الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم والبحث العلمي
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT : ELECTRONIQUE

45/84

PROJET DE FIN D'ETUDES

SUJET

Contribution à la Réalisation d'une Unité de Traitement Rapide

Proposé par : Mr Abdellaoui A.

Etudié par : Med Mihoubi
Med seghir Kahla

Dirigé par :

Promotion Janvier 1984

PROMOTION : Janvier 84

بسم الله الرحمن الرحيم

الله نور السموات والارض مثل نوره كمشكاة
فيها مصباح المصباح في زجاجة الزجاجة
كأنها كوكب دري يوقد من شجرة مباركة
زيتونة لا شرقية ولا غربية يكاد زيتها يضيء
ولو لم تمسسه نار نور على نور يهدي
الله لنوره من يشاء ويضرب الله الامثال
للناس والله بكل شيء عليم

(سورة النور)

DDD E D I C A C E S

/J-/ la mémoire de mon Père.

/J-/ ma Mère.

/J-/ mes Frères et Soeurs.

/J-/ mon Oncle SALAH.

/J-/ tous mes Amis.

MOHAMED MIHOUBI.

/J-/ mon Père.

/J-/ ma Mère.

/J-/ mes Frères et Soeurs.

/J-/ toute ma Famille.

/J-/ tous mes Amis.

MOHAMED SEGHIR KAHLA

REMERCIEMENTS

NOUS tenons à remercier Monsieur ABDELLAOUI directeur du C.D.T.A qui a bien voulu nous accueillir dans son centre et qui nous a guidé pendant la réalisation de ce projet de fin d'étude.
Nos remerciements vont également à Monsieur BENSALAH qui n'a pas hésité à nous aider tout au long de notre travail.
Nous remerçions également Monsieur MERZOUK BELKACEMI et Monsieur MAHDI pour leur aide.
Nos vifs remerçiements à Monsieur BENDIFELLAH et à toute l'équipe du C.D.T.A de leur aide morale.
Nous tenons à remerçier tous ceux qui ont contribués de prés ou de loin à l'élaboration de ce travail, ainsi que ceux qui ont assuré notre formation, trouvent en ces lignes l'expression de notre profonde reconnaissance.

M. MIHOUBI

M.S. KAHLA

SOMMAIRE.

CHAPITRE IV : Etude théorique de la transformée de Fourier Discrète

I- Généralités

II- Etude théorique

- étude de la D.F.T
- étude de la F.F.T

III- Etude comparative

IV- Conclusion

INTRODUCTION

Le microprocesseur peut etre relié par des circuits d'interface appropriés ,à une grande varieté de peripherique

Il apparait toutefois que le developpement de systeme microprocesseur a été entrave par une inssuffisance tres nette dans le domaine du traitement arithmetique

Dans le cadre du projet "ESR" du CDTA a adopté pour réaliser une machine autoumome à base d'un microprocesseur connecté à une unité arithmetique rapide.

Cette machine sera realisée pour une question de porta_ _bilite vu que lorsqu'on travaille sur le champ il est impossible de transporté une grosse machine.

Dans ce sens il nous a été confie l'étude d'une unité arithmetique rapide l'AM9511 et son couplage au micro--processeur le MC 6809 ainsi que la realisation d'une carte memoire RAMet ROM

VU l'indisponibilite du systeme de developpement du 6809 Dont ilaété question initialement nous avions eu recours au microprocesseur MC6800 afin de pouvoir tester le bon fonctii--onnement de cette carte.

A titre de simulation une étude théorique des transformées de Fourier a été menée en vue de son éventuelle application dans les divers domaines du traitement numérique. Dans ce sens les propriétés de la D.F.T (transformée de Fourier Discrête) et de la F.F.T (Fast Fourier Transform) ont été dégagées,ainsi que quelques algorithmes.

CHAPITRE I: ETUDE DE L'UNITE ARITHMETIQUE 9511 ET SIGNAUX DU 6809.

I INTRODUCTION

Cette partie consiste en la réalisation d'une carte CPU organisée autour d'un microprocesseur du type MC 6809.

Le choix de ce composant est lié au fait qu'il présente les avantages (par rapport à ces antécedents de Motorola) de travailler sur des mots de 16 bits.

Le microprocesseur en matière de calcul est compensée par l'emploi d'un processeur arithmétique AM 9511. L'AM intervient surtout pour les types d'opérations qui n'existent pas dans les microprocesseurs. Il semble alors nécessaire d'étudier l'U.A.R. L'A.M.9511 et les signaux du M.PU.6809.

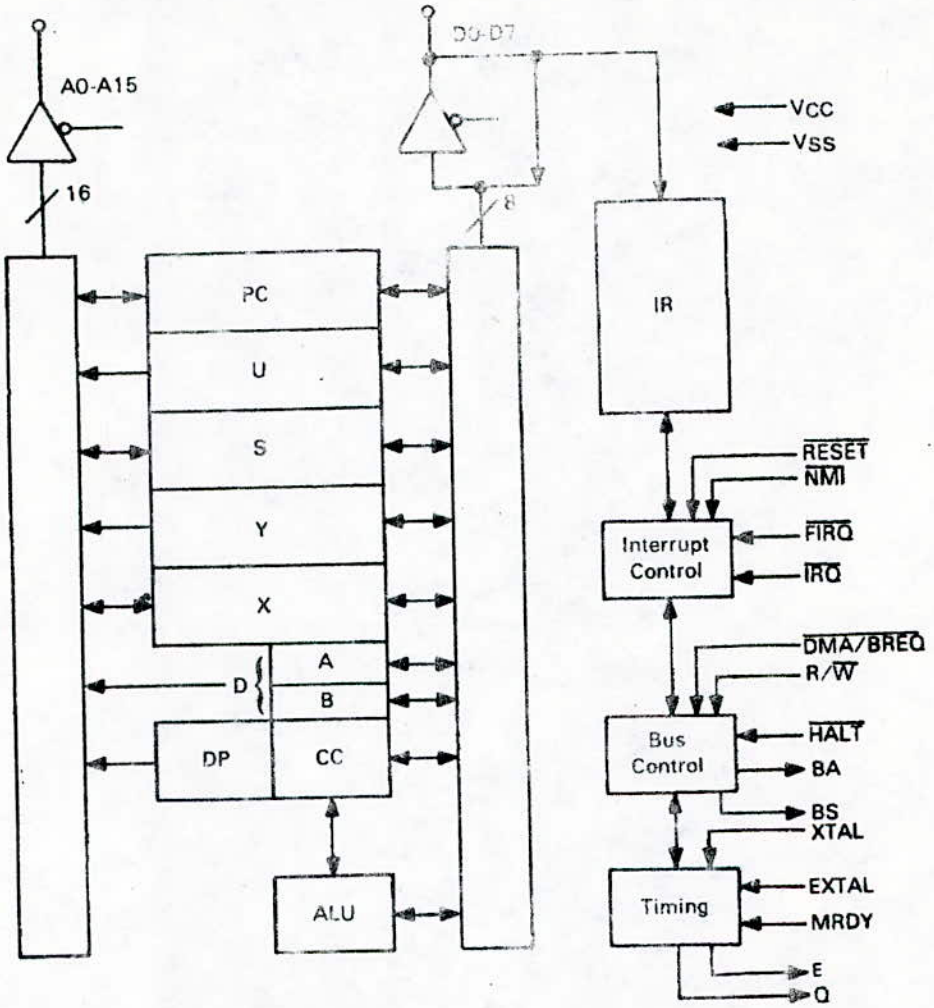
I_1. ORGANISATION INTERNE DU M.P.U.

C'est un microprocesseur monolitique de 8 bits (faux 16 bits) comporte 9 registres programmables.

- Deux registres d'index X et Y (16 bits)
- Deux accumulateurs 8 bits (transformables en un accumulateur D 16 bits).
- Deux registres pointeur pile U et S (16 bits).
- Un compteur programme PC (16 bits).
- Un registre de page D.P. (8 bits).
- Un registre code condition C.C.R. (8 bits).

Voir figure 3.

FIGURE 3 – EF6809 EXPANDED BLOCK DIAGRAM



I_2. SOMMAIRE DES SIGNAUX DU M.P.U.

Les broches d'entrées et de sortie du M.P.U. 6809 peuvent se regrouper en quatre parties:

1. Le bus adresses : Ao - A_{15}.
2. Le bus données : Do - D_7
3. Le bus de contrôle et de commande.

a) Signaux de contrôle

- Lecture écriture R/$\bar{w}$w Ce signal détermine la direction de transfert sur le bus de donnée R/$\bar{w}$ = 1 le microprocesseur est en lecture, Do = D_7 sont des entrées. R/$\bar{w}$ = 0, le processeur est en écriture Do = D_7 sont des sorties.

- Les signaux BA (Bus avaible), BS (bus state) Le sigaal BA indique la présence d'un signal de contrôle interne qui met les bus du microprocesseur dans l'état off

$$BA = 1 \longrightarrow \begin{cases} Ao\text{-}A_{15} \\ Do\text{-}D7 \\ R/w \end{cases} \Longrightarrow H.I.$$

L'état du microprocesseur est reconnu par la combinaison des signaux BA et BS.

BA	BS	Fonctionnement du M/P/U/
0	0	Fonctionnement normal gère le bus (Add) et (Data
0	1	Reconnaissance d'interruption
1	0	Reconnaissance de synchro externe
1	1	Arrêt ou bus accordé

b) Interruptions

- Arrêt du microprocesseur Halt. Cette entrée permet d'interropre le déroulement d'un programme de façon (Hardware) à la fin de l'instruction en cours, sans perdre les données Bus add et data passent à l'état (HI), ce qui est indiqué par l'état haut du signal BA ainsi que BS.

- Initialisation : $\overline{\text{Reset}}$. Un niveau bas sur cette entrée entraine une réinitialisation complète du micro.

Les vecteurs d'initialisation sont accéssibles aux adresses FFFE - FFFF, le processeur éxécute le programme à partir de cette adresse.

Le 6809 possede les 2 lignes d'intérruption du 6800 c'est à dire $\overline{\text{IRQ}}$ et $\overline{\text{NMI}}$, plus une ligne d'intérruption rapide $\overline{\text{FIRQ}}$.

- $\overline{\text{NMI}}$: A l'état bas, cette entrée est active après une séquence d'initialisation cette broche est validée par un changement du pointeur de pile. Le procésseur exécute un programme de traitement d'intérruption approprié. Cette intérruption $\overline{\text{NMI}}$ est la plus prioritaire des interruptions.

- $\overline{\text{FIRQ}}$: A l'etat bas cette entrée est active. Cette interruption rapide ne sauvegarde que le compteur programme et registre d'état, ce qui fait 3 octets au lieu de 12 d'où un gain de neuf cycles processeur.

Un niveau "0" sur $\overline{\text{FIRQ}}$ provoque alors l'éxecution de la séquence suivante .

