DU *MON-68K:*

Visualisation Mémoire : VM

VM [<Numéro du Port>] <Adresse> [<Nbre>] [;<Options>]

La commande VM est utilisée pour visualiser un bloc ou une section de mémoire commençant à <Adresse>], le <Nbre> indiquant le nombre d'octets à visualiser, le contenu de ce bloc de mémoire est affiché soit en héxadécimal avec traduction ASCII (lorsque cela est possible), soit en forme désassemblée en utilisant l'option DI.

Les principales formes de la commande sont :

Commande	Port	Destination
MD	Port 1	Terminal (Défaut)
MD1	Port 1	Terminal
MD2	Port	Imprimante,Host Modem....

Lorsque le <Nbre> n'est pas spécifié dans la commande VM, 16 octets sont affichés ou une seule instruction si l'option DI est utilisée; la commande peut toutefois être relancée par un RETOUR, 16 nouvelles lignes sont alors affichées etc...

Voir aussi : MM , CM.

Exemple:

```
MON-68K 1.0> VM 2000 14
002000      0C 00 00 30 6D 28 0C 00 00 39 6E 10 02 80 00 00   ...0m(...9n.....
002010      00 0F 11 C0 10 38 1E 3C  00 E4 4E 4E 0C  00 00  41  ...@.8.<.dNN....
MON-68K 1.0> VM 2000 12 ;DI
002000    0C000030              CMP.B  #48,D0
002004    6D28                  BLT.S  $00102E
002006     0C000039             CMP.B  #57,D0
00200A    6E10                  BGT.S  $00101C
00200C     0280000000F          AND.L  #15,D0
```

Modification Mémoire MM

MM <Adresse> [;<Option>]

La commande MM permet de visualiser et de modifier éventuellement un octet, un mot ou double mot.

Option	description
Aucune	1 octet est visualisé
; w	2 octets ou 1 mot est visualisé
; L	4 octets ou 1 long mot est visualisé
; O (Odd)	L'octet le plus significatif est visualisé
; V(Even)	L'octet le moins significatif est visualisé
; N	Sans visualisation de la donnée

Note :

Si différentes options sont nécessaire, un point virgule (;) doit précéder chacune d'elles .

Les modifications mémoire peuvent se faire de différentes manières :

[<Donnée>] (retour)	Modification de la donnée et Incrémentation d'adresse
[<Donnée>] ^(retour)	Modification de la donnée et Décrémentation d'adresse
[<Donnée>]=(retour)	Modification et visualisation de la même position mémoire
[<Donnée>].(retour)	Fin de la séquence

Voir aussi : VM, CM

Exemple:

```
MON-68 1.0> MM 3000;W
003000    2200  ?FFFF
003002    2345  ?AAAA
003004    4EFE  ?EEEE^
003002    AAAA  ?
003004    EEEE  ?
003006    3EEF  ?BBBB=
003006    BBBB  ?.
MON-68 1.0> MM 4000;W;N
004000          ?234
004002          ?68.
```

Forme désassemblée :

(;DI) Cette option fait appel à la fonction Assembleur/Désassembleur, les instructions sont ainsi visualisées, sous une forme désassemblée , la visualisation d'une instruction est suivi immédiatement d'un point d'interrogation (?) qui indique qu'une nouvelle ligne programme peut être introduite, assemblée sauvegardée et visualisée.

Note: Si une erreur est détectée dans la nouvelle instruction, la ligne programme est réaffichée avec un "X" suivi d'un point d'interrogation (?) disposés sous l'emplacement de l'erreur dans ligne programme.

Exemple :

```
MON-68K 1.0>MM 4000;DI
004000    5555    SUBQ.W  #2,(A5)  ?(Espace) MOVE.L  A0,A1
004000    2248    MOVE.L  A0,A1
004002    1211    MOVE.B  (A1),D1     ?  MOVE.L A0A1
004002            MOVE.L  A0A1
                                          X?
                                           erreur !!
```

CHANGEMENT DE LA MEMOIRE CM

CM <Adresse> <Donnée..>

La commande CM change le contenu d'une position mémoire en y introduisant une nouvelle donnée sous forme ASCII ou héxa (chaîne de 8 caractères au maximum).

Format de la commande	Description
MON-68 1.0> CM 6000 'ABC'	Une chaîne de caractère est chargé est chargée à partir de l'adresse 6000
MON-68 1.0> CM 4000 7882	Une donnée héxadécimale est chargée.

Exemple:

```
MON-68 1.0> VM 2000
002000    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
MON-68 1.0> CM 2000 'ABC'
MON-68 1.0> VM 2000
002000    41 42 43 00 00 00 00 00 00 00 00 00 00 00 00 00  ABC............
```

VISUALISATION/MODIFICATION DES DIFFERENTS REGISTRE .Rx

. <Registre> <Expression>

. La commande .Rx permet à l'utilisateur de visualiser ou de modifier les différents registres :

.A0-A7	Registres d'Adresses
.D0-D7	Registres de Données
.R0-R6	Registres d'Offset
.PC	Compteur Programme
.SR	Registre d'Etat
.SS	Pointeur de pile Superviseur
.US	Pointeur de pile Utilisateur

Exemple :

```
MON-68K 1.0> .PC
.PC=00000000
MON-68K 1.0> .D2 FFF
MON-68K 1.0> .R5  2000
MON-68K 1.0> VR
PC=00001000  SR=2704= .S7 . . Z . .  US=FFFFFFFF  SS=00001300
D0=00000000   D1=00000000  D2=00000FFF   D3=00000000
D4=000E0DDF   D5=00000000  D6=00000000   D7=00000089
A0=00000000   A1=00000000  A2=00000000   A3=00000000
A4=00000000   A5=00000000  A6=00000000   A7=00000000
----------------------001000            FFFF          DC.W  $FFFF
```

Voir aussi : VR , VO

Visualisation des Registres du 68000 VR

VR

La visualisation du contenu des différents Registres du 68000 est obtenue en utilisant la commande VR.

Voir aussi .Rx (.Ax , .Dx, etc...).

Exemple:

```
PC=00001000  SR=2704= .S7 . . Z . .  US=FFFFFFFF  SS=00001300
D0=00000000   D1=00000000   D2=00000FFF   D3=00000000
D4=000E0DDF   D5=00000000   D6=00000000   D7=00000089
A0=00000000   A1=00000000   A2=00000000   A3=00000000
A4=00000000   A5=00000000   A6=00000000   A7=00000000
------------------001000              0C000030 CMP.B  #48,DO
```

Note :La commande VR permet de visualiser en forme désassemblée, l'instruction localisée à l'adresse pointée par le PC (Ex $001000) pour la mise au point des programmes .

Registres d'Offset R O

RO

La commande RO permet la visualisation du contenu de tout les registres d'Offset (ou décalage) R0-R7,ces registres sont très utiles dans l'adressage mémoire "relatif" .

Voir aussi : .Rx

Exemple :

```
MON-68 1.0> .R1 2000                    Placer un Offset dans R1
MON-68 1.0> .R3 4000                    Placer un Offset dans R3
MON-68 1.0> .R7 0+R7                    Remise à zéro de R5
MON-68 1.0> RO
R1=00000000   R1=00002000   R0=00000000   R3=00004000
R4=00000000   R5=00000000   R6=00000000   R7=00000000
MON-68 1.0> PA D08 1056
PTS D'ARRET
000D08          000D08
000056+R1       001056
MON-68 1.0>.R0 2000                     Un offset est chargé dans R0
MON-68 1.0> PA 10
PTS D'ARRET
000D08          000D08
000056+R1       001056
000010+R0       002010
MON-68 1.0> MM 1000+R7
```

000000+R1 0C ?. Pour accéder à l'adresse 1000 sans"décalage", R7 est rajouté.

Chargement d'un Bloc de Mémoire CB

CB <Adresse1> <Adresse2> <Mot>

CB permet essentiellement de charger un bloc de mémoire pour effacer des informations parasites .

La zone mémoire comprise entre <Adresse1> et <Adresse 2> va être chargée entièrement du Mot spécifié (On ne peut utiliser l'octet ou le mot long) . Avant d'introduire un autre programme , cette commande, qui détruit le "source" , constituera une judicieuse précaution.

Exemple :

```
MON-68K 1.0> VM 3004
003004  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00          ................
MON-68K 1.0> CB 3004 300A   4142
ADRESSE PHYSIQUE=00003004   0000300A
MON-68K 1.0> VM 3004
003004   41 42 41 42 41 42  41 42 00 00 00 00 00 00 00 00       ABABABAB........
```

Déplacement de bloc Mémoire DB

DB <Adresse1> <Adresse 2> <Adresse 3>

Déplacer une zone de mémoire par la commande DB est une opération fréquente par exemple pour :

-Reloger un programme ,c'est à dire l'implanter à un endroit où il sera mieux intégrer .

-Préserver une partie de la mémoire que l'on ne veut pas détruire .

La commande DB fait déplacer la zone mémoire comprise entre <Adresse1> et <Adresse 2> pour la mettre à partir de l'adresse <Adresse 3> . La zone initiale n'est pas affectée.

Exemple :

```
MON-68 1.0> DM 8000 C000 3000
ADRESSE PHYSIQUE= 00008000  0000C000        Transfert du  Le moniteur  en
ADRESSE PHYSIQUE= 00003000                  RAM.
```

MON-68K 1.0> RB 3000 3010 03 0F Recherche d'une donnée, avec

masquage du quartet le moins significatif

ADRESSE PHYSIQUE=00003000 00003010

003000 43

003001 43

Points d'arrêt PA

PA [<Adresse>] [;<Comptage>]

Lorsque la partie d'un programme à explorer n'est activée qu'après le déroulement de multiples instructions, il est commode de placer un point d'arrêt, dont le rôle est de stopper le déroulement du programme à l'adresse spécifiée dans la commande,l'instruction illégale $4AFFB est alors insérer à cet emplacement (l'adresse doit être paire).

A l'arrêt du programme, le moniteur reprend le contrôle après visualisation des différents registres. Il est possible de continuer l'exécution instruction par instruction en utilisant la commande TR (Trace).

Le <comptage> (Nbre sur 32 bits) permet de provoquer l'arrêt du programme et ce après plusieurs passages sur la même adresse avec décrémentation automatique du <Comptage>.

Plusieurs points d'arrêt (8) peuvent être déclarés dans une table , la visualisation de cette table est possible avec la commande PA, la commande INPA permet quand à elle d'INitialiser ou de supprimer tous ies points d'arrêt de la table.

Note:

- Lorsque le contrôle de l'exécution du programme n'est plus possible , agir sur le bouton "Arrêt" pour redonner le contrôle au MON-68K.

Voir aussi: ET, INPA, TT

Exemple:

MON-68K 1.0> .R4 4000

MON-68K> PA 1010 2000;5 2040 4000

PTS D'ARRET

001010 001010

002000 002000;5

002040 002040

000000+R4 004000

MON-68K 1.0 > INPA 1010 2040

PTS D'ARRET

```
00200          002000
00000+R4       004000
MON-68K 1.0> INPA
PTS D'ARRET
```

INitialisation des Points d'Arrêt INPA.

INPA [<Adresse> <Adresse> . . . .]

La commande INPA est utilisée pour initialiser ou supprimer un ou plusieurs points d'arrêt .

Voir aussi: PA, ET, TT

Exemple:

```
MON-68K> PA 1010   2000;5   2040   4000
PTS D'ARRET
001010         001010
002000         002000;5
002040         002040
000000+R4      004000

MON-68K 1.0 > INPA 1010   2040
PTS D'ARRET
00200          002000
00000+R4       004000
MON-68K 1.0> INPA
PTS D'ARRET
```

EXécution d'un programme EX

EX [<Adresse>]

E [<Adresse>]

La commande EX déclenche l'exécution du programme à partir de l'adresse spécifiée jusqu'à:

a- Rencontre d'un point d'arrêt

b- Rencontre d'une séquence anormal provoquant un processus (Ex: division par zéro)

c- Intervention extérieur à travers les bouton poussoir "RESET" et "ARRET".

Format de la commande :

MON-68K 1.0> EX L'exécution commence à l'adresse pointée par le PC.

MON-68K 1.0> EX Adresse L'adresse est chargée dans le PC avant l'exécution

Voir aussi: PA, VR, ED, ET, TR, TT

Exemple:

MON-68k 1.0> MM 2000;L

002000 00002F00 ? 3000.

MON-68k 1.0> GO [2000] Adressage mémoire indirect.

ADRESSE PHYSIQUE=00003000

EXécution jusqu'au point d'arrêt ET

ET <ADRESSE DU POINT D'ARRET>

La commande ET accomplie les fonctions suivantes:

1- Place un point d'arrêt temporaire

2-Place les points d'arrêt contenus dans la table.

3-Charge le PC avec l'adresse de début du programme.

4-Exécution du programme à partir de l'adresse contenu dans le PC.

Lorsque un point d'arrêt est rencontré , il y a initialisation automatique du point d'arrêt temporaire.

Dans la commande ET si l'adresse du point d'arrêt existe déjà dans la table (des points d'arrêt), un message d'erreur avec le contenu de la table sont affichés .

Voir aussi: PA, VR, ED, EX, TR, TT

Exemple:

MON-68K 1.0> PA 2010 3000

PTS D'ARRET

002010 002010

003000 003000

MON-68K 1.0> .PC 2000

MON-68K 1.0> ET 2006

ADRESSE PHYSIQUE=00002006

ADRESSE PHYSIQUE=00002000

MON-68K 1.0> ET 2010

ADRESSE PHYSIQUE=2010

ERR. .

002010 002010

003000 003000

Exécution Directe ED

ED [<Adresse>]

ED accomplie la même séquence que la commande EX , sans tenir compte des points d'arrêt ni modification des vecteurs d'exceptions (localisés entre $0 et $3FF).

Si l'<Adresse> n'est pas mentionner , l'exécution du programme commence à partir de l'adresse pointée par le PC.

Voir aussi: EX, ET

Exemple:

MON-68K 1.0> ED 4000

ADRESSE PHYSIQUE=00004000

Tracer un programme TR

TR [<Comptage>]

T [<Comptage>]

La commande TR permet à l'utilisateur de "TRACER" un programme ou l'exécution instruction par instruction à partir de l'adresse contenu dans le PC.

Dans le mode TRACE le prompt du moniteur est légèrement modifiée (MON-68K 1.0:>) , l'appui sur la touche RETOUR permet de "Tracer" le programme en tenant compte des ponts d'arrêt et du comptage.