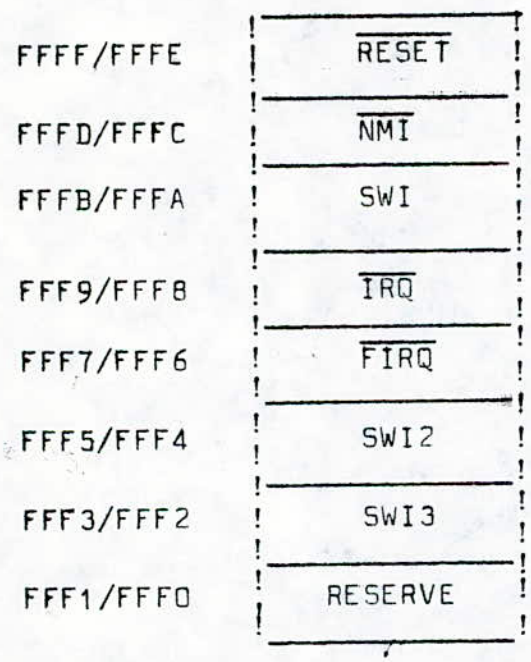
- Sauvegarde partielle du contexte dans la pile système.
- PC et CCR sont sauvegardés dans la pile à partir de SP-1

Dans ce cas cette ligne doit être remise à l'état initial pendant l'exécution du sous-programme d'intérruption.

- $\overline{IRQ}$: Un niveau "0" sur IRQ provoque alors l'exécution de la séquence suivante.

- Sauvegarde du contexte de pile .
- Tous les registres internes sont sauvegardés :
  SP' = SP-12.

Le processeur recherche les vecteurs d'intérruptions BA = 0 et BS = 1, le contenu FFF8 et FFF9 est pris en compte par le microprocesseur.
BS = 0, le fonctionnement redevient normal, le microprocesseur exécute le programme de traitement $\overline{IRQ}$



- Entrées horloge: XTAL et EXTAL

Deux modes de fonctionnement sont possibles.

- Les broches xtal et EXTAL sont resérvées pour l'emplacement d'un quartz externe. La broche EXTAL peut-être pilotée extérieurement par un signal d'entrée TTL si une horloge externe est necéssaire .Dans ce cas la broche XTAL sera mise à zéro.

Le quartz ou la fréquence externe est 4 fois la fréquence bus.

* Sorties horloge E et Q

- E out: signal horloge du système dont la fréquence est celle de base du microprocesseur.

- Q out : signal horloge en quadrature (I/2) avec E.

Les adresses sont validées à partir d'un front montant de Q. Les données sont mémorisées sur un front descendant de E.

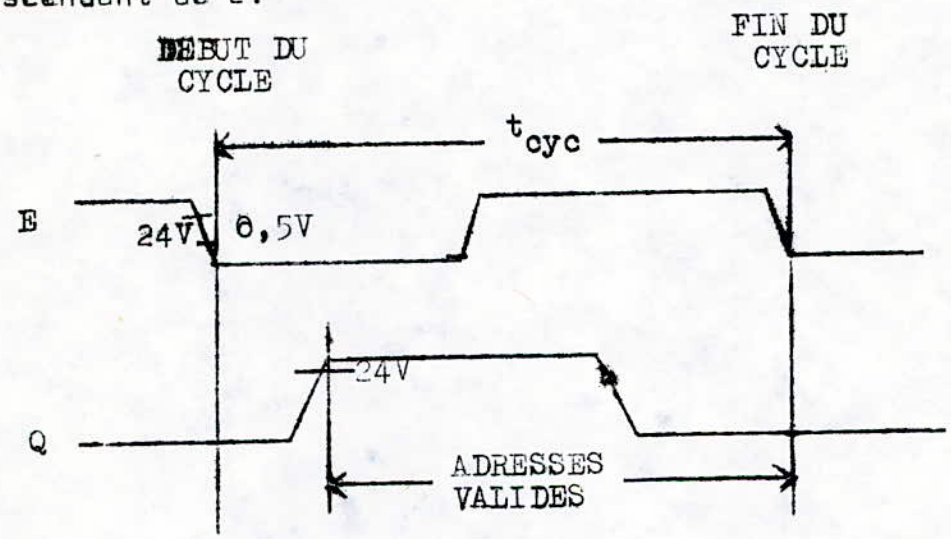


FIG : DIAGRAMMES DES TEMPS DE E et Q

* DMA et rafraichissement mémoire : $\overline{\text{DMA/BREQ}}$

Pour faire l'accès direct ou un raffraichissement mémoire, l'entrée $\overline{\text{DMA/BREQ}}$ permet de suspendre l'utilisation des bus par le microprocesseur.

Lorsque Q est au niveau haut, le passage à l'état bas de $\overline{\text{DMA/BREQ}}$ entraine l'arrêt du programme à la fin de l'instruction en cours BA et BS = 1 les bus data, adresses et R/W sont à l'etat off (haute impédance) , l'horloge externe est bloquée, E et Q continuent à osciller .

A) L'ALIMENTATION

Le M.P.U. 6809 ne nécessite qu'une seule tension d'alimentation Vcc = +5v. D'où son avantage d'être compatible avec les circuits intégrés TTL.

I_3. LE LOGICIEL

Le 6809 dispose de 10 modes d'adressage. Il posséde 59 instructions de base, et il admet 1464 possibilités différentes d'instructions et de modes d'adressage .

Les nouveaux modes d'adressage apparus sur le 6809 et inéxistants sur les antécedants permettent les techniques de programmation modernes. Les modes d'adressage suivants sont disponibles dans le 6809.

$I_{3.1}$ Jeux d'instruction (Voir annexe)

L E L O G I C I E L

Modes d'adressage:

Le 6809 dispose de 10 modes d'adressage. Il possède 59 instructions de base et il admet 1.464 possibilités différentes d'instructions et de modes d'adressage. Les nouveaux modes d'adressage apparus sur le 6809 et inéxistants sur ses antécédants, permettent les techniques de programmation modernes.

Les modes d'adressage suivants sont disponibles dans dans le 6809.

Mode d'adressage et forme de l'instruction	Explication et exemples Ø = symbole notation héxadécimale EA = adressage effective.
Adressage inhérent	Dans ce mode d'adressage, l'instruction contient toute l'information adresse nécéssaire Ex. ASRA, DAA, ABX, CLRB
Adressage immédia LDA ## DATA	Le symbole ## précède l'opérande puisqu'il n'ya pas non plus d'adresse. Les données à utiliser dans l'instruction du 6809 utilise 2 valeurs immédiates 8 et 16 bits selon la taille de l'opérande spécifiée par le code opération Ex. LDA ## 08 , LDX ## Ø 2000
Adressage direct LDA addr	La partie adresse de l'instruction contient l'octet poids faible de l'adresse sur 16 bits, l'octet poids fort est fourni par le registre de page directe. Ce mode d'adressage nécéssite moins de mémoires et s'exécute plus rapidement qu'en adressage étendu. Le registre DP est mis à Ø00 à l'initialisation. EX. LDA Ø CO si (DP) = Ø 80 EA=Ø 80 CO

Adressage étendu LDA addr	En adressage étendu, le contenu des deux octets suivant immédiatement le code opération spécifie complètement l'adresse 16 bits effective utiliséé par l'instruction Ex. LDA $2000 EA = $2000
Adressage par registre	Certains codes opération sont suivis par un octet qui définit un registre ou un jeu de registres devant être utilisés par l'instruction cet octet est appelé post octet Ex. T F R X Y transfert de X dans Y PSHS A,B,X,Y transfert dans S,Y,X,B puis A
Adressage indéxé direct LDAN,R R=XXYS ou U	Le déplacement N peut-être : * Soit un nombre d signé de 5, 8 ou 16 bits, si d est nul la forme de l'instruction est LDA, R. l'adresse effective est EA = (R) ± d EX. LDA $ 20, X Si (X)=$1000 on a EA= $ 1020. * Soit le contenu signé de l'un des accus A,B ou D on a EA = (R) ± (r) si r=A,B ou D. Ex. LDA B,Y si (y)= $ 3000 et (B) =7 on a EA= $ 3007 si (B)=-7 soit F9. on a EA= $FFF9 = $2FF9.
Adressage indéxé indirect LDA (N,R) R=X,Y,S ou U	Ce mode est identique au précedent à l'exception prés de l'adressage indirect. Dans ce cas l'adresse esr donnée par les contenus des positions mémoires d'adresse EA et EA+1 si EA est calculé conformément à l'adressage indéxé direct ci-dessus. Si n est nul, la forme de l'instruction est LDA (0,R) EX; LDA ($ 20,X) si (X) = $1000 et si de plus (1020) = $C0 et (1021) = $FE ⟹ EA= $C0FE

II A PRESENTATION DE L'UNITE ARITHMETIQUE AM 9511

II A_1 INTRODUCTION

Les microprocesseurs sont peu performants en ce qui concerne les calculs arithmétiques. Un circuit de calcul arithmétique très performant appelé AM 9511, se connectant facilement à un microprocesseur . Le circuit effectue les opérations arithmétiques en virgule fixe et en virgule flottante, ainsi que bien d'autres fonctions trigonométriques et mathématiques.

II A_2 CARACTERISTIQUES GENERALES DE L'UNITE AM 9511

L'AM 9511 est monochip LSI, en téchnologie NMOS, qui se présente d'élargir les variétés du système processeur Il travaille à une fréquence maximum d'horloge de 2 MHz et elle est alimentée par +5v et +12v.

II A_3 SIGNAUX AM 9511

* DB_0-DB_7 : ceux sont les 8 bits du bus donnée. Ils permettent l'envoi d'un mot de commande et des opérandes à l'UAR 9511 ainsi que la lecture du mot d'etat et du résultat d'une opération. Notons que les transfert de données avec l'AM 9511 peuvent se faire soit en mode programmé, soit par accès direct mémoire.

* CLK (clock): c'est l'entrée horloge.

* RESET : c'est la commande d'initialisation de l'UAR

* $\overline{CS}$: Cette entrée validation conditionne les commandes $\overline{RD}$ et $\overline{WR}$ et autorise la communication avec le bus donnée.

* $\overline{RD}$: C'est la commande de lecture active par son niveau zéro lorsque $\overline{CS}$ = 0

* $\overline{WR}$: C'est la commande d'écriture active par son niveau zéro lorsque $\overline{CS}$ = 0.

* C/$\overline{D}$ (commande/Data) : cette entrée est un bit de selection de registre utilisé en conjonction avec $\overline{RD}$ et $\overline{WR}$ Le C/$\overline{D}$ d'entrée indique le type de transfert à exécuter sur le bus de donnée.

TRANSFERT µp AM 9511

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Opération réalisée
0	1	0	Ecriture d'un octet donnée dans la pile 9511
0	0	1	Lecture d'un octet donnée dans la pile 9511
1	1	0	Ecriture d'un mot de commande
1	0	1	Lecture d'un mot d'etat

A_0 = 0 transfert d'un octet de donnée

A_0 = 1 transfert d'un octet de commande

le bit A_0 ⟷ C/$\overline{D}$

* $\overline{PAUSE}$: Cette sortie est un bit qui permet un échange asynchrone de données entre le microprocesseur et l'AM 9511. Cette sortie prend la valeur "0" tant que le transfert n'est pas terminé.

* $\overline{END}$:(END execution): cette sortie; active par son niveau zéro, indique la fin d'exécution de la dernière opération programmée. Ce signal peut-être utilisé pour générer une demande d'interruption, il est remis à zéro par un reset ou par $\overline{EACK}$.

* $\overline{\text{EACK}}$ (END ACK NOWLEDGE) : Cette commande d'entrée remet à zéro la sortie END.

* SVREQ (service request): Cette sortie signal par son niveau "1" indique la fin d'exécution de la dernière opération commandée, lorsqu'une demande de service à été programmée dans le mot de commande SVREQ est remis à zéro par un reset ou par $\overline{\text{SVACK}}$.

* $\overline{\text{SVACK}}$: Cette entrée remet à zéro SVREQ.

II A_4 DESCRIPTION FONCTIONNELLE

Le synoptique du composant est donné par la figure

La structure interne est celle d'un processeur 16 bits parallèles, sur ce bus interne de 16 bits sont connéctés:

- L'unité arithmétique logique (ALU) qui reçoit des opérandes de la pile opérande.
- Une pile où sont stockés les opérandes et les résultats de calcul. cette pile est organisée en 8 mots de 16 bits, avec la dernière entrée, première sortie (LIF)
- Une mémoire ROM de 128
- Les registres de travail au nombre de 10, utilisés pour le stockage des valeurs intérmédiaires pendant l'exécution des opérations.

Les opérations de l'AM 9511 sont contrôlées par le programme contenu dans une mémoire control ROM. le programme counter fournit les adresses des instructions et chargé partiellement par le registre de commande.

Les adresses de retour, lors d'un appel de soubroutine sont stockées dans le programme counter. Le chargement du registre de commande fournit l'adresse de départ dans la mémoire du programme.

II $A_{4.1}$ FONCTIONNEMENT DE LA PILE DE DONNEES

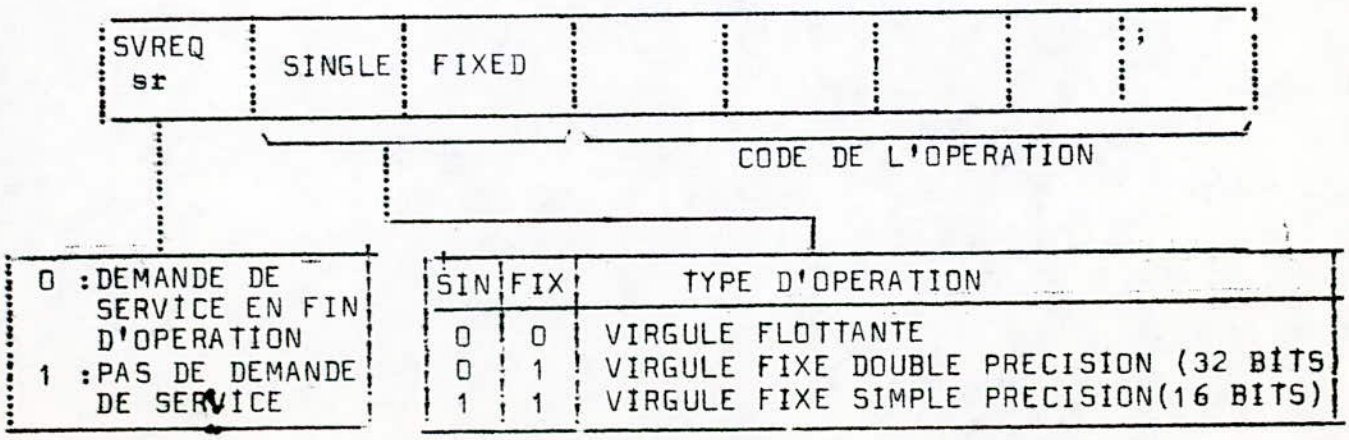
Les opérandes et les résultats des calculs sont rangés dans une pile LIFO organisée en 4 niveaux de 32 bits ou en 16 bits de 8 niveaux en fonction du format de travail.

Les opérandes sont rangées dans la pile, avant le début d'une opération. L'octet de poids faible LS Byte en tête. L'opération elle même s'effectue sur les opérandes situées au sommet de la pile TOS (Topof Stack) et à l'emplacement suivant NOS (Next Of Stack). A la fin de l'opération, le résultat se retrouve au sommet de la pile après une éventuelle opération de rotation (Pop Stack) implicitement contenu dans le code opération.

L'acquisition du résultat par le microprocesseur s'effectue par la lecture de la pile octet de poids fort en tête suivant le format choisi.

II A_{42} REGISTRE DE COMMANDE

Le registre de commande spécifiant le type de traitement à effectuer. Chaque chaque commande exécutée par l'APU est spécifiée par un simple byte avec le format de la figure

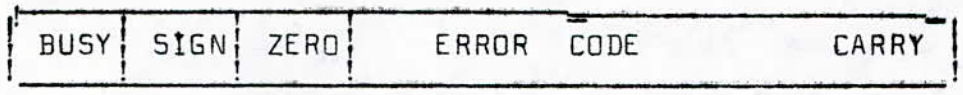


Push stack : indique une opération d'enfoncement dans la pile (avec perte de l'opérande situé à la base de la pile)

Pop stack : indique une opération d'extraction de la pile

II A_{43} REGISTRE D'ETAT

Le registre d'etat de l'AM 9511 est montré Quand le bit busy (bit 7) est à "1" l'APU est procédé auparavant d'entré commande du registre d'état qui ne sera pas considéré valide. Quand busy est à zéro, l'opération est achevée et les bits d'autres états sont validés.



REGISTRE D'ETAT

* Busy (bit 7) est zéro, indique que l'opération en cours est achevé et une autre opération peut être initiée. Notons bien qu'on peut lire le registre d'état pendant l'exécution d'une commande .

Si bit 7= 1, il indique q'une commande est en cours d'exécution.

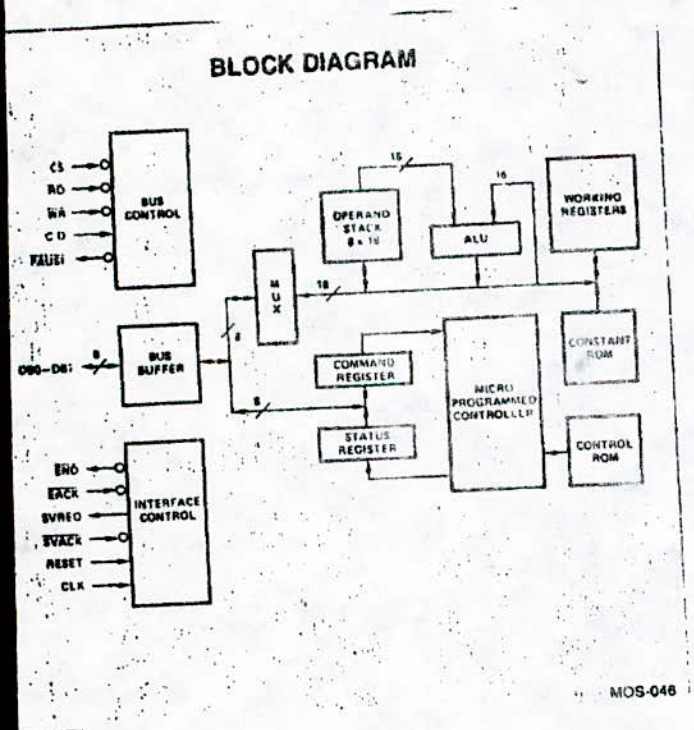
* Signe (bit6) Ce bit indique que le sommet de pile est < 0 si bit 6= 1.

* Zéro (bit 5) : Ce bit indique la valeur du sommet de la pile est nulle.

* Carry "1" présence de carry/borrow
"0" sinon.

* CODE ERREUR

0000	PAS D'ERREUR
I000	DIVISION PAR ZERO
0I00	OU LOG D'UN NOMBRE INFERIEUR A ZERO
II00	ARGUMENT DE I/SIN I/COS OU e^X TROP GRAND
xx0I	OVERFLOW
xxI0	UNDERFLOW



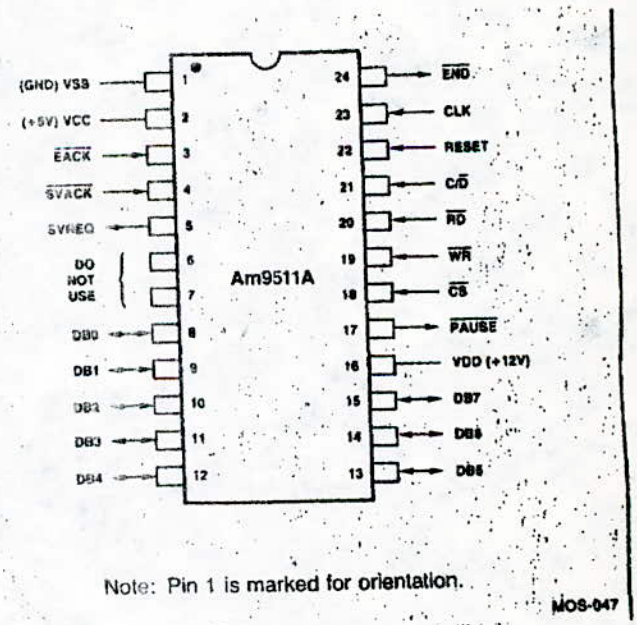


Table 1.