Pour quitter ce mode , n'importe qu'elle commande peut être introduite .

Voir aussi : VR, EX, ET, ED

Exemple:

MON-68K 1.0> .R6 2000

MON-68K 1.0> .PC 0+R6

MON-68K 1.0> TR 3

```
ADRESSE PHYSIQUE=00002000
PC=00002002  SR=2700=.S7. . . . . US=FFFFFFFF   SS=000007BC
D0=0030FFFF   D1=0030FFFF   D2=00FFFFFF   D3=00000000
D4=00FFFFFC   D5=00E0FFFF   D6=00OOOO02   D7=00000000
A0=00010040   A1=00020040   A2=00000878   A3=00000004
A4=000FFF56   A5=00000040   A6=00000086   A7=000007BC
------------------000002+R6       45F82056                    LEA.L    $00002056,A2
PC=00002006  SR=2700=.S7. . . . . US=FFFFFFFF   SS=000007BC
D0=0030FFFF   D1=0030FFFF   D2=00FFFFFF   D3=00000000
D4=00FFFFFC   D5=00E0FFFF   D6=00OOOO02   D7=00000000
A0=00010040   A1=00020040   A2=00000878   A3=00000004
A4=000FFF56   A5=00000040   A6=00000086   A7=000007BC
------------------000006+R6       4EF90008152                 JMP      $00008152
.PC sous "DEBUGGER".
PC=00002002  SR=2700=.S7. . . . . US=FFFFFFFF   SS=000007BC
D0=0030FFFF   D1=0030FFFF   D2=00FFFFFF   D3=00000000
D4=00FFFFFC   D5=00E0FFFF   D6=00OOOO02   D7=00000000
A0=00010040   A1=00020040   A2=00000878   A3=00000004
A4=000FFF56   A5=00000040   A6=00000086   A7=000007BC
------------------006152+R6       48B800008152                MOVE.W   D0,$0406
MON-68K 1.0:>
```

Trace jusqu'au point d'arrêt — TT

TT <Adresse du point d'arrêt>

La commande TT accomplie la séquence suivante :

a- Place un PA temporaire à l'adresse spécifiée

b-Commence l'exécution du programme dans le mode "TRACE"

c-"TRACE" le programme jusqu'a rencontre d'un PA

d- Initialise le PA temporaire.

Voir aussi: VR, EX , ET, TR

Exemple:

```
MON-68K 1.0> .PC 2000
MON-68K 1.0> TT  2004
ADRESSE PHYSIQUE=00002004
```

Adresse du PA temporaire - $2004

```
ADRESSE PHYSIQUE=00002000                    Adresse d'Exécution - $2000
PC=00002002  SR=2700=.S7. . . . . US=FFFFFFFF   SS=000007BC
D0=0030FFFF   D1=0030FFFF   D2=00FFFFFF   D3=00000000
D4=00FFFFFC   D5=00E0FFFF   D6=00OOOO02   D7=00000000
A0=00010040    A1=00020040   A2=00000878   A3=00000004
A4=000FFF56    A5=00000040   A6=00000086   A7=000007BC
------------------002002        48B800008152              MOVE.W      D0,$0406
AU PTS D'ARRET
PC=00002004  SR=2700=.S7. . . . . US=FFFFFFFF   SS=000007BC
D0=0030FFFF   D1=0030FFFF   D2=00FFFFFF   D3=00000000
D4=00FFFFFC   D5=00E0FFFF   D6=00OOOO02   D7=00000000
A0=00010040    A1=00020040   A2=00000878   A3=00000004
A4=000FFF56    A5=00000040   A6=00000086   A7=000007BC
------------------002004            60FA                        BRA.S     $002000
```

Conversion de données CD

CD <Expression>

La commande CD est utilisée pour convertir une donnée en Héxadécimale ou en Décimale,Le résultat sera affiché dans les 2 modes .

Cette commande est très utile dans le calcul des déplacements d'adresses utilisés soit avec les instructions de branchement relatif ou comme compteur programme en mode adressage relatif.

Un Offset peut être utilisé avec la commande DC(l'Offset R0 est rajouté par défaut).

Format de la commande:	description
MON-68K 1.O> CD $Donnée	Conversion Héxa/Décimal
MON-68K 1.O> CD &Donnée	Conversion Décimal/Héxa

Exemple:

```
MON-68 1.0> CD &120
$78=&120
MON-68 1.0> CD &15+$4-&13
$0=&0
MON-68 1.0> CD -1000
$FFFFF000=$-1000=-&4096
MON-68 1.0> .R0  1000
MON-68 1.0> CD 10+10+30
```

```
$1050=&4176
MON-68 1.0> CD 10+10+30+R7
$50=&80
```

Aide AI

AI

La commande AI permet de visualiser la liste des différentes commandes du MON-68K .

Exemple:

```
MON-68K 1.0> AI
.PC .SR .US .SS
.D0  .D1  .D2    .D3   .D4  .D5   .D6  .D7
.A0  .A1  .A2    .A3   .A4  .A5   .A6  .A7
.R0  .R1         .R2   .R3  .R4   .R5  .R6  .R7
CB   DB  PA      INPA  RB   TB    CD   VR
VB   E   ED      EX    ET   AI    CH   M
VM   MM          CM    RO   IC    INIC FP   T
MT   TR  TT      VE
```

5.4 La fonction TRAP #14:

5.4.1 *Qu'est ce que la fonction TRAP#14?*

Le MON-68K contient une fonction supplémentaire, appelée TRAP #14, qui peut être introduite dans un programme, pour prendre en main diverses routines sous contrôle du moniteur. Ce chapitre décrit les différentes routines de la fonction TRAP #14 et leurs utilisation.

Les "TRAPs" (pièges) sont des instructions qui génèrent des procédures d'exception ou d'interruption, ces vecteurs sont au nombre de 16 et sont très utiles pour l'implantation de routines ou la définition de macrocommandes pouvant être appelées par le programme principal.

Pour répondre à ce besoin, la fonction TRAP #14 est utilisée pour simplifier considérablement la programmation, 255 routines sont accessibles à l'utilisateur par programmation. Le numéro de la routine ou la fonction désirée est chargé dans l'octet le moins significatif du registre de données D7 suivant la séquence:

MOVE.B #< numéro de la fonction >, D7

TRAP # 1 4

parmi les 255 routines existantes 127 d'entre elles (Numéro 128 à 254) sont réservées, l'utilisateur peut bénéficier des fonctions restantes (0 à 127) pour implanter ces sous programmes.

Les routines TRAP #14:

Les différentes routines que supporte la fonction TRAP #14, peuvent être classées en 5 groupes à savoir:

a- Transfert de caractères ou chaînes de caractères entre les différents ports.

b- Routines de conversion:

Hexadécimale/Décimale (Format ASCII).

Hexadécimale/ASCII 1,2,4,6,8 caractères.

ASCII/Hexadécimale 1 seul caractère.

ASCII (BCD)/Hexadécimal.

ASCII (Décimal)/Hexadécimal.

c- Routines de contrôle des buffers.

d- Transfert du contrôle au moniteur avec ou non une séquence d'initialisation.

e- Insertion de nouvelles routines utilisateur à l'intérieur de la table de référence TRAP #14.

Pour l'utilisation des différentes fonctions TRAP se référer à (11).

Les différentes fonctions TARPs sont résumées dans le tableau ci-après.

Fonction	Nom de la Fonction	Description de la fonction
255	_	Fonction reservée- Indicateur de fin de table
254	_	Fonction reservée- Utilisée pour lier des tables
253	LINKIT	Ajoute la table utilisateur à la table TRAP#14
252	FTXDADD	Met une chaîne de caractères dans le buffer
251	FIXBUF	Initialise A5 et A6 à la valeur du pointeur de la table TRAP#14:BUFFER
250	FIXDATA	Initalise A6 à 'BUFFER' et Met une chaîne de caractères dans le buffer
249	FIXDCRLF	Transfert les caractères 'CR' et 'LF' au buffer
248	OUTCH	Transfert d'un caractère au port1
247	INCHE	Lit un caractère du port1
246,245	---	Fonction réservée
244	CHRPRINT	Transfert d'un caractère au port3
243	OUTPUT	Transfert d'une chaîne de caractères au port1
242	OUTPUT21	Transfert d'une chaîne de caractères au port2
241	PORTIN1	Lit une chaîne de caractères du port1
240	PORTIN20	Lit une chaîne de caractères du port2
239	TAPEOUT	Transfert d'une chaîne de caractères au port4
238	TAPEIN	Lit une chaîne de caractères du port4
237	PRCRLF	Transfert d'une chaîne de caractères au port3
236	HEX2DEC	Conversion d' une valeur héxadécimale en ASCII codée décimal
235	GETHEX	Conversion d' une valeur ASCII en héxadécimal
234	PUTHEX	Conversion d' un digit héxadécimal en ASCII
233	PNTEHX	Conversion de 2 digits héxadécimal en ASCII
232	PNT4HEX	Conversion de 4 digits héxadécimal en ASCII
231	PNT6HX	Conversion de 6 digits héxadécimal en ASCII
230	PNT8HX	Convertie 8 digit héxadécimal en ASCII
229	START	Initialisation du système et retour au moniteur
228	TUTOR	retour au moniteur, Affichage du prompt
227	OUT1CR	Transfert d'une chaîne de caractères avec 'CR' et 'LF' au port1
226	GETNUMA	Conversion ASCII codé héxadécimal en héxadécimal
225	GETNUMD	Conversion ASCII codé décimal en héxadécima
224	PORTIN1N	Lit une chaîne de caractères du port1, fin de ligne non automatique
223-128	---	Réservée
127-0	---	Fonctions utilisateur à définir

CONCLUSION :

Au terme de cette étude, qui nous a été profitable sur de nombreux plan, nous avons eu l'occasion d'approfondir nos connaissances dans le domaine de la mico-électronique en générale et celui des systèmes à microprocesseurs en particulier.

La conception et la réalisation des différents modules de la carte UC, nous a permis de mettre en pratique nos connaissances théoriques acquises tout le long de notre formation . Enfin, le stage effectué a l'ENSI nous a permis d'acquérir une certaine expérience professionnelle enrichissante.

Le kit d'initiation FC68 qui a fait l'objet de ce travail, a été conçu de manière a supporté plusieurs extensions et permet grâce à son soft de développer plusieurs applications pouvant même aller jusqu'à substituer le moniteur existant par un moniteur doter d'un vrai éditeur de page et la possibilité de programmer en langages évolués courant tel BASIC, PASCAL etc..

Enfin, nous espérons que ce travail sera poursuivi pour aboutir à la réalisation de nouvelles applications autre que celle pour laquelle il été destiné.

Annexe A:

REGLES ET METHODES DE CONCEPTION

La mise au point d'un système logique complexe nécessite la connaissance de certaines règles et méthodes heuristiques de conception.le présent chapitre donnera un aperçu assez large de ces règles avant de passer à l'étude et la description du kit pédagogique FC68.

3.1 Règles et méthodes heuristiques de conception:

Il est nécessaire d'être méthodique lors de la conception de système logique complexe pour aboutir à un système opérationnel. Un organigramme est donné à cet effet(voir fig A.1).

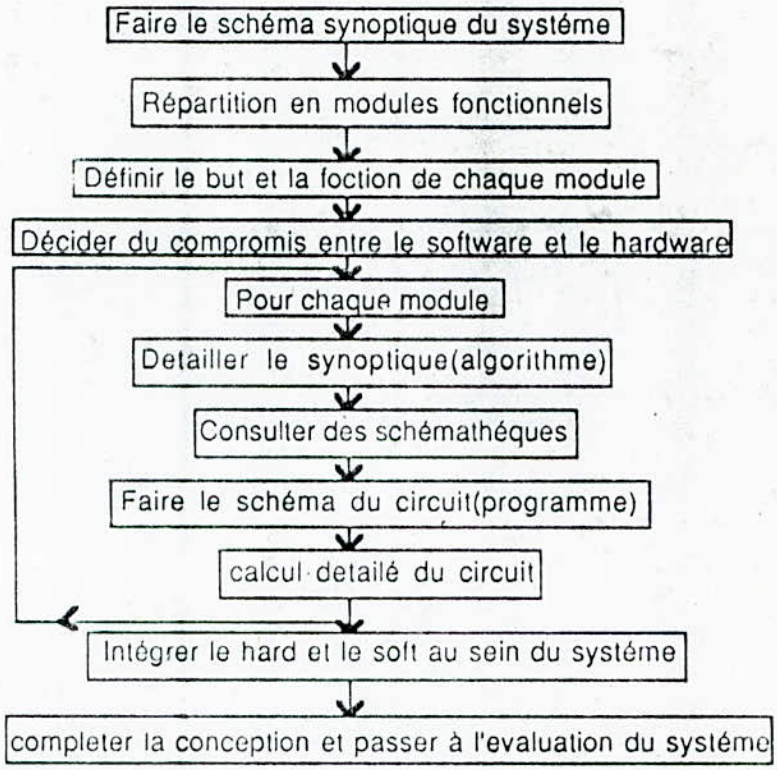


Fig A.1

La méthode à elle seule n'est pas suffisante, la connaissance de certaines règles techniques contribue pour beaucoup dans le bon déroulement de la réalisation et l'implantation du système.

3.1.1- Règles de conception HARDWARE:

La plupart des circuits logiques sont de la famille TTL ou CMOS. Chacune des deux familles offre certains avantages qu'il est utile de connaître. Généralement le besoin de rapidité indique particulièrement la famille TTL, l'immunité au bruit et la consommation minimale de puissance

sont les points forts de la famille CMOS.

l'interconnexion de circuits intégrés de familles différentes peut être la source de problèmes pour le bon fonctionnement du système. Alimentés sous 5V les circuits CMOS délivrent un 0 logique comme ayant un niveau de tension compris entre 0 et 1.5 V, le 1 logique compris entre 3.5 et 5 V. Les circuits TTL quant à eux reconnaissent un 0 présentant un niveau de tension inférieur à 0.8 V.La figure A.2 résume les niveau de tension permis en entrée et délivrés en sortie par un circuit TTL typique.