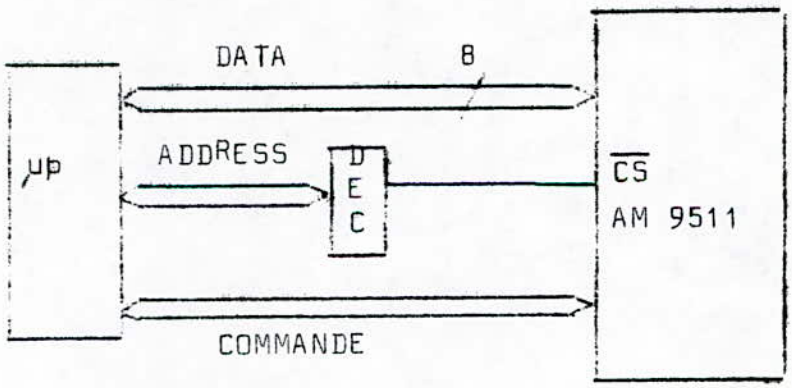
ommand hemonic	Hex Code (sr = 1)	Hex Code (sr = 0)	Execution Cycles	Summary Description
			16-BIT FIXED-POINT OPERATIONS	
SADD	EC	6C	16-18	Add TOS to NOS. Result to NOS. Pop Stack.
SSUB	ED	6D	30-32	Subtract TOS from NOS. Result to NOS. Pop Stack.
SMUL	EE	6E	84-94	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
SMUU	F6	76	80-98	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
SDIV	EF	6F	84-94	Divide NOS by TOS. Result to NOS. Pop Stack.
			32-BIT FIXED-POINT OPERATIONS	
DADD	AC	2C	20-22	Add TOS to NOS. Result to NOS. Pop Stack.
DSUB	AD	2D	38-40	Subtract TOS from NOS. Result to NOS. Pop Stack.
DMUL	AE	2E	194-210	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
DMUU	B6	36	182-218	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
DDIV	AF	2F	196-210	Divide NOS by TOS. Result to NOS. Pop Stack.
			32-BIT FLOATING-POINT PRIMARY OPERATIONS	
FADD	90	10	54-368	Add TOS to NOS. Result to NOS. Pop Stack.
FSUB	91	11	70-370	Subtract TOS from NOS. Result to NOS. Pop Stack.
FMUL	92	12	146-168	Multiply NOS by TOS. Result to NOS. Pop Stack.
FDIV	93	13	154-184	Divide NOS by TOS. Result to NOS. Pop Stack.
			32-BIT FLOATING-POINT DERIVED OPERATIONS	
SQRT	81	01	782-870	Square Root of TOS. Result to TOS.
SIN	82	02	3796-4808	Sine of TOS. Result to TOS.
COS	83	03	3840-4878	Cosine of TOS. Result to TOS.
TAN	84	04	4894-5886	Tangent of TOS. Result to TOS.
ASIN	85	05	6230-7938	Inverse Sine of TOS. Result to TOS.
ACOS	86	06	6304-8284	Inverse Cosine of TOS. Result to TOS.
ATAN	87	07	4992-6536	Inverse Tangent of TOS. Result to TOS.
LOG	88	08	4474-7132	Common Logarithm of TOS. Result to TOS.
LN	89	09	4298-6956	Natural Logarithm of TOS. Result to TOS.
EXP	8A	0A	3794-4878	e raised to power in TOS. Result to TOS.
PWR	8B	0B	8290-12032	NOS raised to power in TOS. Result to NOS. Pop Stack.
			DATA AND STACK MANIPULATION OPERATIONS	
NOP	80	00	4	No Operation. Clear or set SVREQ.
FIXS	9F	1F	90-214	Convert TOS from floating point format to fixed point format.
FIXD	9E	1E	90-336	
FLTS	9D	1D	62-156	Convert TOS from fixed point format to floating point format.
FLTD	9C	1C	56-342	
CHSS	F4	74	22-24	Change sign of fixed point operand on TOS.
CHSD	B4	34	26-28	
CHSF	95	15	16-20	Change sign of floating point operand on TOS.
DUPS	F7	77	16	Push stack. Duplicate NOS in TOS.
DUPD	B7	37	20	
DUPF	97	17	20	
POPS	F8	78	10	Pop stack. Old NOS becomes new TOS. Old TOS rotates to bottom.
POPD	B8	38	12	
POPF	98	18	12	
EXCHS	F9	79	18	Exchange TOS and NOS.
EXCHD	B9	39	26	
EXCHF	99	19	26	
PUPI	9A	1A	16	Push floating point constant π onto TOS. Previous TOS becomes NOS.

CHAPITRE II

COUPLAGE DE L'UNITE DE TRAITEMENT
AM 9511 AU MICROPROCESSEUR 6809.

I INTRODUCTION

La connexion de l'unite arithmétique rapide de AM 9511 est destinée à accroitre la capacité de calculs arithmétiques des microprocésseurs afin d'avoir un système de traitement optimal. Dans notre cas, le couplage se fait avec le MC 6809.



SYNOPTIQUE GENERALE DE LA CARTE C P U

II A STRUCTURE DU SYSTEME

- Présentation de la carte CPU

La carte MPU comporte un bus interne qui permet la liaison entre les divers circuits qui la compose à travers un connecteur de base qui est deposé sur le fond de panier du système.

Les informations de données seront échangées avec les autres modules par un ensemble d'amplificateurs, trois états permettant de relier le bus interne au bus de fond de panier.

II B ORGANISATION DU SYSTEME

La configauration adoptée pour réaliser cette machine, centrée sur le MC 6809 et UAR (AM 9511) dont le schéma synoptique est donné en fig

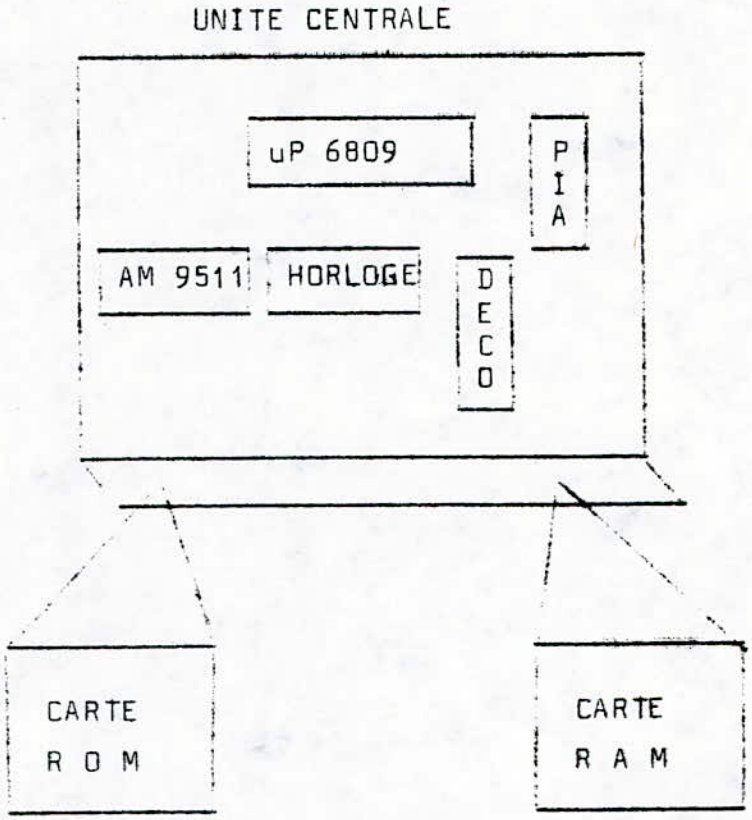


FIG.

Vu la taille du programme nous avons dû réaliser un système modulaire qui se compose d'une carte MPU rapide.

Le module carte RAM qui occupe un espace mémoire de 16 K et d'une carte ROM dont la capacité est de 32 K.

Un espace mémoire pour l'ACIA est prévu pour la réalisation future.

II B_1 CONSTITUTION DU MODULE MPU

La carte MPU est organisée autour du MC 6809 , Ce microprocesseur ne peut rien faire seul, le module MPU doit necéssairement comporter, en plus du microprocesseur, les circuits suivants:

II $B_{1.1}$ Circuits d'interface:

Les circuits d'interface ne servent pas uniquement à fournir de la puissance (amplification, adaptation ...) leur rôle ne se limite pas, non plus, à la protéction du microprocesseur en particulier, ils permettent, et c'est là leur principale caractéristique, d'isoler (de connecter) le bus d'adresse, de données et certaines lignes de commande ou de contrôle.

Ces interfaces sont caractérisées par trois états:

- Un état haut : état haute impédance.
- Deux états bas : état logique "1"
  état logique "0"

II $B_{1.2}$ Interface d'adresses :

Les lignes d'adresses passent par des amplificateurs unidirectionnels non inverseurs et sont appelés buffers

3 états unidirectionnels:

Les buffers utilisés dans le module MPU sont du type MC 8T95 (brochage à l'annexe). Les lignes d'adresses etant au nombre de 16, on aura donc besoin de 3 buffers, car chaque buffer peut prendre 6 lignes de transmission.

II B.$_{11b}$ Interface de données

Les lignes de données aboutissent sur un ampli bidirectionnels . Cet ampli reçoit sur son entrée DIR le signal de commande R/$\overline{w}$ qui lui permet d'indiquer dans quel sens l'information circule.

(µp ⟶ mémoire: écriture, mémoire ⟶ µp lécture

Les lignes de données étant de 8; 2 buffers MC 8T26 sont suffisants.

II B.$_{12}$ LOGIQUE DE COMMANDE ET DE CONTROLE

II B.$_{1.21}$ Une ligne R/$\overline{W}$ (lecture écriture) amplifiée determine s'il faut autoriser un transfert ou une recéption de données suivant qu'il reçoit un ordre de lecture ou d'écriture sur les intérfaces de données.

II B$_{1.21a}$ Opération d'écriture

Cette opération n'a lieu que si R/W = 0 c'est à dire le signal d'écriture est validé. Le bus de données du microprocésseur doit être validé afin de pouvoir transferer les données.

Le signal d'écriture agit sur la commande de validation du sens sortant des MC 8 T26.

$$Se = \overline{R/\overline{w}} \cdot \emptyset_2 \cdot \overline{RG}$$

Table de vérité de 8T26 de la carte CPU

R/$\overline{w}$	Ø	$\overline{RG}$	P1	P_{15}
0	0	0	1	0
0	0	1	1	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	0	0
1	1	1	1	0

II. $B_{1.21b}$ Opération de lecture

Le signal de lecture R/$\overline{w}$ doit être égal à 1 (R/$\overline{w}$=1) Le signal $Ø_2$ est indispensable car les éléments à lire (mémoire, AM 9511, PIA, etc.) ne sont activés que pendant ce temps.

$$S_L = R/W \; Ø_2 \; \overline{RG}$$

Le signal d'activation de l'entrée de donnée dans le cas des 8T26 doit être à l'état bas , donc on doit inverser le signal S_L

$$\overline{S_L} = R/\overline{w} \; Ø_2 \; \overline{RG}$$

II $_{123}$ Circuit d'horloge:

Ce circuit est prévu pour fournir les fréquences $2f_o$ pour l'utilisation de l'UARet $4f_o$ pour le up 6809.

Cette horloge comporte un oscillateur intégré, ainsi qu'un quartz ou un réseau RC déterminant la fréquence de fonctionnement.

La broche d'alimentation du MC 6875 doit être découplée par une capacité de 0,1 uF la reliant directement à la masse .

Les entrées inutilisées doivent être raccordées à Vcc, ou à la masse. Les entrées mémoires prêtes, $\overline{DMA/Ref\ Req}$ et Power on reset doivent être connectées à Vcc si elles ne sont pas utilisées.

Ex int. doit être reliée à la masse si elle n'est pas utilisée.

OSCILLATEUR

Pour maintenir la stabilité de fonctionnement du quartz, il est recommandé de relier un quartz entre X_2 et la masse pour eviter que l'oscillateur ne démarre à une fréquence autre que celle désirée.

La fréquence choisie pour attaquer EXTAL du up est de 4 fois f_o, donc on doit UTILISER un quartz de 4 MHz. Ce générateur d'horloge délivre une fréquence de $4f_o$ et $2f_o$.