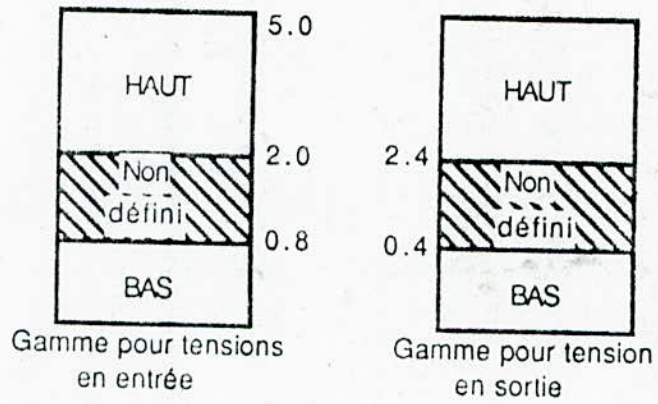


Fig A.2 Niveaux de tension pour un circuit TTL typique

On voit bien qu'un circuit CMOS ne peut pas attaquer directement un circuit TTL. L'inverse est toutefois possible. Au sein d'une même famille des incompatibilités similaires peuvent se poser. On va s'intéresser à l'étude des paramètres électriques et temporels des circuits intégrés de la famille TTL. En e aminant un DATA BOOK, on remarque l'existence de 3 types d'étages de sorties à savoir TOTE M-POLE, collecteur ouvert et 3 états. La différence entre ces derniers réside dans le fait que,pour les circuits ayant une sortie à collecteur ouvert le transistor de sortie n'est pas chargé par une résistance. La conception de systèmes logiques peut se en utilisant uniquement des dispositifs TOTEM-POLE et à 3 états; seulement le besoin de connecter les sorties de plusieurs étages ne peut se faire qu'en utilisant les dispositifs à collecteur ouvert ou 3 états. Les avantages qu'offrent les dispositifs à collecteur ouvert peuvent se résumer en deux points:

-Sortance réglable et élevée.

-Possibilité de faire un et/ou logique câblé d'ou l'économie d'une porte logique.

Les deux premiers types de dispositifs ont les mêmes caractéristiques d'entrée et doivent être traités comme équivalents lors de l'établissement des logigrammes.

On dans le tableau ci-après, un résumé des principaux paramètres électriques et temporels qu'on peut trouvé dans les DATA BOOK (voir figure A.3).

CARACTERISTIQUES	CONDITIONS DE MESURE	TYP			UNITE
		Min	Nom	Max	
Vcc:tension d'alimentation		4.75	5	5.25	V
I_{OH}:courant de sortie,niveau haut				-400	µA
I_{OL}:courant de sortie,niveau bas				8	mA
T_A:température de fonctionnement		0		70	ºC
V_{IH}:tension d'entrée,niveau haut		2			V
V_{IL}: tension d'entrée,niveau bas				0.8	V
V_{IK}:tension d'écrêtage d'entrée	Vcc=Min, I_I=-18mA			-1.5	V
V_{OH}:tension de sortie,niveau haut	Vcc=Min,V_{IL}=VIL Max	2.7	3.4		V
	I_{OH}=Max				
VOL:tension de sortie,niveau bas	& Vcc=Min I_{OL}=Max		0.25	0.5	V
	V_{IH}=2 V I_{OL}=4 mA			0.4	V
I_I:courant d'entrée,pour Vcc=Max	Vcc=Max,V_I=7 V			0.1	V
I_{IH}:courant d'entrée,niveau haut	Vcc=Max,V_{IH}=2.7 V			20	µA
I_{IL}:courant d'entrée,niveau bas	Vcc=Max,V_{IL}=0.4 V			-0.4	mA
I_{OS}:courant de sortie au niveau haut en court-circuit	Vcc=max	-20		-100	mA
I_{CCH}:courant total d'alimentation avec sorties au niveau haut	Vcc=Max		0.8	1.6	mA
I_{CCL}:courant total d'alimentation avec sorties au niveau bas	Vcc=Max		2.4	4.4	mA
I_{CC}:courant moyen par porte avec un rapport cyclique de 50%	Vcc=Max		0.4		mA
t_{PLH}:temps de propagation du niveau haut au niveau bas	C_L=15pF R_L=2KΩ Vcc=5 V T_A=25ºC		9	15	ns
t_{PLH}:temps de propagation du niveau bas au niveau haut	C_L=15pF R_L=2KΩ Vcc=5 V T_A=25ºC		10	15	ns

Fig A.3 Paramètres d'un circuit TTL typique

Certaines règles ont été établies pour faciliter l'utilisation des circuits intégrés vue la diversité des paramètres électriques et temporels de ces derniers.

a) Règles relatives au niveau des signaux et à l'interfaçage des boîtiers:

- Utiliser les spécifications les plus défavorables pour faire les calculs.
- Connecter les entrées non utilisées au VCC à travers une résistance de un KΩ. Les circuits TTL-LS possèdent une diode interne et peuvent être connectés directement au VCC.
- Examiner la sortance de tout les dispositifs, amplifier si nécessaire.
- Connecter un maximum de 10 entrées de portes 74XX à une seule sortie de porte 74XX.
- Connecter un maximum de 10 entrées de portes 74LSXX à une seule sortie de porte 74LSXX.
- Connecter un maximum de 20 entrées de portes 74LSXX à une seule sortie de porte 74XX,le calcul de la sortance est toutefois indiqué vue la diversité des courants absorbés par les entrées des dispositifs 74LSXX.
- Connecter un maximum de 3 entrées de portes 74XX à une seule sortie de porte 74LSXX.
- Utiliser des dispositifs à collecteur ouvert pour un besoin de et/ou câblé, remplir les fonctions d'un relais et l'interfaçage pour un bus.
- Calculer la résistance minimale et maximale à utiliser comme charge pour une sortie à collecteur ouvert. Le choix se fait ensuite pour des considération de vitesse et de puissance consommée.

b) Règles relatives aux paramètres temporels:

- Estimer le temps de propagation d'un signal comme étant la somme des temps les plus défavorables trouvés dans les fiches techniques des composants utilisés.
- Estimer le temps de propagation typique en faisant la moyenne entre t_{plh} et t_{phl}.
- Calculer la fréquence maximale de l'horloge en utilisant le temps de propagation le plus long entre deux dispositifs synchrones (bascules). Utiliser la valeur du temps la plus défavorable pour les portes logiques le long du chemin que traverse le signal.
- Ajuster le temps de propagation suivant la charge utilisée, comparée à la charge trouvé dans le fiches techniques.
- vérifier que la fréquence de l'horloge n'est pas excessive.
- Utiliser des résistances de charge pour chaque porte non utilisée, ceci assurera u meilleur temps de commutation et une meilleure immunité au bruit.
- Vérifier que le rapport cyclique de l'horloge est bien celui spécifié.
- Vérifier que les temps d'accès et de maintient répondent bien aux spécificatio

techniques.

• Utiliser si nécessaire, les états d'attentes du processeur(WAIT STATES) pour ralentir certains cycles du bus afin d'éviter des anomalies de fonctionnement.

• Tous les dispositifs synchrones devraient répondre à la même horloge. Relier toutes les entrées horloge entre elles.

• Ne pas utiliser un inverseur pour obtenir le complémentaire de l'horloge.

• Eviter d'utiliser des dispositifs séquentiels asynchrones.

• Ne pas utiliser les dispositifs logiques à leurs vitesses limites.

c) Règles relatives au bruit:

• Relier toutes les entrées non utilisées au VCC.

• Valider les lignes de communication à l'état bas.

• Utiliser des capacités de 0.01 µF pour chaque deux ou 3 circuits intégrés.

• Séparer les lignes de données des lignes d'horloges.

• Faire les connections les plus courtes possibles.

• utiliser la famille logique la moins rapide tout en respectant les spécifications techniques.

• Protéger la ligne d'horloge ou l'isoler des circuits environnants.

• S'assure que le niveau du bruit est faible relativement aux niveau logiques.

• Distribuer d'une façon équilibrée les lignes d'alimentation et relier toutes les lignes de masse. Utiliser des fils de connexion épais pour alimenter le système.

3.1.2- Méthodes heuristique de conception:

• Ne pas réinventer la roue, lire les fiches techniques et les notes d'application.

• Réduire le problème à des choses qu'on a résolu auparavant.

• Si on ne peut pas honorer le cahier des charges, il faut négocier et poser le problème.

• Il faut toujours avoir une réponse:on doit commencer quelque part.

• Changer une variable à la fois quant on corrige la conception.

• développer les circuits et programmes module par module: corriger au fur et à mesure.

• Utiliser des dispositifs LSI quant c'est possible.

• Chercher le pourquoi des erreurs rencontrées.

• Résoudre le vrai problème.

• Quant le doute s'installe revoir et s'assurer

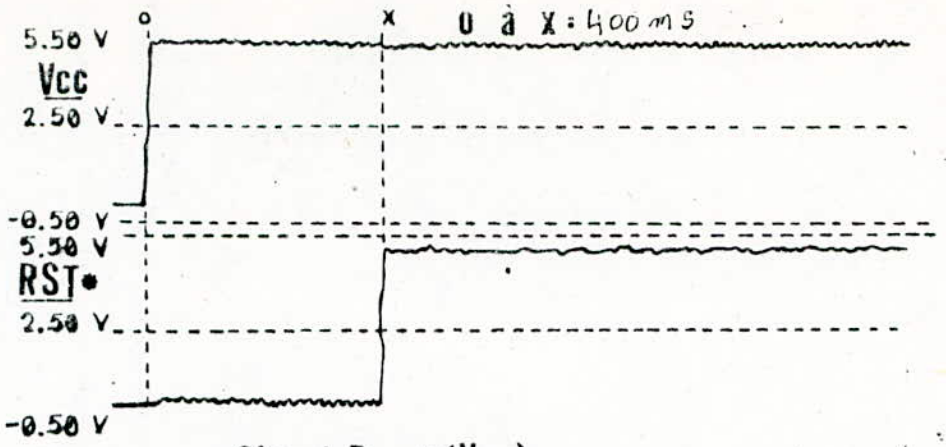
• Agir plutôt que réagir.

• Se fixer un délai pour accomplir chaque tâche, résoudre les problèmes les plus difficiles le plus tôt possible.

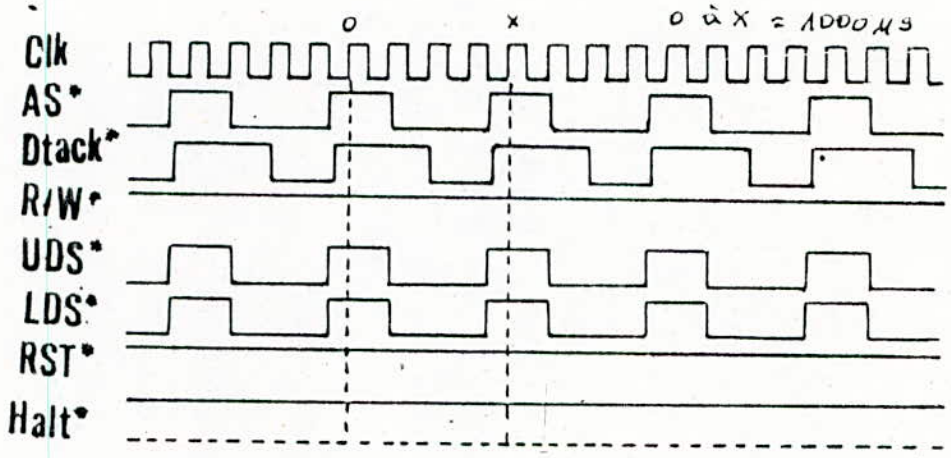
• Fixer une date limite et revoir la progression du travail

ANNEX B

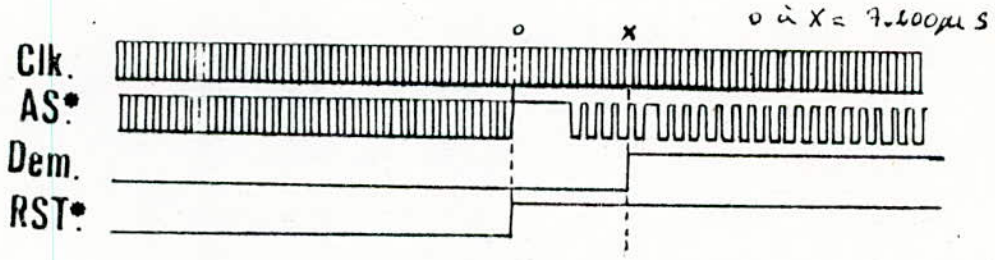
SCHEMATHEQUE
Localisation des circuits, modules et SWITCHs.



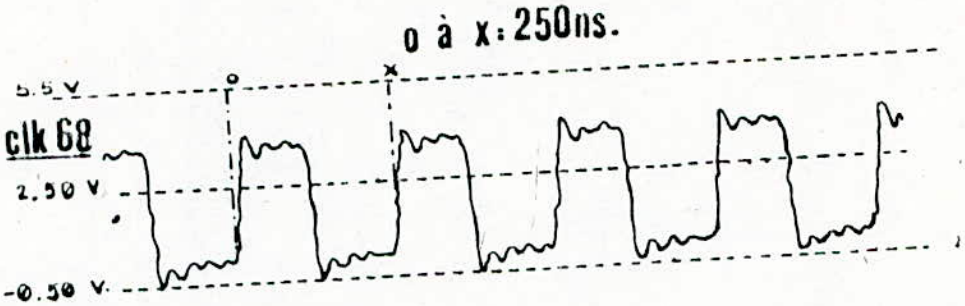
Signal Reset (Vcc)



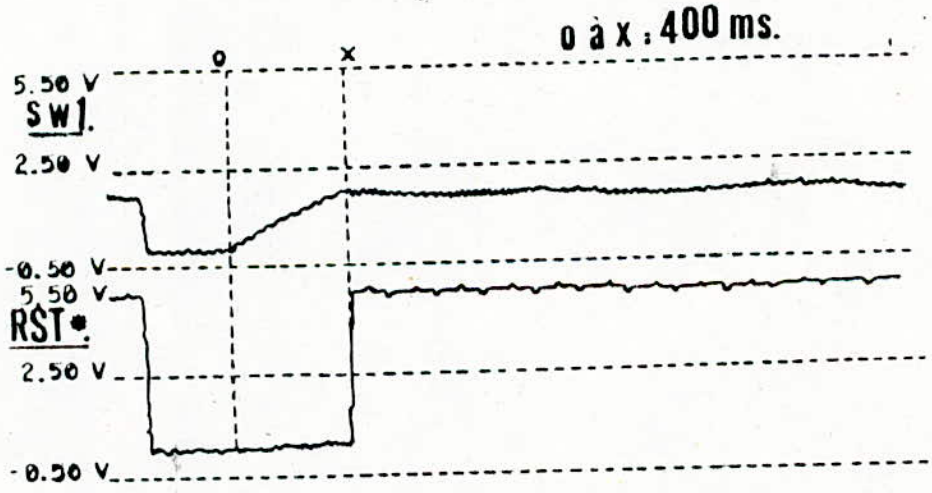
"Fonctionnement libre" à zéro Etat d'attente.