Le circuit d'horloge doit être en mesure de generer les signaux d'horloge requis par le système entrée bâti autour du MC 6809 (up et elément de support). Ces signaux se décomposent en :

$4 F_o$: sortie d'un oscillateur fonctionnant à 4 fois la fréquence d'horloge du utilisable pour la synchronistation d'un système.

$2 f_B$: sortie d'un oscillateur fonctionnant à 2 fois la fréquence d'horloge du µp.

Mémoire prête: entrée asynchrone utilisée pour bloquer les phases d'horloge du up dans l'état $Ø_1$ bas, $Ø_2$ haut; pour l'utilisation avec une mémoire lente.

$\overline{\text{Power ou reset}}$: entrée d'uhe bascule (Trigger de Shmitt)

X_1, X_2 : Broches prévues pour connecter un quartz à résonnance série , ou un réseau RC.

EXTIN : Permet de piloter le circuit par un signal extérieur TTL pour assurer la synchronisation du µp et d'un système extérieur.

Circuit de réinisialisation (II 124)

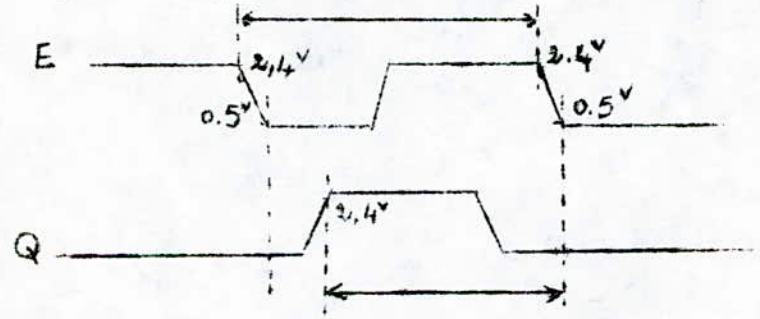
A la mise sous tension, la broche $\overline{\text{Reset}}$ (37) reste à zéro, la capacité se charge lentement; quand la tension de la broche 37 dépasse 4 volts, le up commence sa séquence d'initialisation. La diode permet de décharger rapidement la capacité en cas de coupure de courant ce qui entraine une initialisation correcte lorsque le courant réapparait.

II 1.2.5) CIRCUIT DE REINITIALISATION:

A la mise sous tension ,la broche $\overline{\text{RESET}}$(37) reste à zéro,la capacité se charge lentement;quand la tension de(37) dépasse 4V le µp commence sa séquence d'initialisation.La diode permet de décharger rapidement la capacité,en cas de coupure de courant ce qui entraine une initialisation correcte lorsque le courant réapparait.

II 1.2.5) LE SIGNAL V.M.A:

Le signal VMA est générer à partir de E,Q et BA. Ce signal est nécessaire à la commande des autres circuits . On peut le générer à l'aide de deux portes NOR



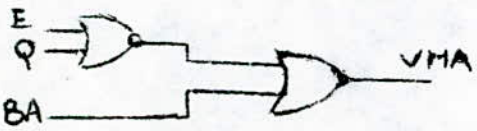


Table de vérité du signal VMA

E	Q	BA	VMA
0	0	X	0
0	1	0	1
1	0	0	1
1	1	0	1
X	X	1	0

Circuit de rafraichissement :

Un circuit de rafraichissement est prévu dans ce système, dans le cas ou l'utilisateur utilise des mémoires dynamiques. Afin de conserver intactes les informations de la memoire vive, il faut régulièrement simuler une lecture cette opération de lecture par une demande de DMA permettant d'accéder directement à la memoire pour effectuer le raffraichissement.

II C DECODAGE D'ADRESSE DES CIRCUITS DE LA CARTE MPU

PRINCIPE: Le décodeur 74 LS 154 va servir à décoder les differents boitiers (AM 9511, PIA, ACIA) dont les 3 lignes d'adresse de poids fort A_{15}, A_{14}, A_{13} sont à l'etat haut pour activer le décodeur en synchronisme avec le signal VMA

L'entrée A du décodeur est reliée à une porte NAND à 4 entrées qui combine les adresses de A_7 à A_4 . L'entrée
C est liée à la ligne A_{12}
D est liée à la ligne A_{11}
B est liée à la ligne A_{10}

II C1. DECODAGE DES CIRCUITS:

Les 3 circuits de la carte CPU (AM, PIA, ACIA) sont différenciés par les lignes d'adresses A_0, A_1, A_2 pour attaquer les $\overline{CS}$ de chaque boitier.

II C2. DECODAGE D'ADRESSE DE L'AM 9511

Le circuit AM 9511 est vu par le microprocesseur comme deux positions mémoire, validée par la sortie du décodeur correspondant à l'adresse EBF (sortie 5) et la combinaison de A_1 A_2 A_3 par une sortie NAND.

Ces deux sorties sont connectées à une porte OR qui validera le $\overline{CS}$ du circuit AM 9511.

Une fois ce circuit selectionné; la ligne Ao reliée au C/$\overline{D}$ (Commande/DATA), indique à l'UAR si le transfert concerne un octet de commande ou un octet de donnée, on choisit comme adresse de cette position memoire EBF8, EBF9 de façon que Ao = 0 ou Ao = 1.

II C2.1 LOGIQUE DE COMMANDE ET DE CONTROLE:

L'UAR (AM 9511) travaille à une fréquence de 2 MHz, on relie la sortie 2Fo de l'horloge MC 6875 directement à l'entrée CLK de l'AM 9511. après avoir inversé la ligne $\overline{Reset}$ du système en l'envoi sur reset de l'AM 9511.

II C2.1.1 CIRCUIT DE LECTURE, ECRITURE.

L'autorisation d'un transfert ou d'une reception de données selon que le circuit reçoit un ordre de lecture ou d'écriture sur les interfaces de données.

II C2.1.1.a OPERATION D'ECRITURE

L'opération d'ecriture n'a lieu que si le signal d'ecriture $\overline{WR}$ est validé, et la commande R/$\overline{W}$ = 0. La présence du signal $\emptyset_2$ est indispensable car les eléments à ecrire ou à lire (memoires) ne sont activés que pendant ce temps. Pour cela un décodeur 74 LS 139 est necessaire.

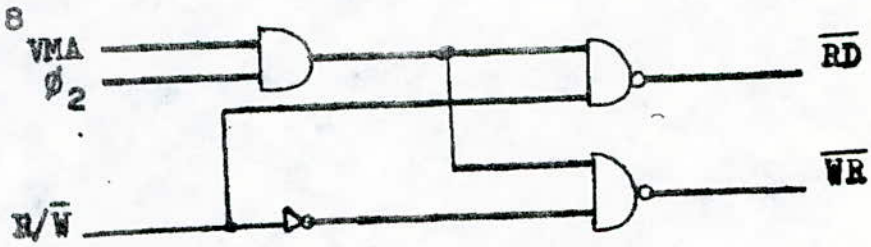
REMARQUE : Par manque de ce décodeur on a réalisé le décodage de $\overline{RD}$ et $\overline{WR}$ à l'aide de deux portes NAND, une porte AND et un inverseur.

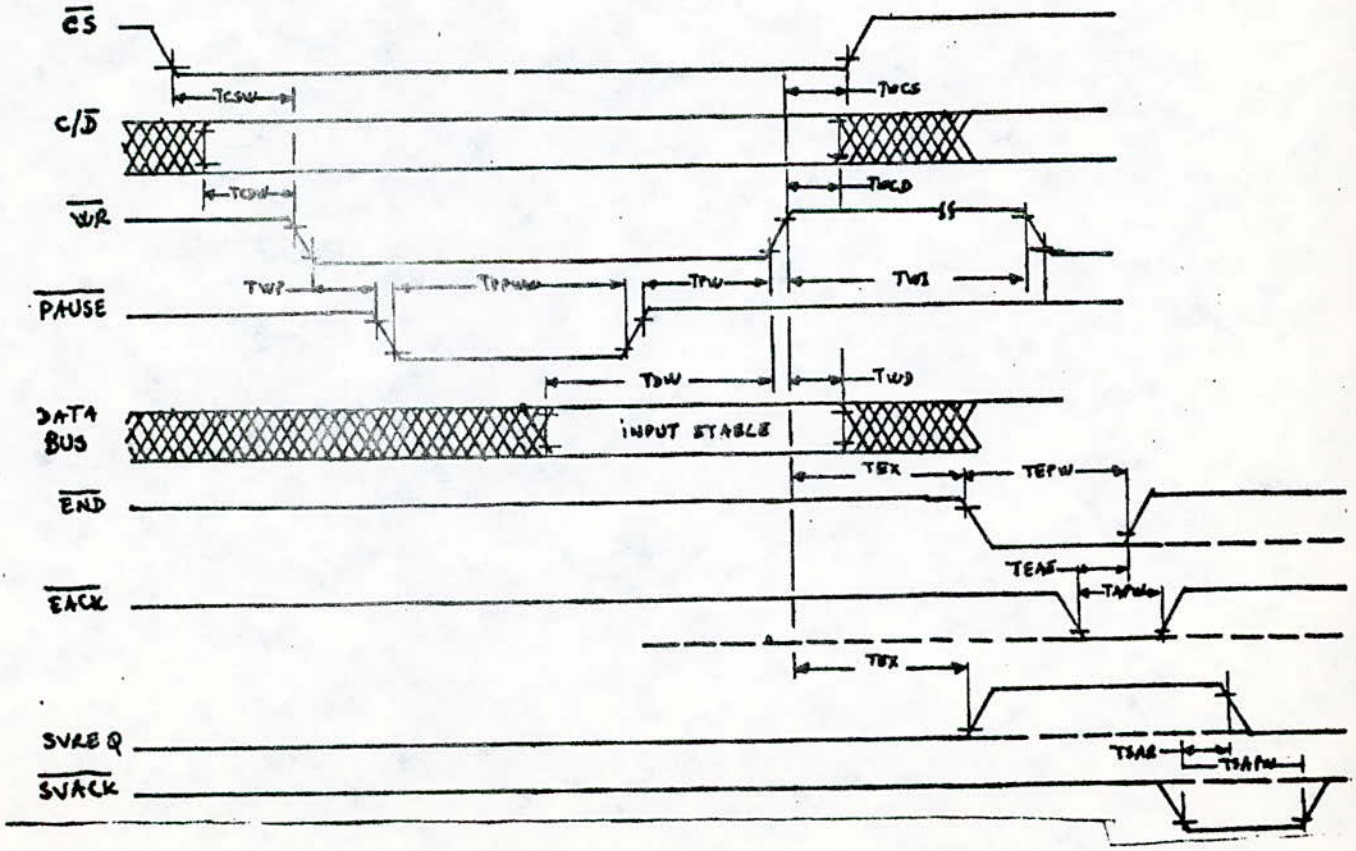
Signal d'ecriture : $\overline{WR} = VMA\ \emptyset_2\ \overline{R/\overline{W}}$

TABLE DE VERITE

VMA	ϕ_2	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$
1	1	0	1	0
1	1	1	0	1