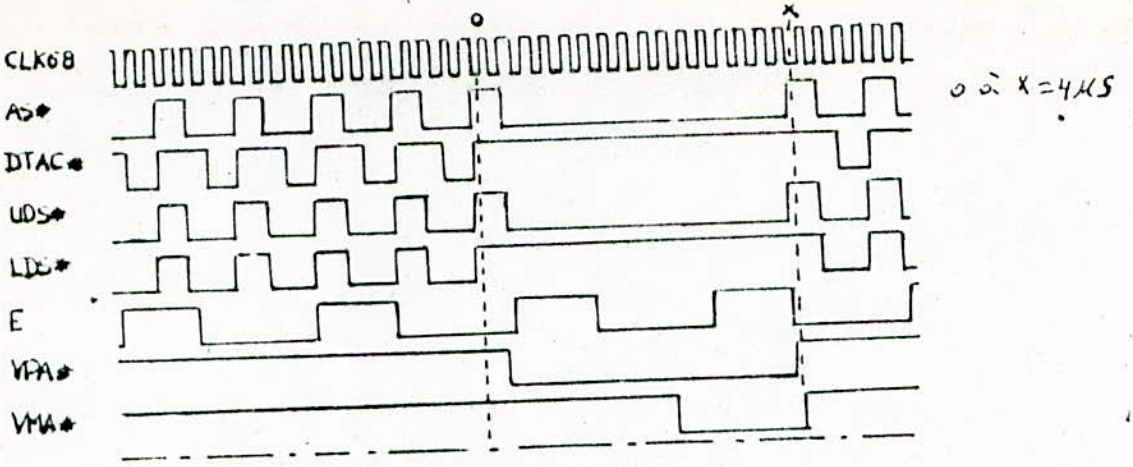
Signal "Dem" au reset.



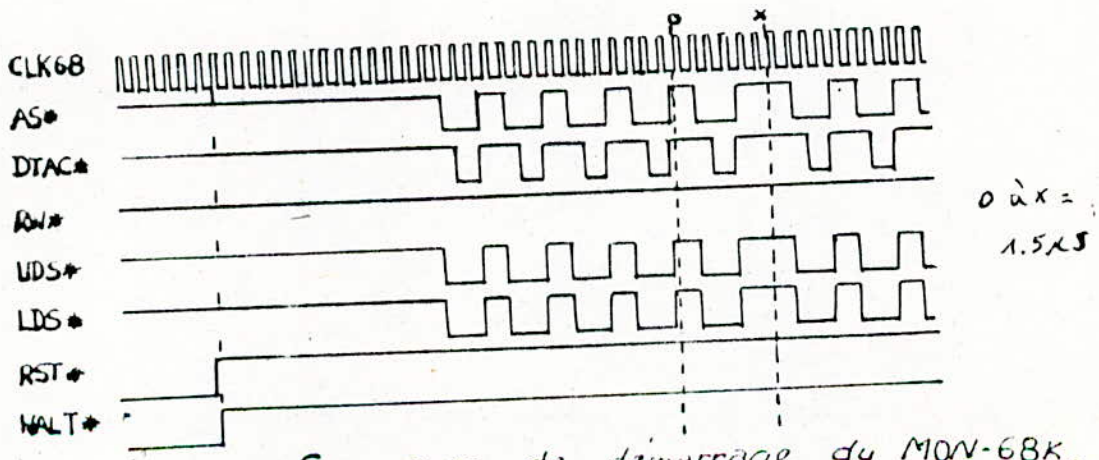
"Clk 68" freq = 4Mhz.



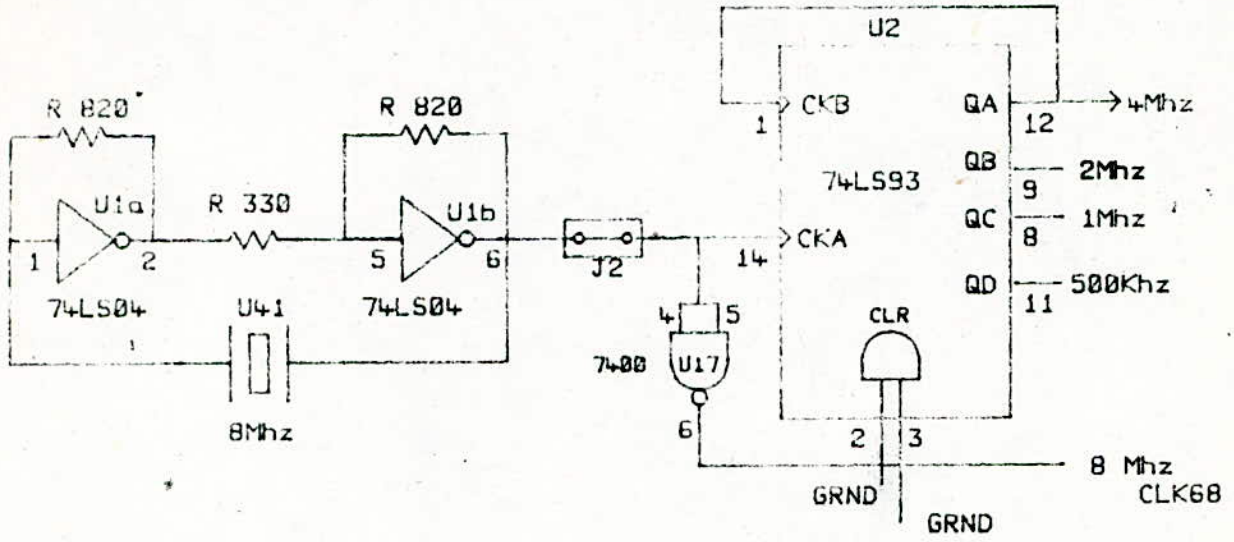
Signal Reset (switch).

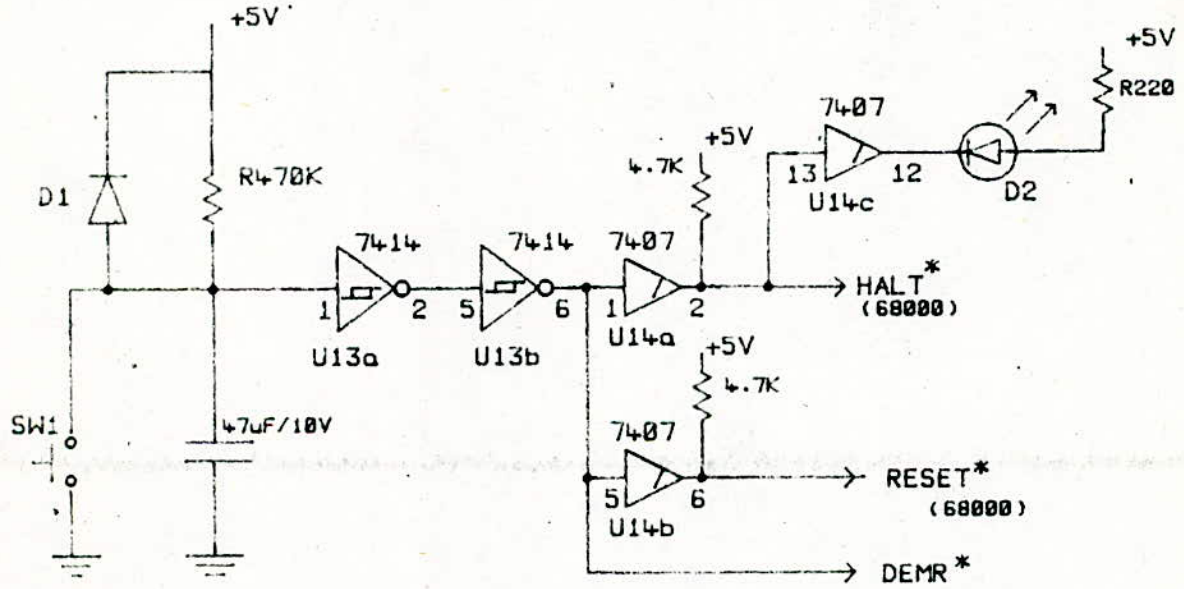


Cycle bus Synchrone.

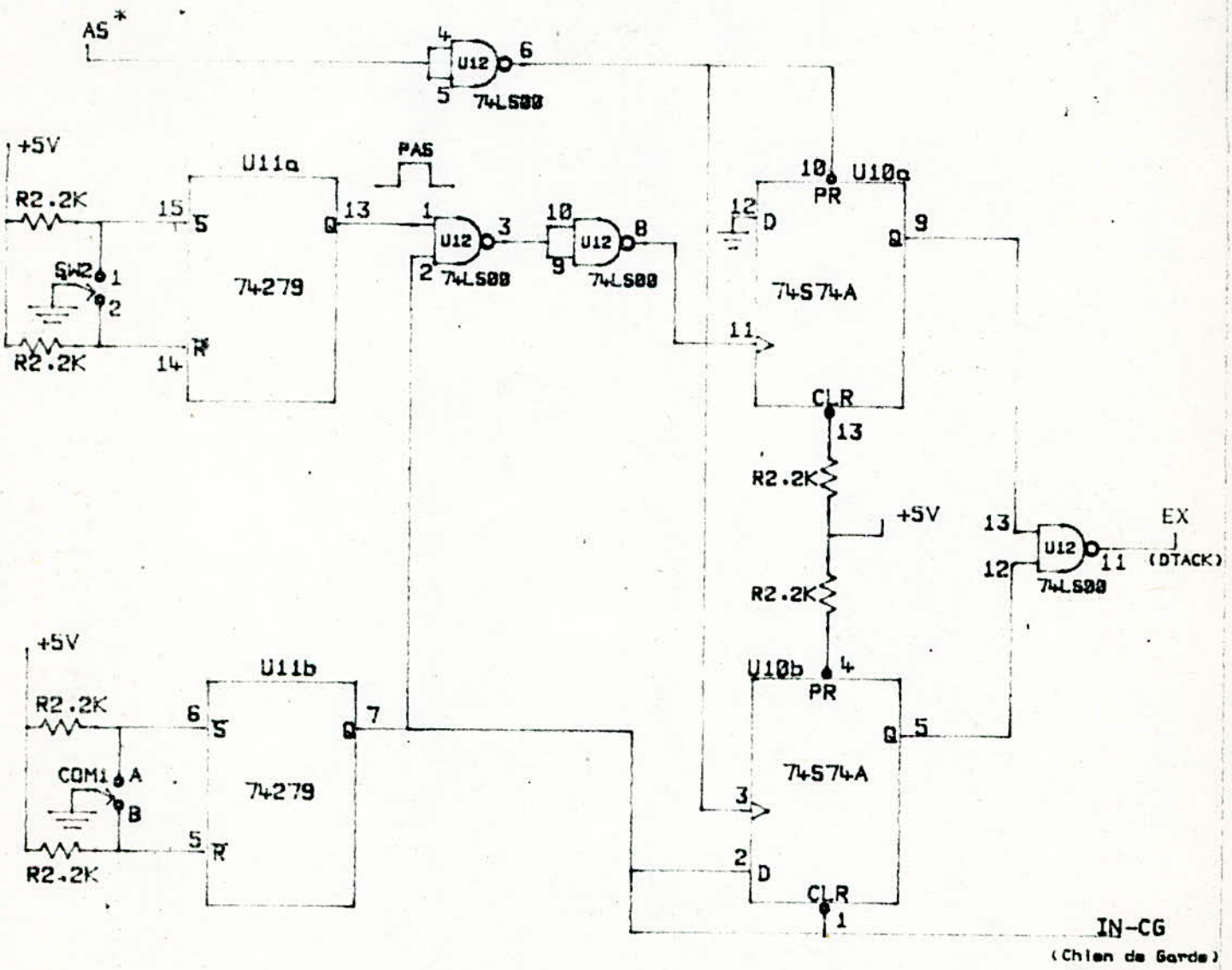


Sequence de démarrage du MON-68K.

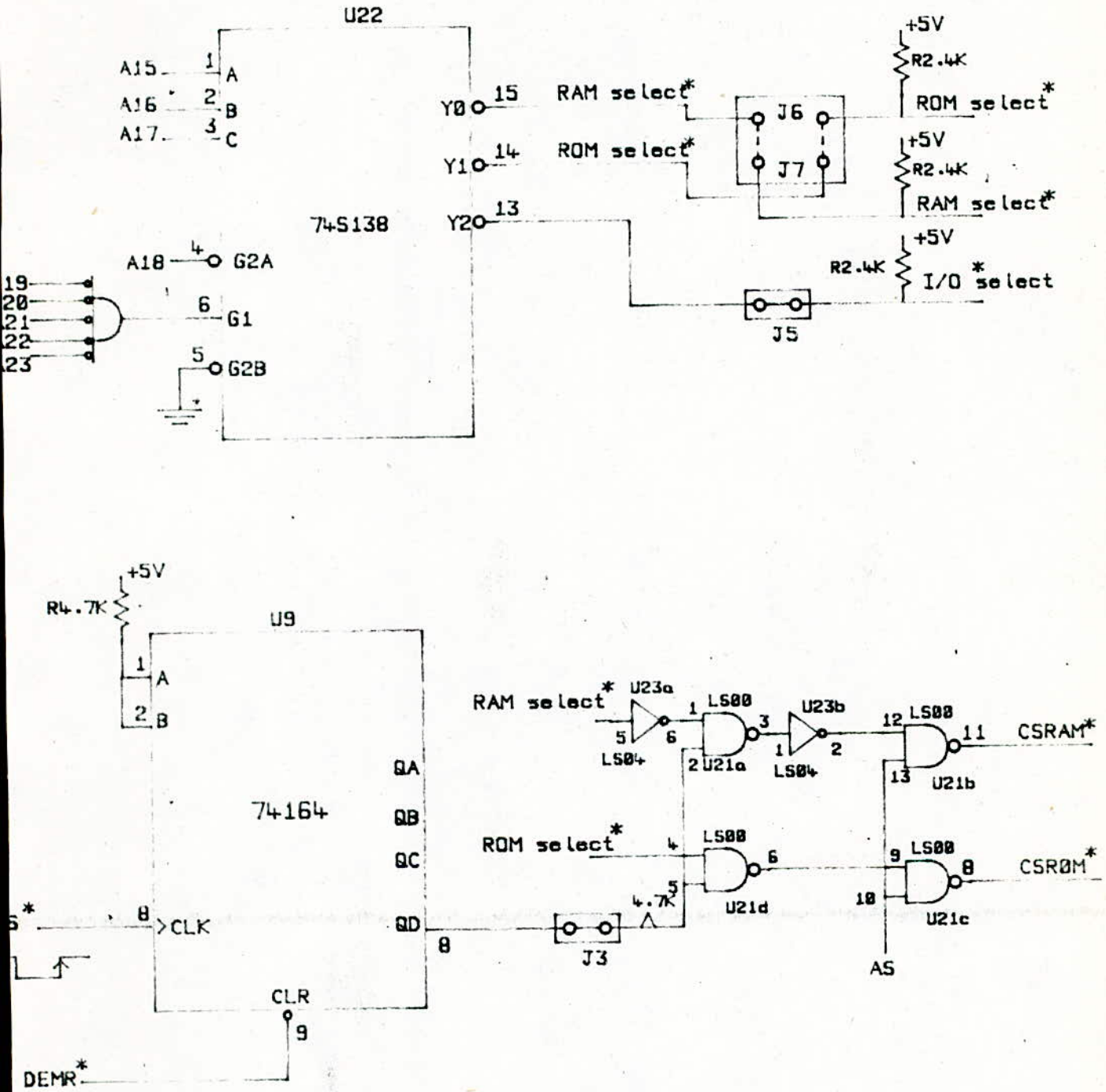




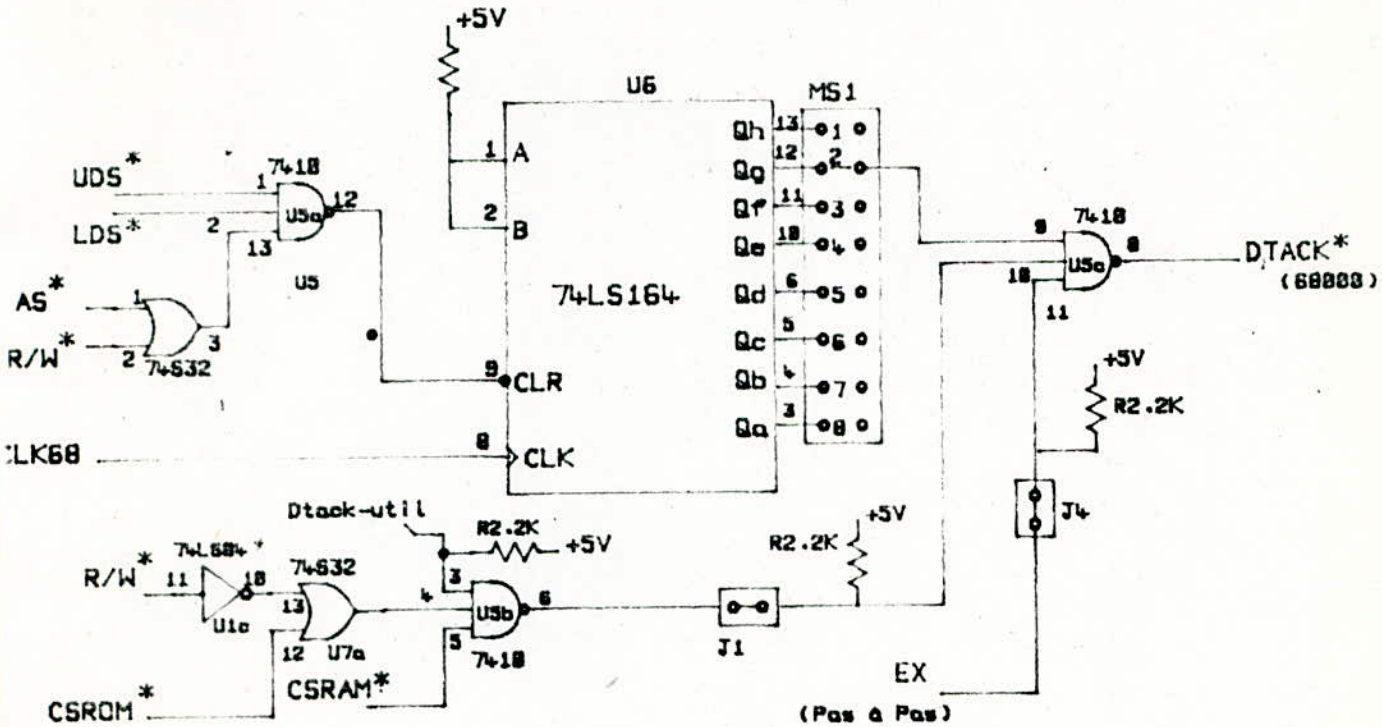
FC68	ENSI/ENP	EL HARRACH ALGER 1989
EXECUTE : F.C 5/89	CIRCUIT D'INITIALISATION ET MODULE D'HORLOGE	
REVISIONS		
TEST	ECHELLE	SCHEMA N° 1/8

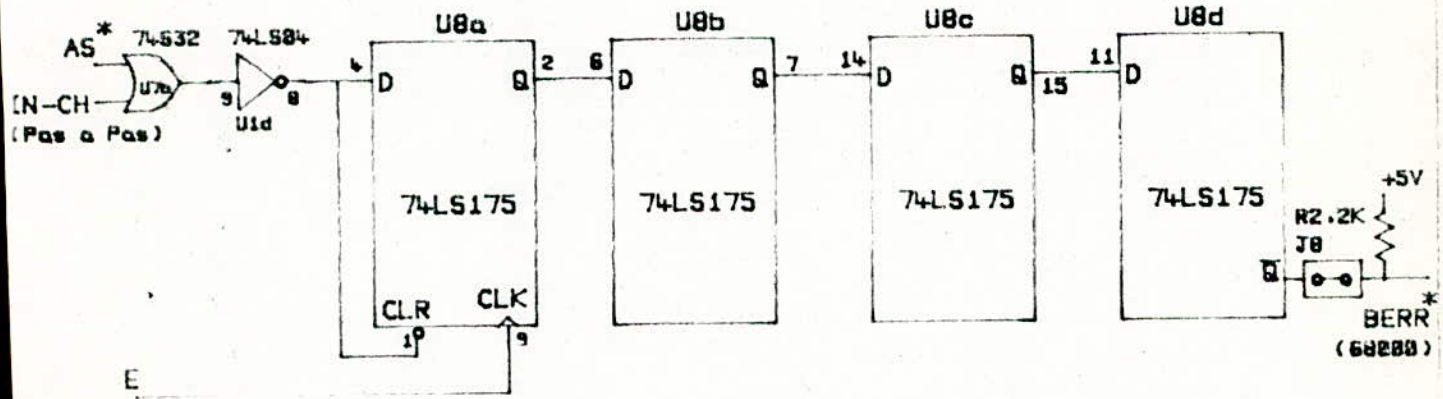


FC68	ENSI/ENP	EL HARRACH ALGER 1989
EXECUTE: F.C 5/89	TITRE: MODULE PAS à PAS	
REVISIONS		
TEST	ECHELLE	SCHEMA N°2/8

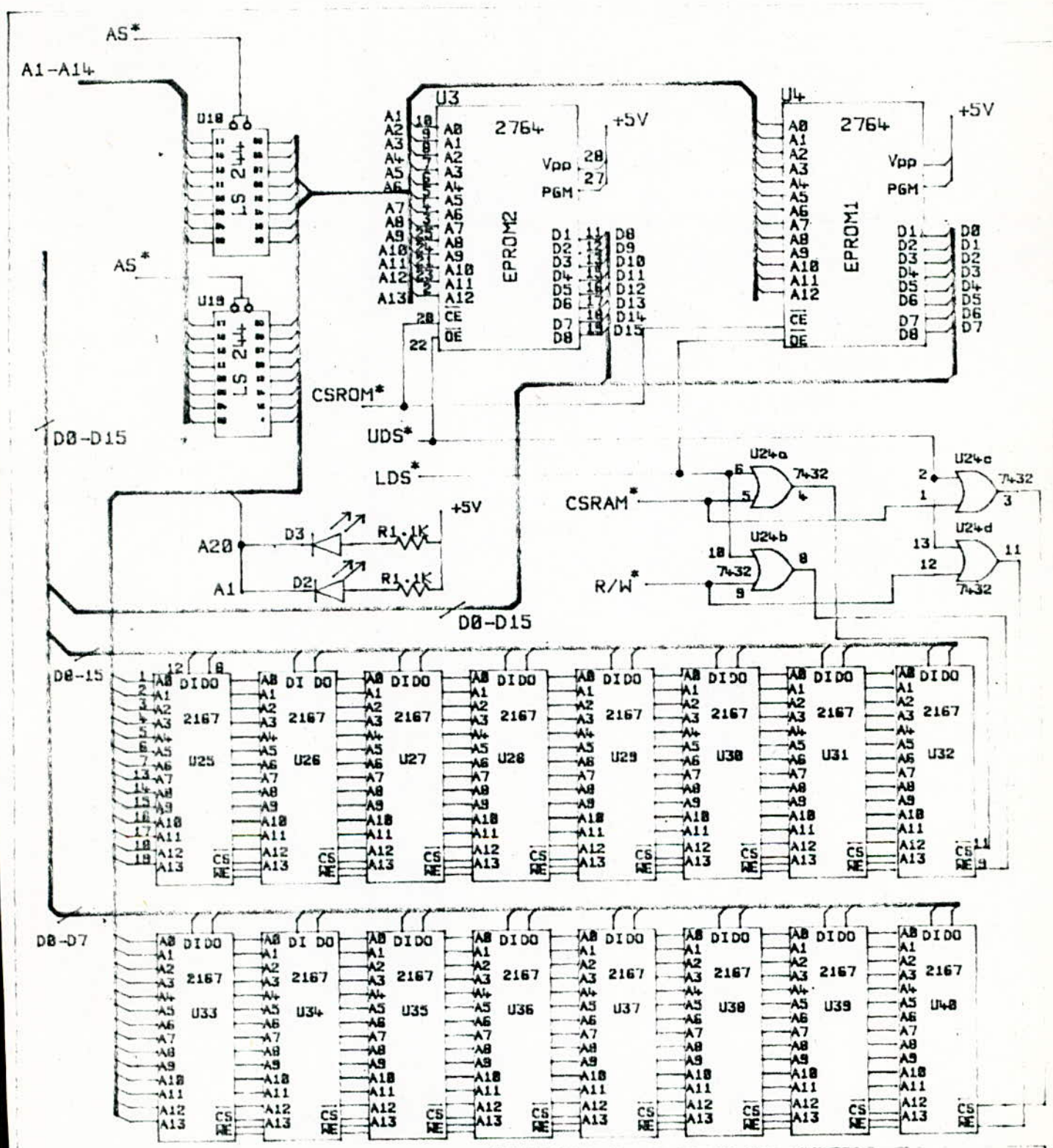


FC68	ENSI/ENP	EL HARRACH ALGER 1989
EXECUTE : F.C 5/89	MODULE DE DECODAGE ET CIRCUIT DE DEMARRAGE	
REVISIONS		
TEST	ECHELLE	SCHEMA N° 8