LOGIQUE DE COMMANDE



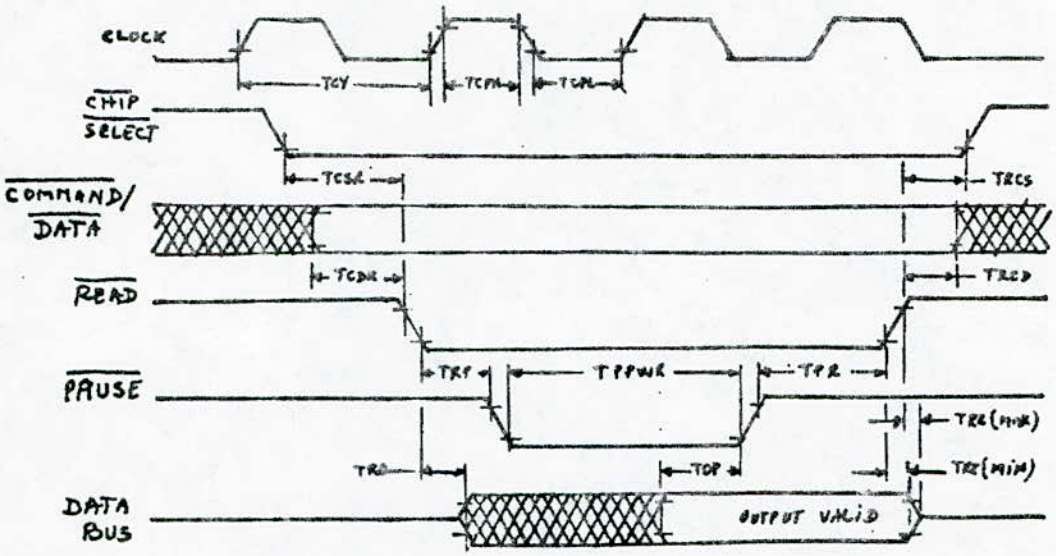


OPERATION DE LECTURE

Le $\overline{RD}$ (lecture) doit être à l'état bas, le signal du µp doit être un signal de lecture, soit $R/\overline{W} = 1$, et le signal d'horloge doit être à l'état "1" ($\emptyset_2$=1). Le signal de lecture est donné par l'équation suivante:

$$\overline{RD} = VMA \cdot \emptyset_2 \cdot R/\overline{W}$$

READ OPERATIONS



INDICATEUR DE L'ETAT DE TRANSFERT

$\overline{PAUSE}$: Cette sortie est reliée à l'entrée MEMORY READY pour prolonger l'état $\emptyset_2$ pour que le transfert, ait le temps d'un échange asynchron de données entre le µp et l'AM9511.

L'AM9511 travaille à une fréquence de 2MHZ ce qui accroit la vitess de transfert tandis que le up travaille à 1MHZ pour synchroniser les deux circuits on doit ralentir l'AM9511. pour cela nous devons créer une logique externe à l'aide de deux inverseurs pour retarder ce signal et une bascule (D) pour la synchronisation.

III ETUDE DE LA MAQUETTE DE SIMULATION

INTRODUCTION

Le developpement d'uh système à microprocesseur consiste soit à developper un interface particulier pour un système à micro ordinateur, soit à developper à des spécifications données. Dans les deux cas à devolopper le logiciel correspondant.

Le développement d'un micro ordinateur pose un problème des plus complexes qui est celui de : comment démarrer la machine sans l'existence d'un système de développement.

En effet, pour demarrer le système il doit se composer au minimum, un microprocesseur, un cristal ou une horloge, une ROM, un port d'E/S, un circuit de contrôle minimum, un décodeur au moins et quelques portes logiques.

Voir figure III.1 .

Finalement le démarrage du système exige un logiciel minimum, Ce logiciel s'occupe de l'initialisation d'un PIA et du système à la mise sous-tension et permettant de vérifier l'essentiel du fonctionnement du système. Ce logiciel intégré dans une EPROM (ou moniteur) qui s'occupe de l'interface avec les E/S et permet l'exécution de quelques commandes élémentaires.

La carte CPU construite qui ne peut compter sur aucune autre pour se démarrer.

Il est important que ce premier programme de mise au pont " soit aussi court que possible et aussi simple que possible sous peine de ne jamais savoir qui, du circuit ou du programme est en faute en cas de non démarrage . En ce qui concerne de 1er programme, il y'a 2 approches possibles

1er approche:

C'est l'approche standard qui consiste à developper un moniteur minimum, à savoir l'initialisation de tout le système ainsi que celui du PIA et l'exécution des instructions qui se trouvent à l'intérieur de l'EPROM (moniteur).

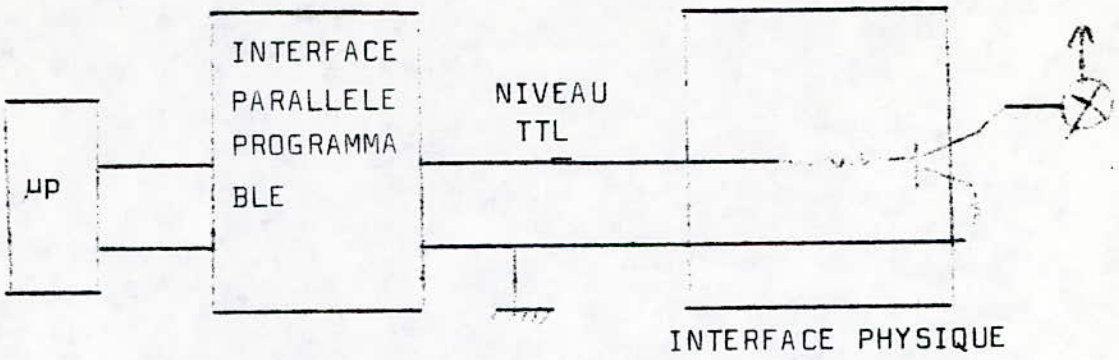
DESCRIPTION DE LA MAQUETTE DE SIMULATION D'ENTREES SORTIES

Le port A qui sera programmé généralement en entrée est attaqué par 8 clés (interrupteurs) qui présentent une configuration de données d'un certains nombres en héxadécimal(binaire) et l'afficher sur le port B qui sera programmé en sortie attaquant 8 Leds qui servent à la visualisation du contenu du,registre de données ORA du PIA.

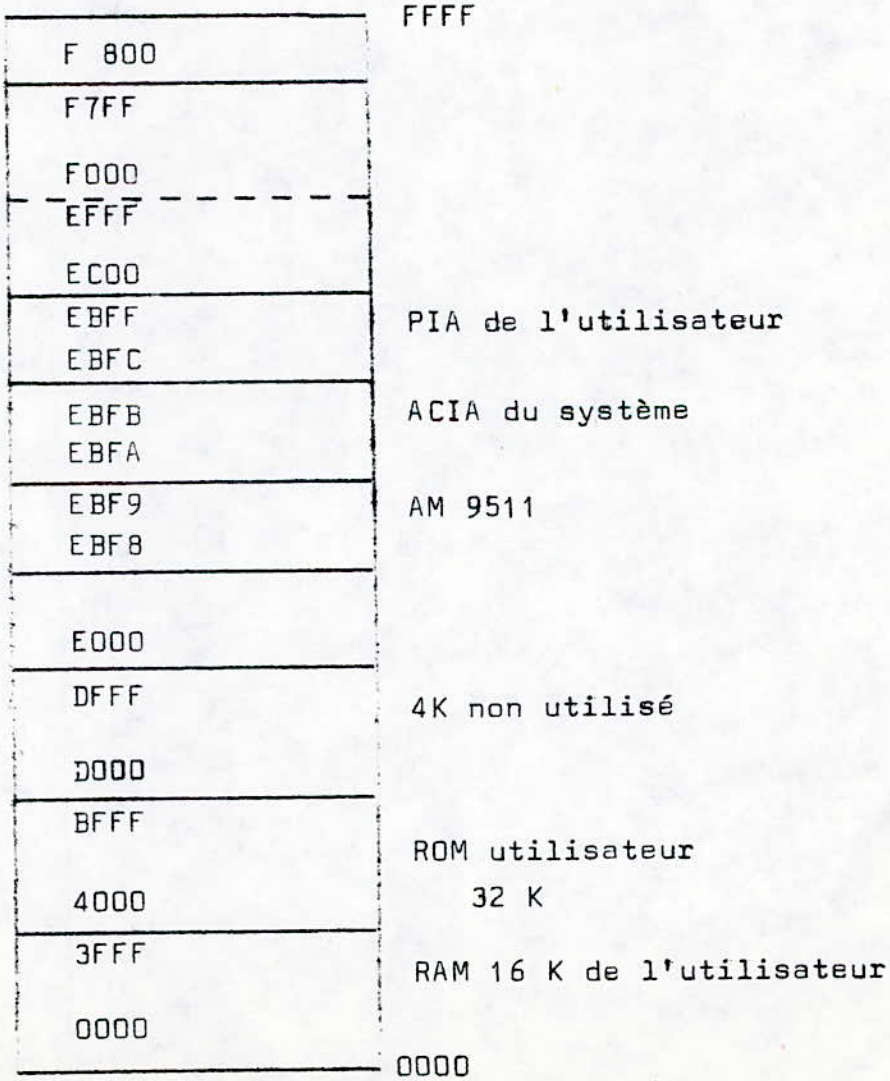
Notons que ces leds sont protégées par un transistor et une résistance attaquant la base du transistor.

La donnée acquise sur le port B sera alors visualisée sur les leds placées de PB_0 à PB_7.

Ramarque : Le système à microprocesseur enverra par l'intermédiaire µp et de l'interface parallèle des niveaux de tension de "0" volt (0:logique) ou +3,5 v (1 logique) sur les organes de sorties (niveaux TTL); Lorsque ces dernières necéssitent des tensions plus elevées ou des courants plus importants que ceux débités par l'interface parallèle, il devient necessaire d'intercaler une interface physique qui peut se réduire à un simple transistor entre le circuit électronique et l'interface programmable.