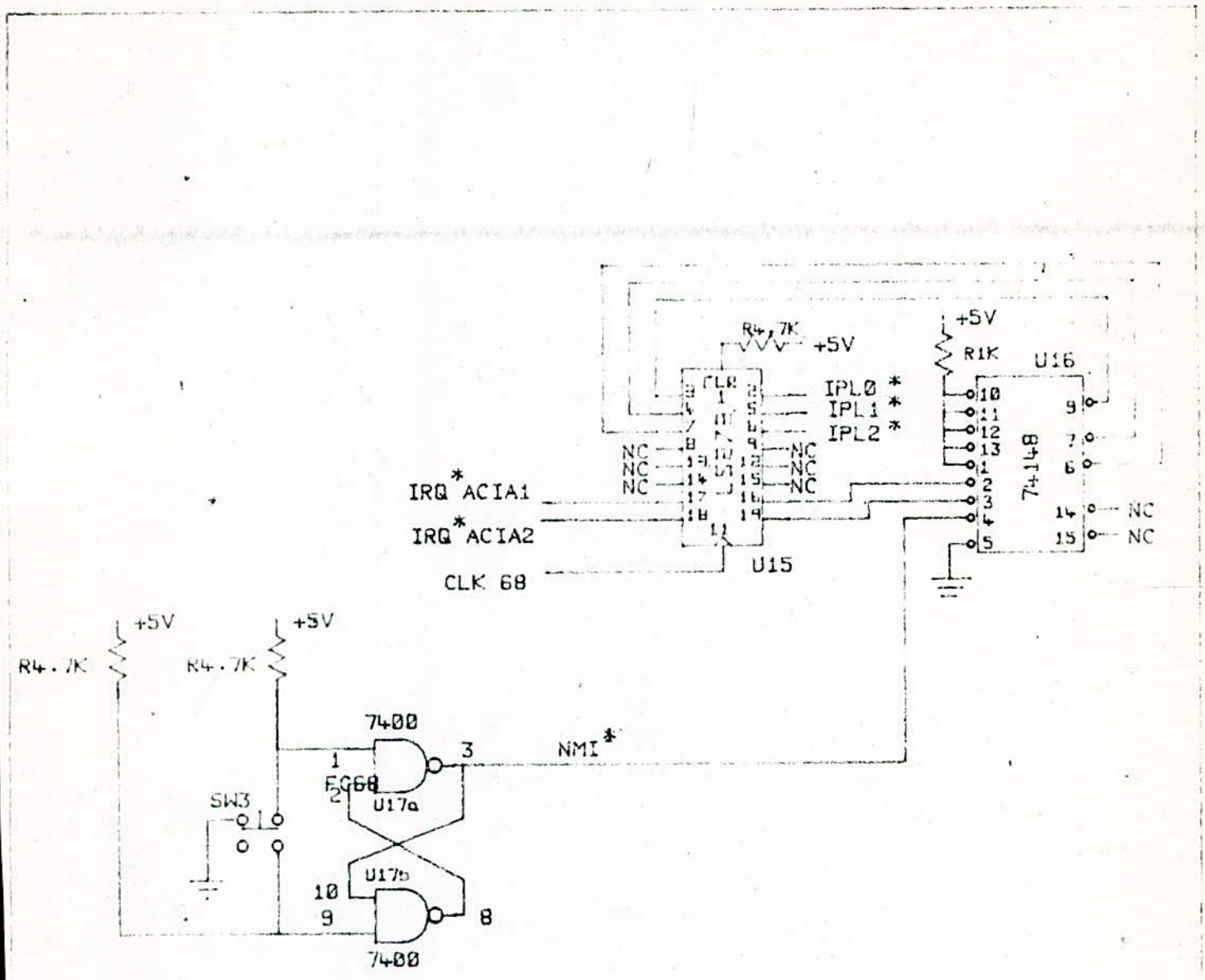




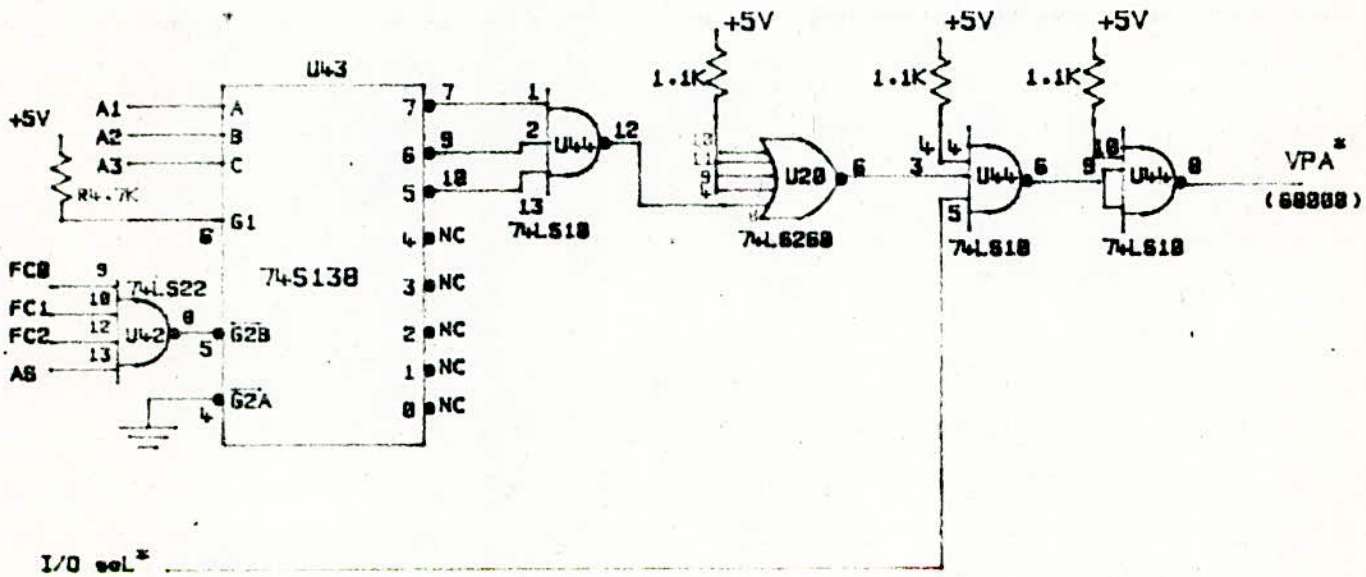
FC68	ENSI/ENP	EL HARRACH ALGER 1989
EXECUTE : F.C 5/89	TITRE : CIRCUIT DTACK * ET CIRCUIT CHIEN DE GARDE	
REVISIONS		
TEST	ECHELLE	SCHEMA N° 4/8



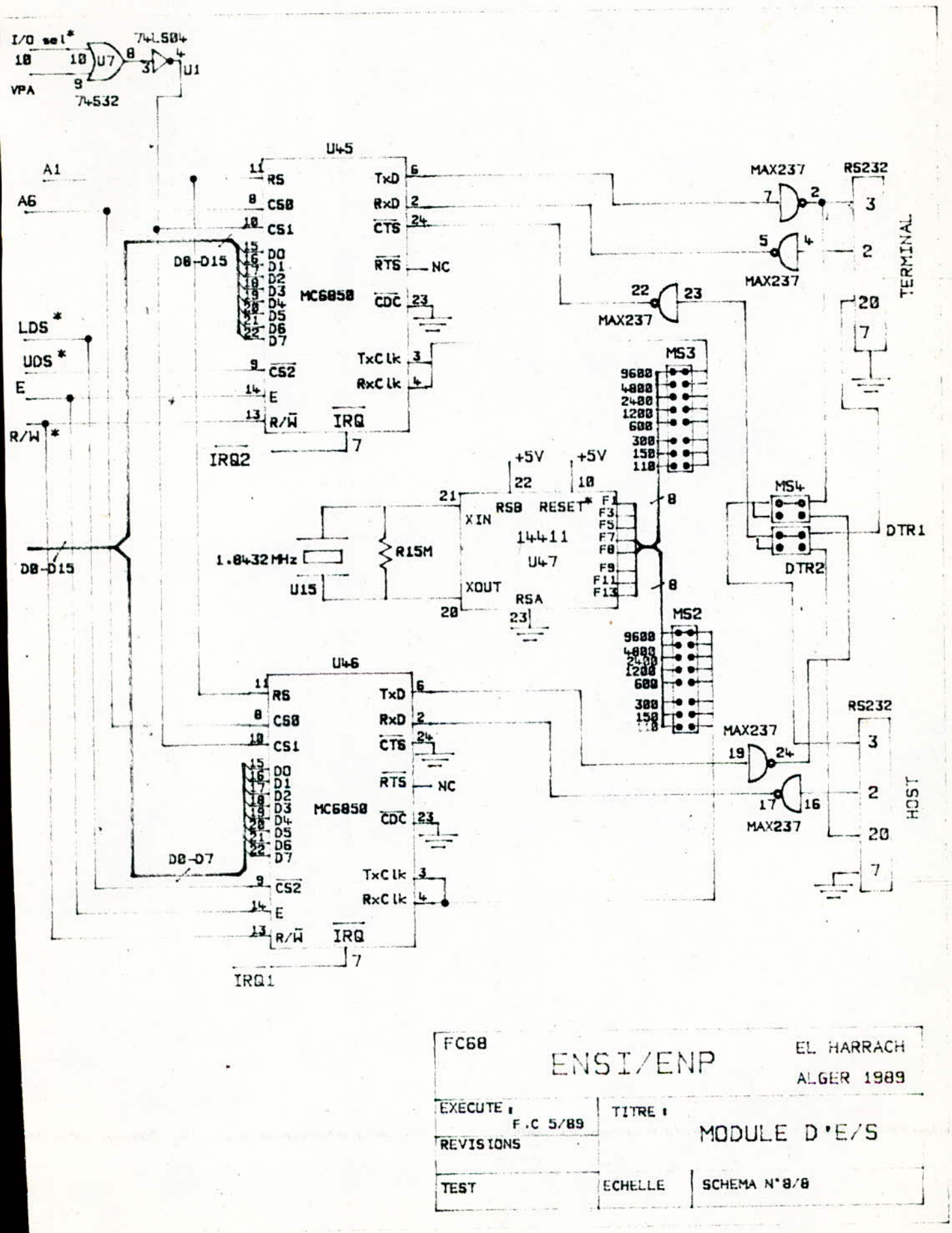
FC68	ENSI/ENP	EL HARRACH ALGER 1989
EXECUTE : F.C 5/89	TITRE : MODULE MEMOIRE EPROM . RAM	
REVISIONS		
TEST	ECHELLE	SCHEMA N°5/8



FC68	LNSI/ENP	EL HARRACH ALGER 1989
EXECUTE : F.C 5/89	TITRE : MODULE D'INTERRUPTIONS	
REVISIONS		
TEST	ECHELLE	SCHEMA N° 6/8

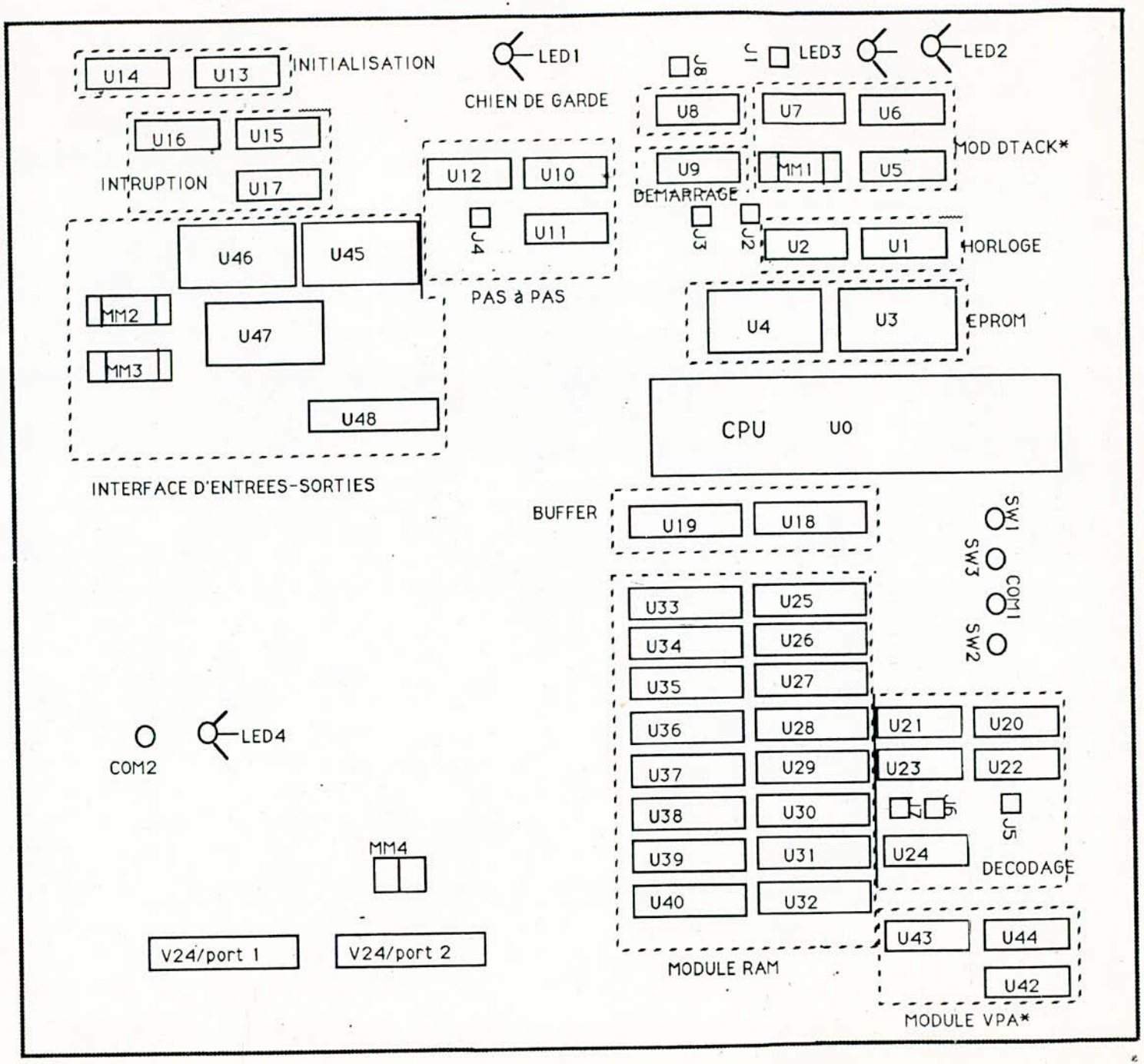


FC68	ENSI/ENP	EL HARRACH ALGER 1989
EXECUTE : F.C 5/89 REVISIONS	TITRE : MODULE VPA*	
TEST	ECHELLE	SCHEMA N° 7/8

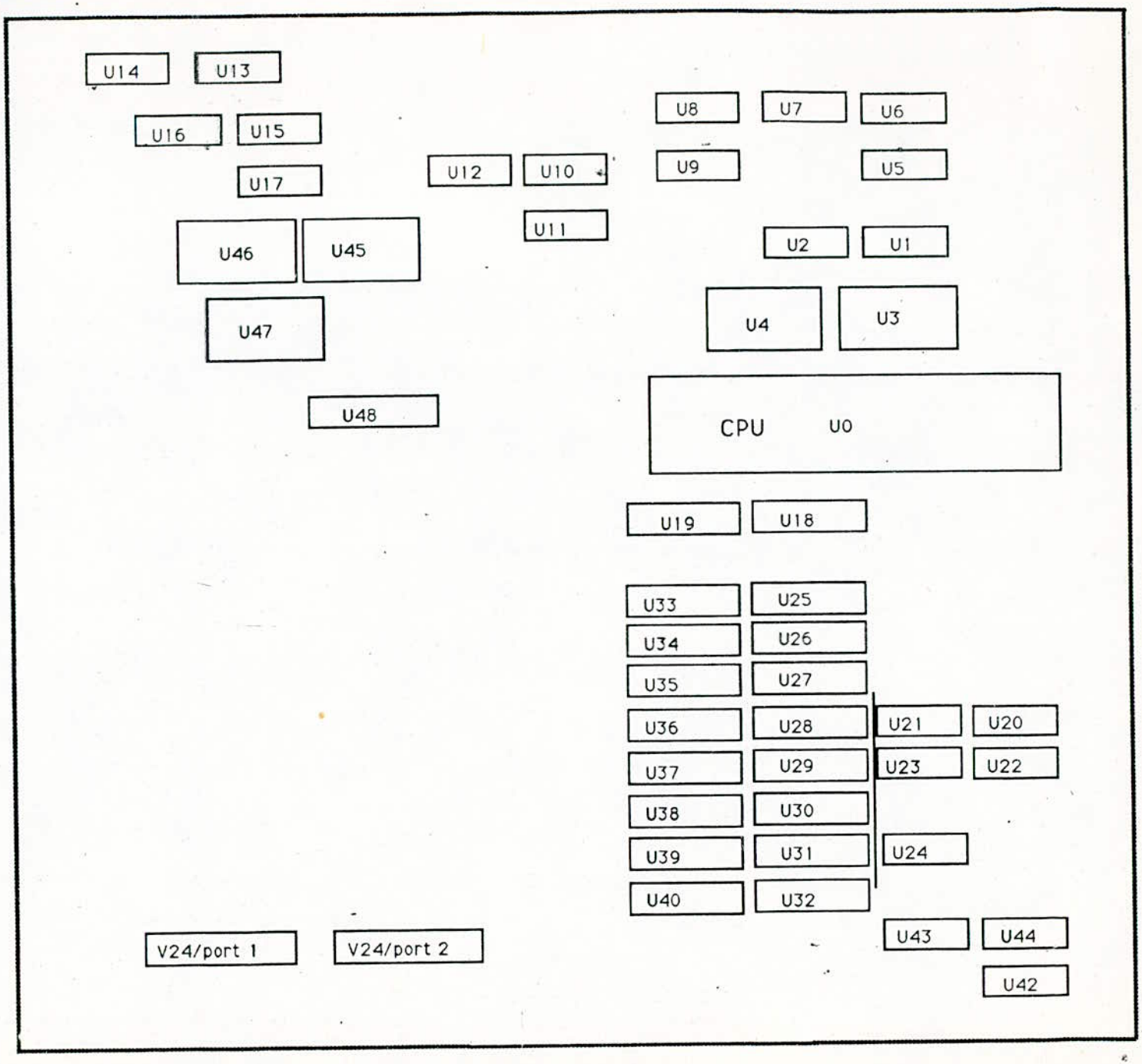


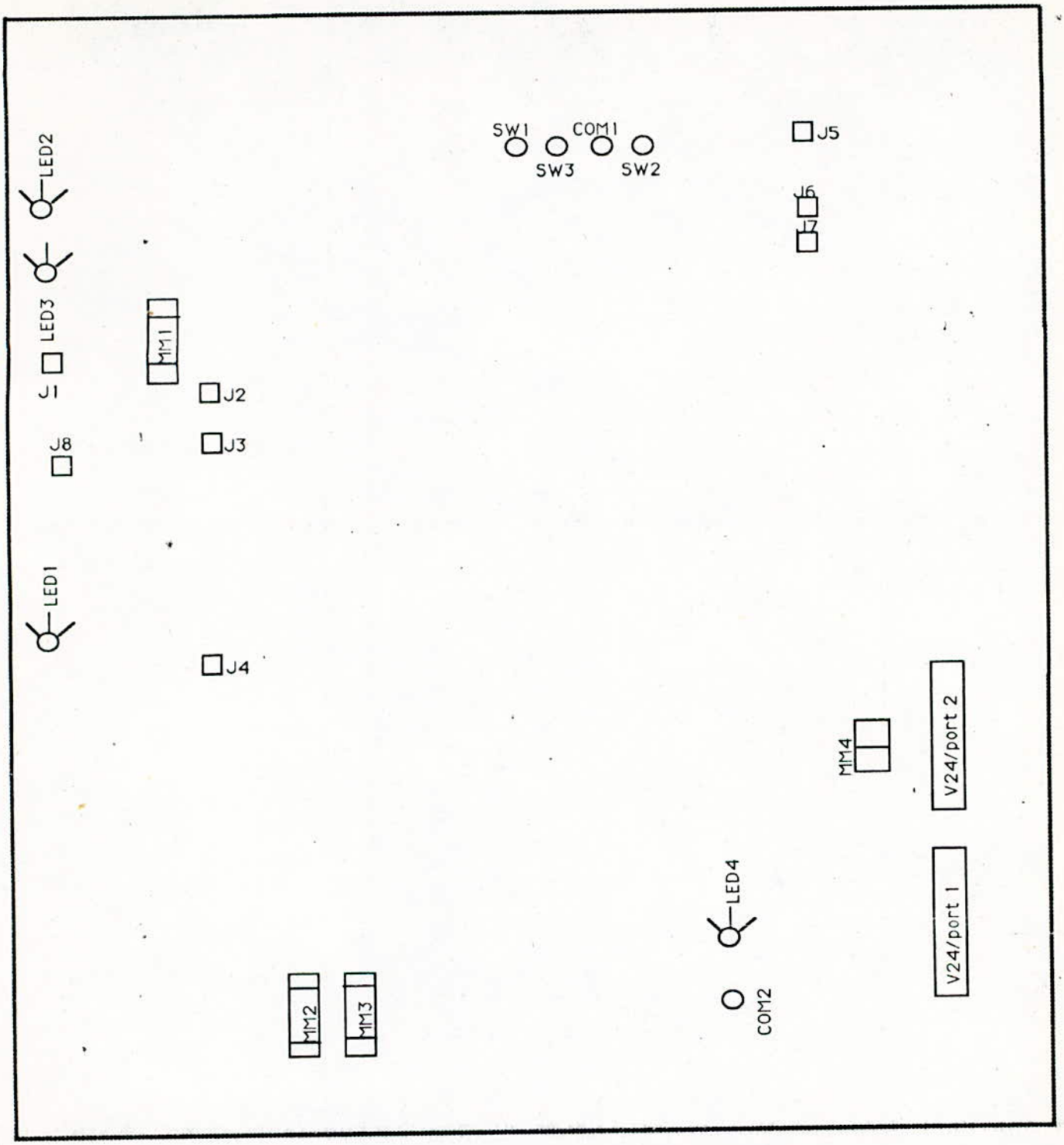
FC68	ENSI/ENP	EL HARRACH ALGER 1989
EXECUTE : F.C 5/89	TITRE : MODULE D'E/S	
REVISIONS		
TEST	ECHELLE	SCHEMA N°8/8

LOCALISATION DES MODULES



LOCALISATION DES CIRCUITS





LOCALISATION DES CAVALIERS
ET SWITCHS

Annexe C

SOMMAIRE DES CAVALIERS, INTERRUPTEURS ET BOUTONS POUSSOIRS

1) Cavaliers:

fonction	Désignation	Présence	Observation
Adressage de tout le champs mémoire	J1	Non	Utilisation pour auto-test
Déconecter l'horloge système	J2	Non	Utilisation d'horloge externe
Inhiber la fonction démarrage	J3	Non	Implanter un autre moniteur
Interdiction de l'accès aux interfaces sérielles	J5	Non	Utilisation pour auto-test
Iterdiction de l'accès à la RAM et l'EPROM	J6,J7	Non	Utilisation pour auto-test
Inhiber le "chien de garde"	J8	Non	Dépannage et auto-test

Cas particuliers , Cavaliers J6 et J7:

Adresse de L'EPROM débute à l'adresse 8000H

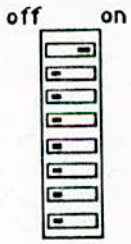
Adresse de L'EPROM débute à l'adresse 0000H

Utilité: Reconfiguration de la carte mémoire, plus grande fléxibilté de la carte.

2) Les microswitchs:

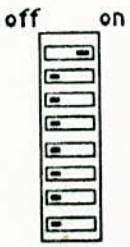
Fonction	Désignation	Observation
Choix du nombre d'états d'attent	MM1	Implantation de dispositifs lents.
Choix de BAUD RATE	MM2	Port 1
Choix de BAUDE RATE	MM3	Port 2

MM1:



1 état d'attente
2 états d'attente
3 états d'attente
4 états d'attente
5 états d'attente
6 états d'attente
7 états d'attente
sans états d'attente

MM2,MM3 : La figure ci-après indique le choix de 9600 BAUD (bits par seconde).

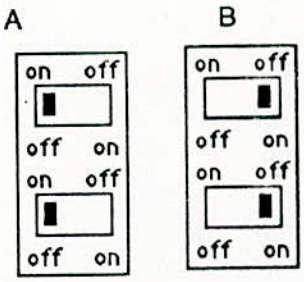


9600 BAUD
4800 BAUD
2400 BAUD
1200 BAUD
600 BAUD
300 BAUD
150 BAUD
100 BAUD

MM4 :Deux les quatres positions possibles, deux seulement devrait être choisies par l'uitilisateur réspectivement pour relier ou non les deux ports.

A: Ports non reliés

B: ports reliés.



3) Boutons poussoirs:

Fonction	Désignation	Observation
Réalise l'initialisation	SW1	
Exécution d'un pas	SW2	Mode pas à pas
Arrête l'exécution d'un programme	SW3	Interruption non masquable

4) LEDs :

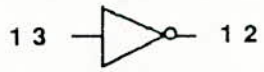
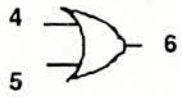
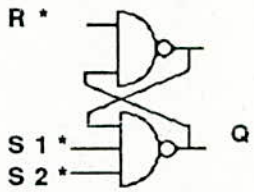
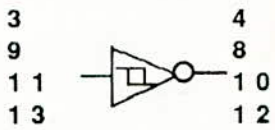
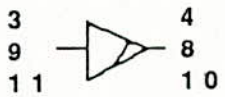
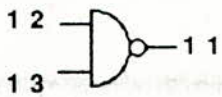
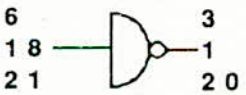
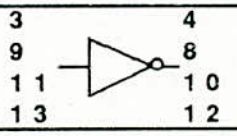
LED1 : Allumée, elle indique l'état halt du processeur.
LED2 : Clignautant, elle indique le bon foctionnement du KIT FC68 sous l'auto-test.
LED3 : Indique l'état du bit A1, utilisée pour le dépannage et le mode pas à pas.
LED4 :Témoin de la mise sous tension du KIT.

5) Interrupteurs:

Fonction	Désignation	Observation
Sélection du mode pas à pas	COM1	
Alimentation du KIT	COM2	

Annexe D:

Index des modules, cicuits et portes non utilisées

LOCALISATION	Désignation	BROCHAGE
U1	SN74LS04	
U7	SN74S32	
U11	SN74279	S1* 11 | 2 S2* 12 | 3 R* 10 | 1 Q 9 | 4 
U13	SN7414	
U14	SN7407	
U17	SN7400	
U48	MAX237	
U23	SN74LS04	

Portes non utilisées

MODULE	Repère des circuits intégrés	Shéma numéro
RESET	U13, U14	1
Horloge	U1, U2, U17, U41	1
Interruption	U15, U16, U17	6
EPROM	U3, U4	5
RAM	U25-U40, U24	5
Pas à Pas	U10, U11, U12	2
BUFFER D'adresse	U18, U19	5
VPA*	U20, U42, U43, U44	7
I/O *	U1, U7, U45, U46 U47, U48, U49	8
Décodage	U20, U21, U22, U23	3
Démarrage	U9	3
DTACK*	U1, U5, U6, U7	4
Chien de garde	U8	4
CPU	U0	-

Index des modules

Repère	NOMBRE	Désignation
UO	1	MC68000
U1	1	SN74LS04
U2	1	SN74LS93
U3, U4	2	AMD2764
U5	1	SN7410
U6	1	SN74LS164
U7	1	SN74S32
U8	1	SN74LS175
U9	1	SN74164
U10	1	SN74S74
U11	1	SN74279
U12	1	SN7400
U13	1	SN7414
U14	1	SN7407
U15	1	SN74273
U16	1	SN74148
U17	1	SN7400
U18	1	SN74LS244
U19	1	SN74LS244
U20	1	74S260
U21	1	SN74S00
U22	1	SN74S138
U23	1	SN74LS04
U24	1	SN7432
U25-40	16	µPD 2167
U41	1	QUARTZ 8MHz
U42	1	SN74LS22
U43	1	SN74S138
U44	1	SN74LS10
U45-U46	2	MC6850
U47	1	MC14411
U48	1	MAX 237 CNG
U49	1	QUARTZ 1.843 MHz

Index des circuits intégrés

Annexe E:

CARACTERISTIQUES DE CI UTILISES

EPROM 2764

MC 14411

RAM STATIQUES 51C67 INTEL

ADVANCE INFORMATION

51C67
HIGH SPEED CHMOS 16,384 x 1 BIT STATIC RAM

	51C67-35
Max. Access Time (ns)	35
Max. Active Current (mA)	100
Max. Standby Current (mA)	10

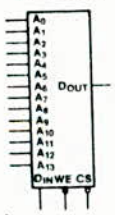
- **Double Metal CHMOS III Technology**
- **Equal Access and Cycle Times**
- **Automatic Power Down**
- **0.8-2.0V Output Timing Reference**
- **High Density 20-Pin Package**
- **Directly TTL Compatible—All Inputs and Output**
- **Separate Data Input and Output, Output Three-State**
- **2147H Upgrade**

The Intel 51C67 is a 16,384-bit static random access memory organized as 16,384 words by 1 bit. This memory is fabricated using Intel's high performance double metal CHMOS III technology, with a 6T cell. This state of the art technology with HMOS III scaled transistors brings high performance to CMOS Static RAMS. The design of the 51C67 offers a 4x density improvement over the industry standard 2147H with compatible performance. The 51C67 offers the automatic power-down feature pioneered by the Intel 2147H.

$\overline{CS}$ controls the power-down feature. In no more than a cycle time after $\overline{CS}$ goes high (deselecting the 51C67), the part automatically reduces its power requirements and remains in this low power standby mode as long as $\overline{CS}$ remains high. This device feature results in system power savings as great as 90% in larger systems where the majority of devices are deselected.

The 51C67 is placed in a 20-pin 300mil package configured with the industry standard 16K x 1 pinout. It is directly TTL compatible in all respects: inputs, output; and a single +5V supply. The data is read out non-destructively and has the same polarity as the input data. A data input and a separate three-state output are used.

LOGIC SYMBOL



PIN CONFIGURATION

PIN NAMES

A_0-A_{13}	ADDRESS INPUTS	V_{CC} POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND GROUND
$\overline{CS}$	CHIP SELECT	
D_{IN}	DATA INPUT	
D_{OUT}	DATA OUTPUT	

TRUTH TABLE

$\overline{CS}$	$\overline{WE}$	MODE	OUTPUT	POWER
H	X	NOT SELECTED	HIGH Z	STANDBY
L	L	WRITE	HIGH Z	ACTIVE
L	H	READ	D_{OUT}	ACTIVE

BLOCK DIAGRAM

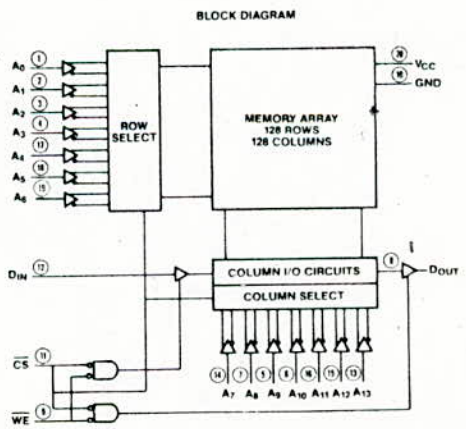


Figure 1. 51C67 Block Diagram

MAY 1984
ORDER NUMBER 231094-001

PRELIMINARY

P2764A
ADVANCED 64K (8Kx8) PRODUCTION EPROM

- **200 nsec Access Time**
  —HMOS II*-E Technology
- **Low Power**
  —60 mA Maximum Active
  —20 mA Maximum Standby
- **Two-line Control**
- **int$_e$ligent Programming™ Algorithm**
  —Fastest EPROM Programming
- **int$_e$ligent Identifier™ Mode**
  —Automated Programming Operations
- **Compatible with 2764, 27128, 27256**

The Intel P2764A is a 5V only, 65,536-bit electrically programmable read-only memory (EPROM). The P2764A is an advanced version of the P2764 and is fabricated with Intel's HMOSII-E technology which significantly reduces die size and greatly improves the device's performance, power consumption, reliability and producibility.