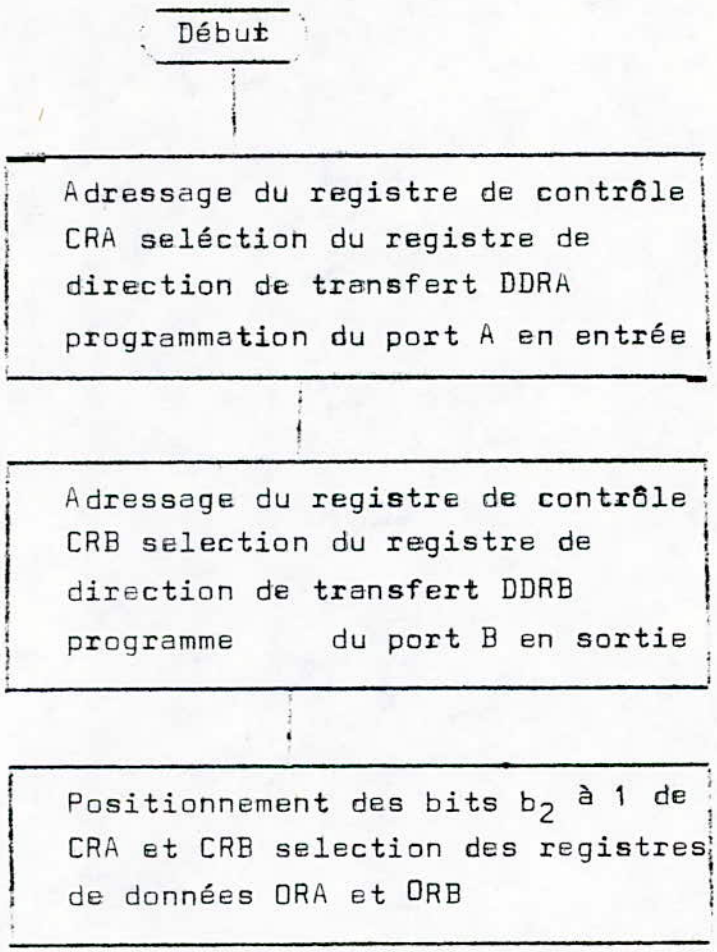
REPARTITION DU CHAMP MEMOIRE



PROGRAMME DE TEST :

Les programmes de test de maquette de simulation seront constitués par:

1) Programme d'allumage des leds



Après initialisation du PIA fixant le port A en entrée. Les instructions (CLR EBFC et CLR EBFF) permettent la mise à zéro du contenu des registres CRB.

Le bit CRB_2 se trouvent alors à "0" et permettent l'accès aux registres DDRB. L'instruction CLR EBFC met à zéro les bits du registre DDRA. Le port A est donc maintenant programmé en entrée.

Pour fixer chaque ligne du port B en sortie, les instructions: (LDA ## $ FF et STA EBFE) sont utilisées pour changer changer l'information entre le MPU et le PIA, il suffit pour cela de charger l'accumulateur A avec la quantité $ 04 puis de la ranger aux adresses EBFD (CRA) et EBFF (CRB) Cette opération fixe les bits (b_2) de CRA et CRB à 1.

Cette sequence s'efféctue grâce aux instructions:

LDA ## $ 04 , STA $ EBFD et STA $ EBFF.

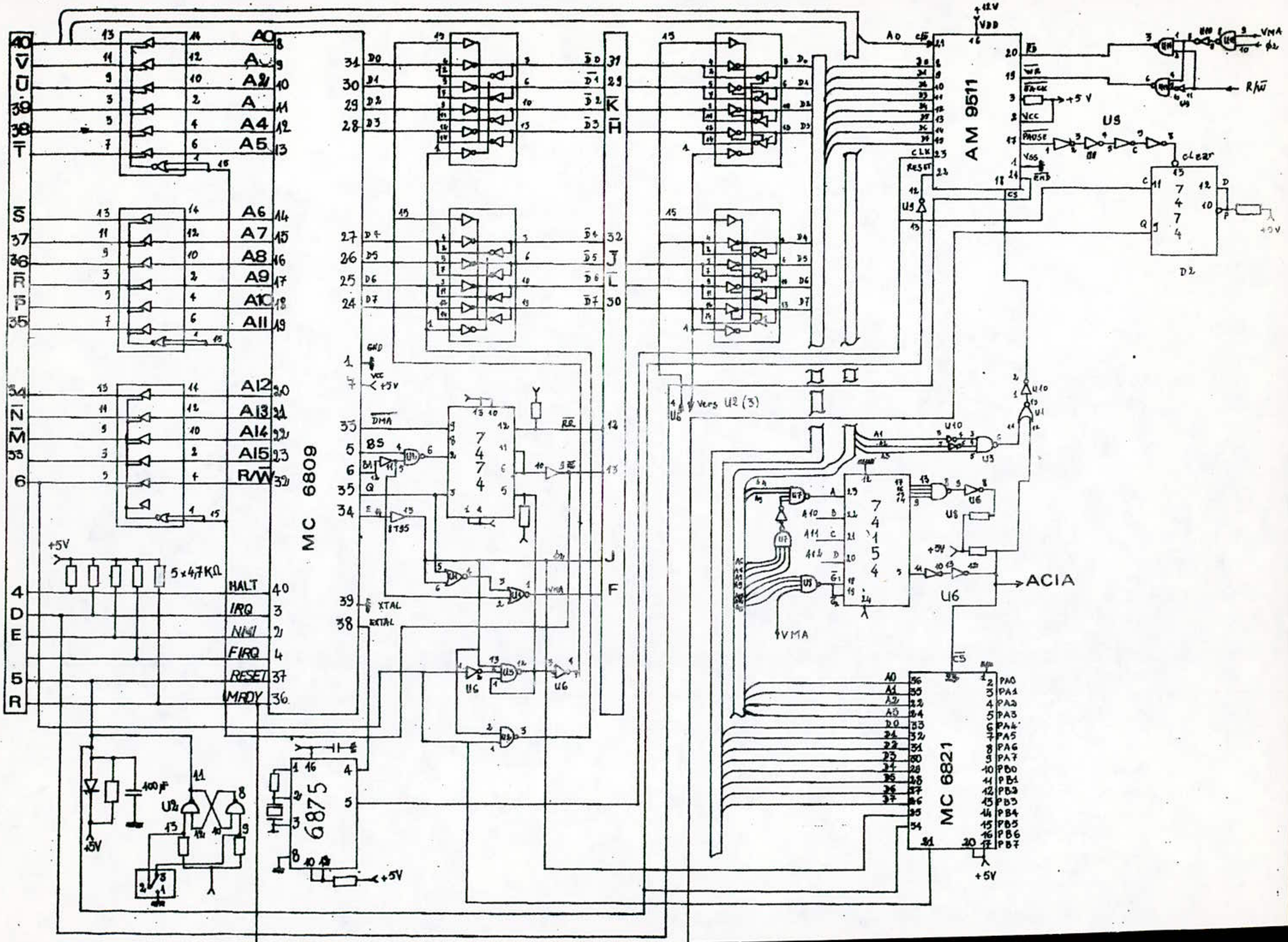
Le programme est mis en exécution à partir de l'adresse F800 de l'EPROM après avoir actionné le bouton de RESET.

Le bouton poussoir BP1 permet l'initialisation de tout le système, une bascule RS réalisée à l'aide de 2 portes NAND élimine l'effet de rebondissement qui est provoqué par l'action sur ce bouton poussoir. L'une des sorties de cette bascule RS est reliée directement à la broche Reset du 6809 commune à tous les circuits de la carte CPU.

2ème approche :

Utilise le fait que l'on a sa disposition un système de développement puissant grâce au mini ordinateur

Il est cependant clair que dans de nombreux cas le système ne démarrera pas et il peut y avoir à cela de trés nombreuses raisons, l'un ou plusieurs des circuits intégrés utilisés sont défectueux, l'une ou plusieurs des 500 connéctions sont mal reliées, une ou plusieurs lignes de code sont erronnées, ou de combinaisons de toutes ces raisons, et cela prend parfois beaucoup d'ingéniosité pour repérer les déféctuosités et démarrer le système.


MC 6809
AM 9511
MC 6821
6875
7474
74154
ACIA
VMA
R/W
+5V
+12V
VDD
5×4,7KΩ
HALT
IRQ
NMI
FIRQ
RESET
MRDY
XTAL
EXTAL
DMA
GND
VCC
CLK
PAUSE
clear
Vers U2 (3)
D2
U6
U8
U9
U10
400 pF
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
R/W
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7

IV ETUDE ET REALISATION DE CARTES MEMOIRES

INTRODUCTION :

Cette réalisation nous permet de toucher concrétement la notion d'adressage de mémoires (logiques, cablage etc.) En outre, il faut remarquer que cette carte est necéssaire à la réalisation du système up qui gerera le processus.

I PRESENTATION GENERALE :

Occupant une zone mémoire de 16K octets (0000-3FFF) necessitant par conséquent 16 boitiers RAM TMM 2016 de 2K×8 bits statiques. Cette carte a été, en fait divisée en 8 pages de 2K octets.

La capacité de la carte ROM est de 32K octets allant de 4000 à BFFF necéssitant par conséquents 8 boitiers EPROM de NE 2732 de 4K octets, cette carte a son tour est divisée en 8 pages et chaque page occupe 4K.

II DECODAGE ET REPARTITION DES PAGES MEMOIRES

Le bus d'adresse du microprocesseur 6809 ne décode que partiellement les adresses des boitiers mémoires (ROM RAM)

En effet, c'est l'etat du bit A_{14} et A_{15} qui détermine en conjonction avec des signaux baptisés $\overline{CS}$ RAM et $\overline{CS}$ ROM, le boitier selectionné, et ce, de la façon suivante:

$A_{15} = A_{14} = 0$ selectionne le boitier RAM par $\overline{CS} = 0$

$A_{15} = 1$ et $A_{14} = 0$

ou $A_{15} = 0$ et $A_{14} = 1$ selectionne le boitier ROM par $\overline{CS} = 0$

Il en résulte que le champ d'adresses défini par les valeurs 0000 à 3FFF est alloué à la RAM et le champ défini de 4000 à BFFF à la mémoire ROM

Le tableau ci-dessous présente la carte de l'espace mémoire des cartes (RAM et ROM).

Adresse héxadécimal	Contenu affécté	Taille
0000 3FFF	RAM	16 K
4000 BFFF	ROM	32 K

En fait, il apparait que la selection du boitier ROM se fait dans la page de 16 K octets. Pour diminuer au maximum le nombre de composants, nous utilisons des EPROM de 4 K octets chacun et un décodeur de type 74 LS 138 1 parmi 8 afin de segmenter les 32 K octets en 8 parties de 4K octets chacune.

III ADRESSAGE DE LA CARTE RAM :

a) Les différents signaux utilisés

- Bus adresse: Les 16 lignes de ce bus sont amplifiées par des buffers 8T43 amplis unidirectionnels non inverseurs validés en permanence et sont appliqués sur un décodeur d'adresse, les seules lignes à décoder etaient les 5 lignes de poids fort des adresses.

Les lignes d'adresses (A_{14}, A_{15}) servent à décoder l'adresse de la carte complète, tandis que A_{11}, A_{12}, et A_{13} sur le décodeur 74 LS 138 qui permet de selectionner 1 boitier parmi 8 et comme validation du décodeur $\overline{G}_{2A} = A_{15}$ et $\overline{G}_{2B} = A_{14}$ qui fixent l'adresse de la RAM de 0000 à 3FFF sans oublier le signal VMA.$\emptyset_2$ sur G_1. Voir figure (IV 1)

Cette solution est trés largement simple et efficace pour la réalisation de nombreuses cartes pour notre réalisation ou sur des micro ordinateurs de tous types.