The P2764A provides access times to 200 ns (P2764A-2) which is an improvement over the fastest P2764 access time of 250 ns. This is compatible with high-performance microprocessors, such as Intel's 8 MHz iAPX 186 allowing full speed operation without the addition of WAIT states. The P2764A is also directly compatible with the 12 MHz 8051 family.

Several advanced features have been designed into the P2764A that allow fast and reliable programming — the int$_e$ligent Programming Algorithm and the int$_e$ligent Identifier Mode. Programming equipment that takes advantage of these innovations will electronically identify the P2764A and then rapidly program it using an efficient programming method.

The P2764A also offers reduced power consumption compared to the P2764. The maximum active current on faster speed parts is 60 mA while the maximum standby current is only 20 mA. The standby mode lowers power consumption without increasing access time.

Two-line control and JEDEC-approved, 28 pin packaging are standard features of all Intel higher density EPROMs. This ensures easy microprocessor interfacing and minimum design efforts when upgrading, adding or choosing between non-volatile memory alternatives.

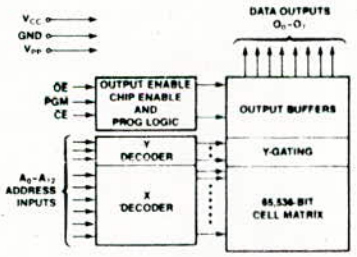
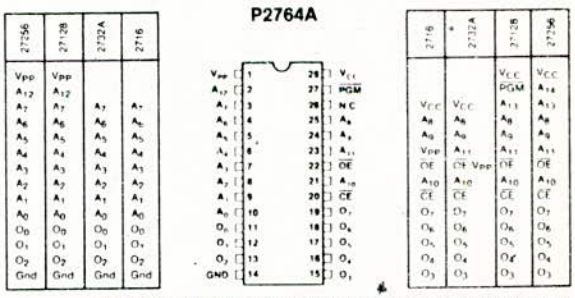


Figure 1. Block Diagram



NOTE: INTEL "UNIVERSAL SITE"-COMPATIBLE EPROM PIN CONFIGURATIONS ARE SHOWN IN THE BLOCKS ADJACENT TO THE 2764A PINS

Figure 2. Pin Configurations

MODE SELECTION

MODE \ PINS	CE (20)	OE (22)	PGM (27)	A_9 (24)	V_{PP} (1)	V_{CC} (28)	Outputs (11-13, 15-19)
Read	V_{IL}	V_{IL}	V_{IH}	X	V_{CC}	V_{CC}	D_{OUT}
Output Disable	V_{IL}	V_{IH}	V_{IH}	X	V_{CC}	V_{CC}	High Z
Standby	V_{IH}	X	X	X	V_{CC}	V_{CC}	High Z
Verify	V_{IL}	V_{IL}	V_{IH}	X	V_{PP}	V_{CC}	D_{OUT}
Program Inhibit	V_{IH}	X	X	X	V_{PP}	V_{CC}	High Z
int$_e$ligent Identifier	V_{IL}	V_{IL}	V_{IH}	V_H	V_{CC}	V_{CC}	Code
int$_e$ligent Programming	V_{IL}	V_{IH}	V_{IL}	X	V_{PP}	V_{CC}	D_{IN}

1. X can be V_{IH} or V_{IL}
2. V_H = 12.0V ± 0.5V

PIN NAMES

A_0-A_{12}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O_0-O_7	OUTPUTS
PGM	PROGRAM
N.C.	NO CONNECT

*HMOS is a patented process of Intel Corporation

MARCH 1984
ORDER NUMBER: 230864-002

MOTOROLA

MC14411

CMOS LSI

(LOW POWER COMPLEMENTARY MOS)

BIT RATE GENERATOR

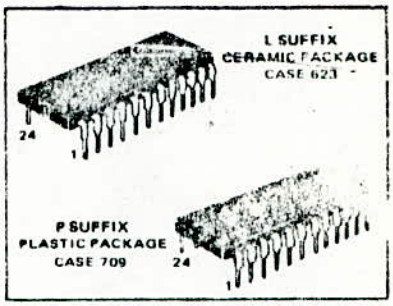


FIGURE 1 -- DYNAMIC SIGNAL WAVEFORMS

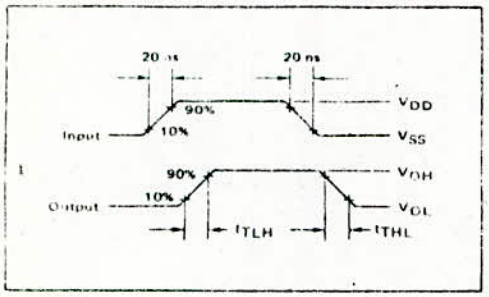
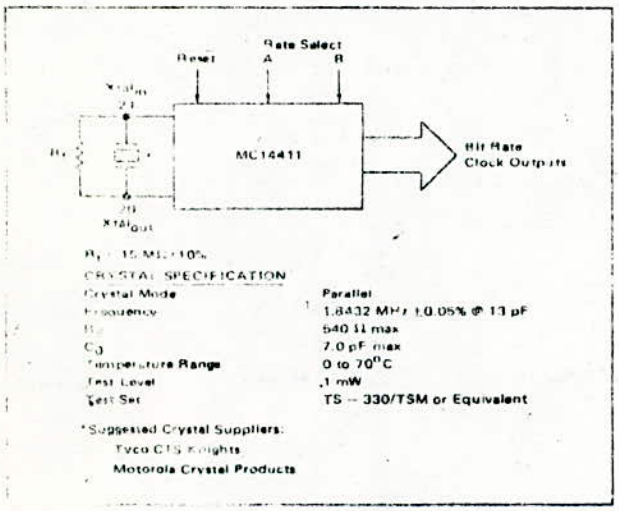


FIGURE 2 -- TYPICAL CRYSTAL OSCILLATOR CIRCUIT



R_S = 15 MΩ ±10%

CRYSTAL SPECIFICATION

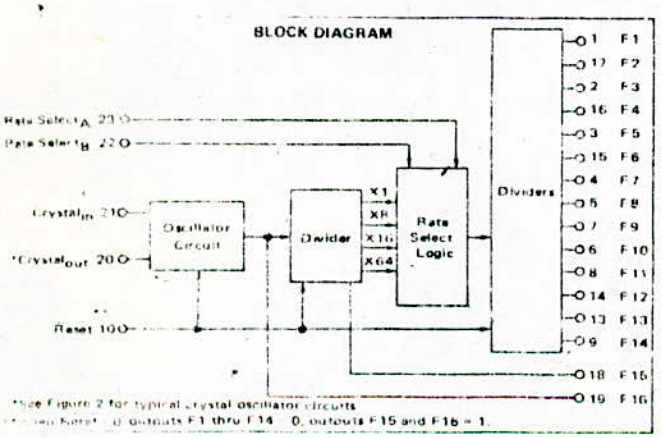
Crystal Mode	Parallel
Frequency	1.8432 MHz ±0.05% @ 13 pF
R_S	540 Ω max
C_0	7.0 pF max
Temperature Range	0 to 70°C
Test Level	1 mW
Test Set	TS -- 330/TSM or Equivalent

*Suggested Crystal Suppliers:

Tyco CTS Knights

Motorola Crystal Products

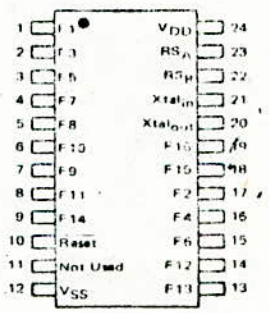
BLOCK DIAGRAM



*See Figure 2 for typical crystal oscillator circuits.

**Rate Select = 0 outputs F1 thru F14 = 0, outputs F15 and F16 = 1.

PIN ASSIGNMENT



V_{DD} = Pin 24

V_{SS} = Pin 12

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{DD}$.

Unused inputs must always be tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

INTRODUCTORY

SPECIFICATIONS BASED ON EVALUATION OF LIMITED NUMBER OF DEVICES

MAXIM

+5V Powered RS-232 Drivers/Receivers

MAX230-239*

General Description

The MAX230 family of line drivers/receivers is intended for all RS-232 and V.28/V.24 communications interfaces, and in particular, for those applications where ±12V is not available. The MAX230 and MAX238 are particularly useful in battery powered systems since their low power shutdown mode reduces power dissipation to less than 5µW. The MAX233 and MAX235 use no external components and are recommended for applications where printed circuit board space is critical.

All of the MAX230 family except the MAX231 and MAX239 need only a single +5V supply for operation. The MAX230 family RS-232 driver/receivers have on-board charge pump voltage converters which convert the +5V input power to the ±10V needed to generate the RS-232 output levels. The MAX231 and MAX239, designed to operate from +5V and +12V, contain a +12V to -12V charge pump voltage converter.

Since nearly all RS-232 applications need both line drivers and receivers, the MAX230 family includes both receivers and drivers in one package. Since the wide variety of RS-232 applications require differing numbers of transmitters and receivers, Maxim offers a wide selection of RS-232 driver/receiver combinations in order to minimize the package count (see table below).

Both the receivers and the line drivers (transmitters) meet all EIA RS-232C and CCITT V.28 specifications.

Features

- Operates from Single 5V Power Supply (+5V and +12V — MAX231 and MAX239)
- Meets All RS-232C and V.28 Specifications
- Multiple Drivers and Receivers
- Onboard DC-DC Converters
- ±9V Output Swing with +5V Supply
- Low Power Shutdown — <1 µA
- 3-State TTL/CMOS Receiver Outputs
- ±30V Receiver Input Levels

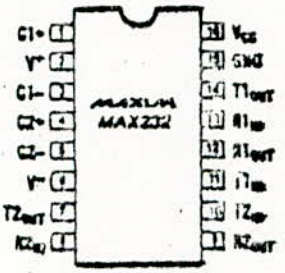
Applications

Computers
Peripherals
Modems
Printers
Instruments

Selection Table

Part Number	Power Supply Voltage	No. of RS-232 Drivers	No. of RS-232 Receivers	External Components	Low Power Shutdown and TTL 3-State	No. of Pins
MAX230	+5V	5	0	4 capacitors	Yes	20
MAX231	+5V and +7.5V to 13.2V	2	2	2 capacitors	No	14
MAX232	+5V	2	2	4 capacitors	No	16
MAX233	+5V	2	2	None	No	20
MAX234	+5V	4	0	4 capacitors	No	16
MAX235	+5V	5	5	None	Yes	24
MAX236	+5V	4	3	4 capacitors	Yes	24
MAX237	+5V	5	3	4 capacitors	No	24
MAX238	+5V	4	4	4 capacitors	No	24
MAX239	+5V and +7.5V to 13.2V	3	5	2 capacitors	Yes	24

*Patent Pending



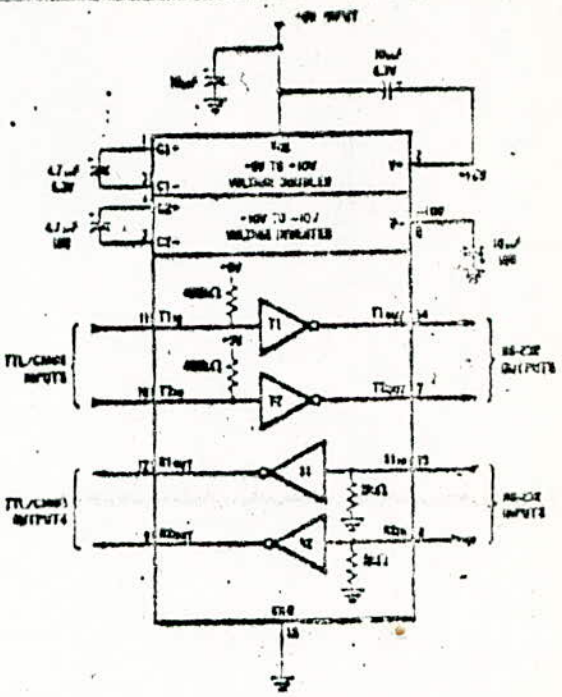


Figure 5. MAX232 Typical Operating Circuit

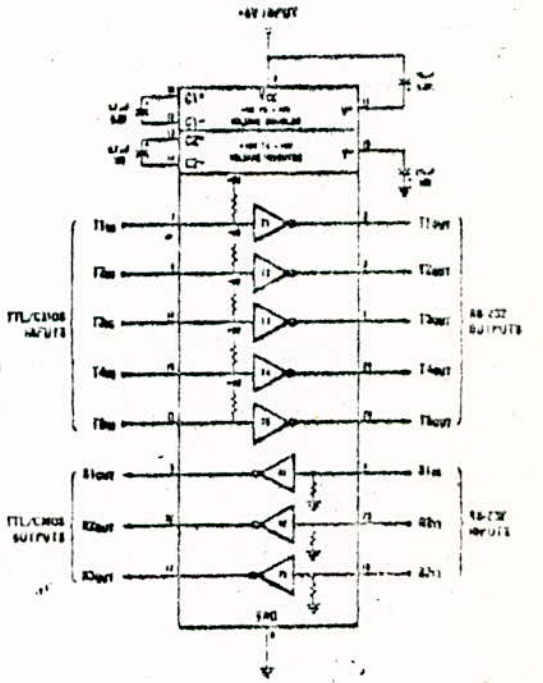


Figure 10. MAX237 Typical Operating Circuit