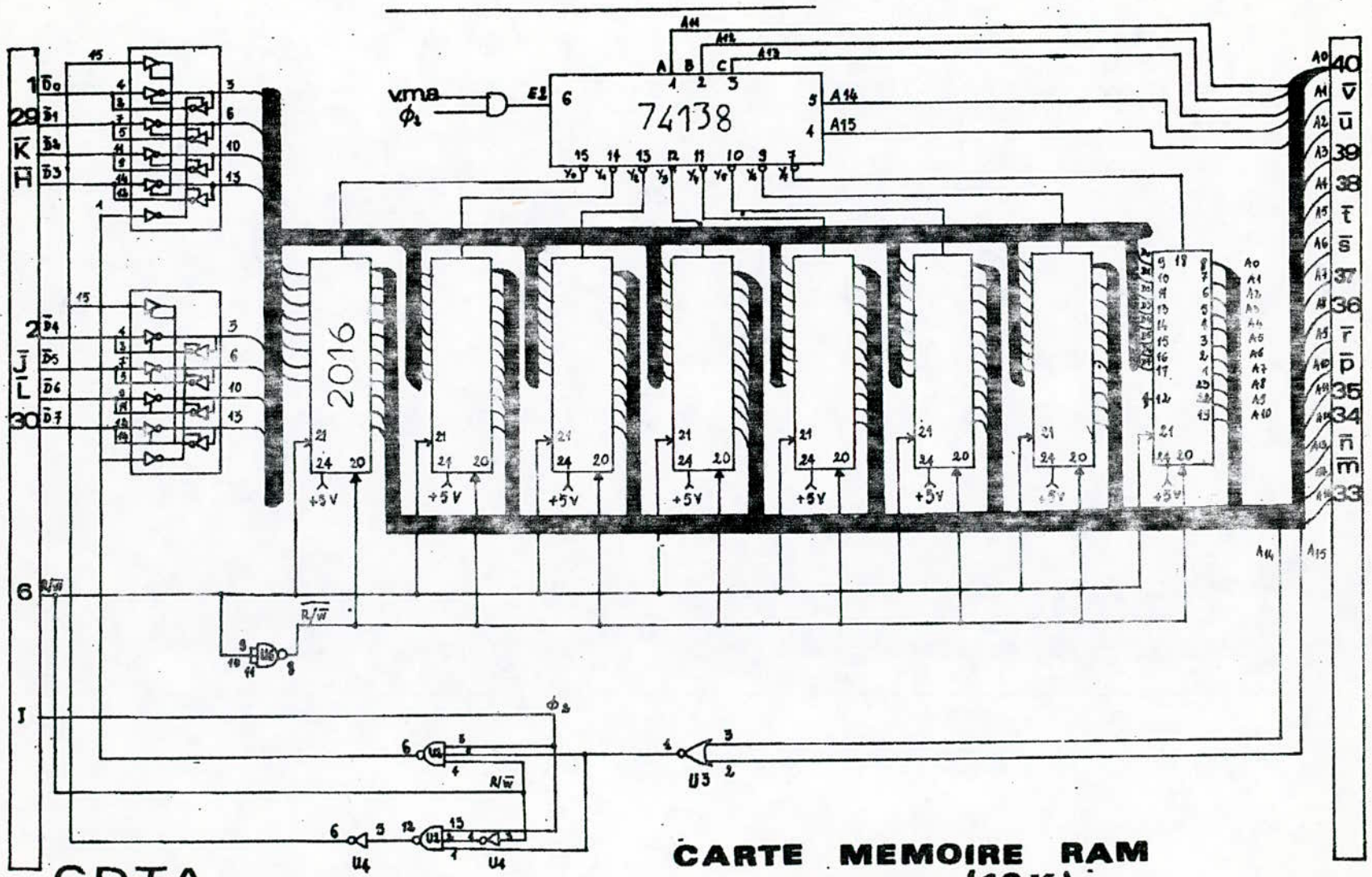
- Bus données : Ces 8 lignes sont amplifiées par des buffers bidirectionnels (8T26) qui seront donc, selon le cas, validés soit en sortie (lecture) soit en entrée (ecriture).

- $\emptyset_2$, R/$\overline{w}$ après amplification ces 2 signaux seront utilisés pour la validation du sens de transfert des données (au niveau des 8T26) cela en combinaisons avec certaines lignes adresse pour l'adressage de la carte RAM.

$$DEI = \overline{R/\overline{w} \cdot \emptyset_2 \cdot \overline{CS}}$$

$$\overline{REI} = \overline{R/\overline{w} \cdot \emptyset_2 \cdot \overline{CS}} \qquad \text{avec } \overline{CS} = \overline{A_{14} + A_{15}}$$

Le signal R/$\overline{w}$: Ce signal; après amplification sert à déterminer le sens de transfert des données (ecriture - lecture) . Il est relié directement aux boitiers mémoires à la broche $\overline{w}$ et inversé sur la broche $\overline{G}$ (output Enable).



CARTE MEMOIRE RAM

de capacité **(16 K)**

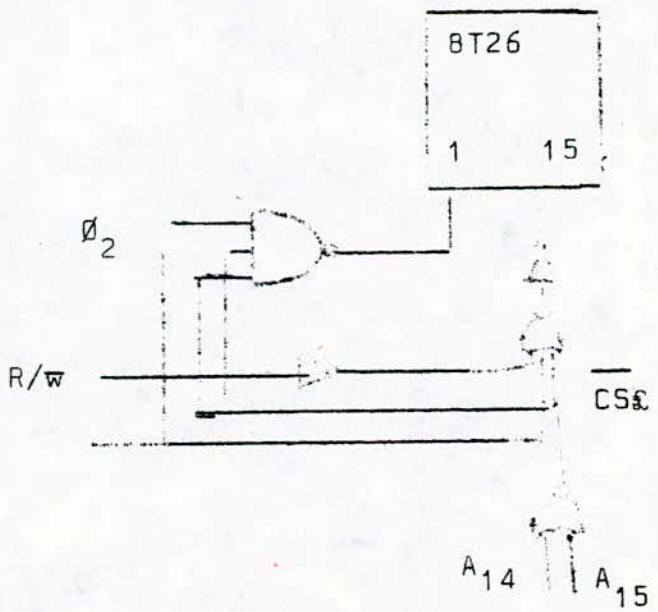
C-D-T-A

LABO-ESR

TABLE DE VERITE DES COMMANDES 8T26

R/$\overline{w}$	$\emptyset_2$	$\overline{CS}$	P1	$\overline{P}_{15}$	Ordre
0	0	1	1*	0*	
0	0	0	1*	0*	
0	1	1	1*	1°	Ecriture
0	1	0	1*	0*	
1	0	1	1*	0*	
1	0	0	1*	0*	
1	1	1	1*	0*	
1	1	0	0*	0*	Lecture

LOGIQUE DE VALIDATION DES BUFFERS 8T26



4. ADRESSAGE DE LA CARTE ROM

L'information $\overline{CS}$ est disponible sur l'une des sorties du décodeur qui selectionnera 1 boitier parmi 8 selon le bit d'adresse A_{15}, A_{14} qui arrivent sur la validation du décodeur 74 LS138 ($\overline{G_0}$, $\overline{G_1}$) , qui génere un (0) logique sur l'une de ses sorties du décodeur et séléctionne le boitier voulu. Voir figure IV 2 .

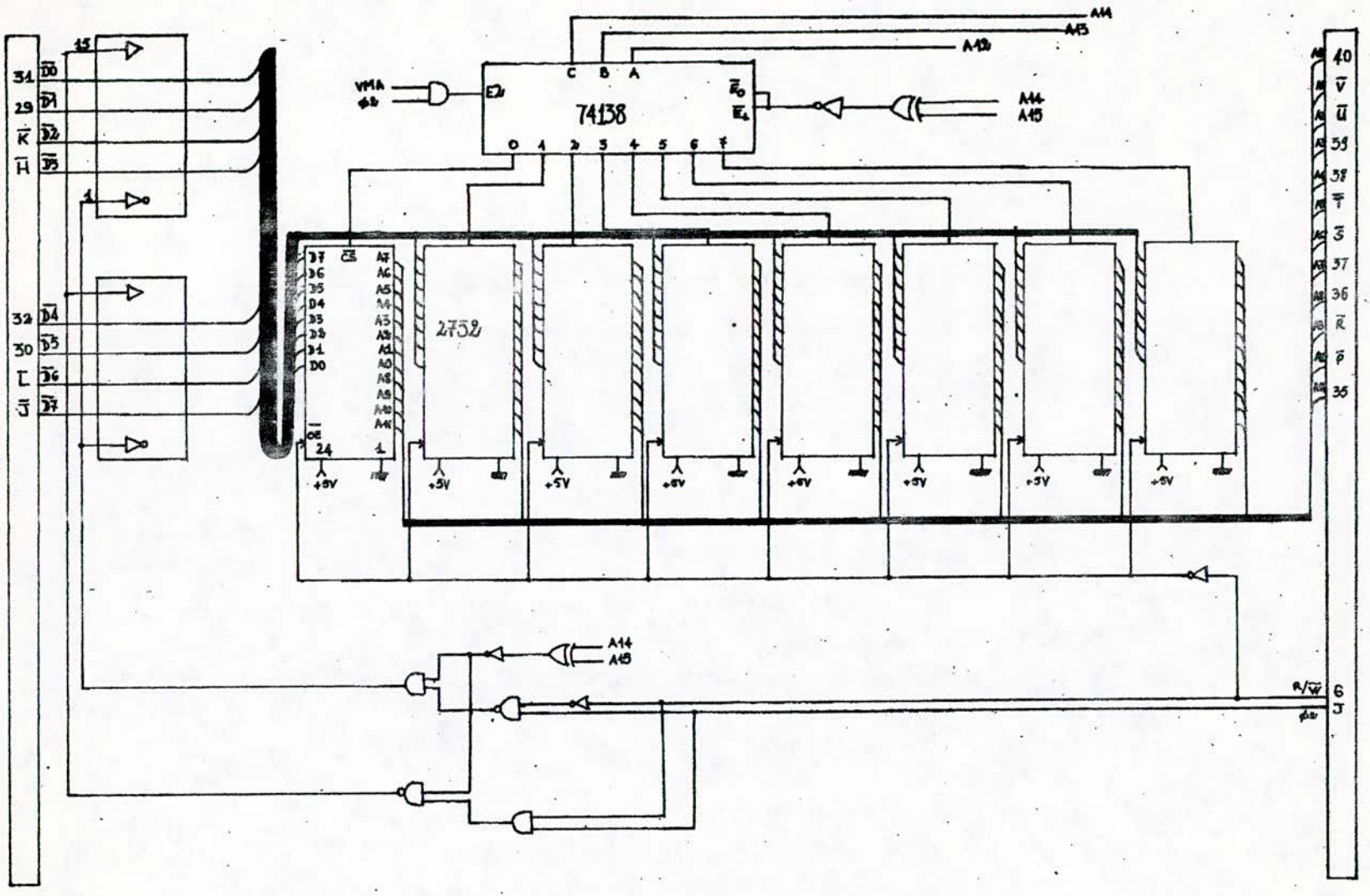
Les différents signaux utilisés:

- Bus adresse : Les 16 lignes d'adresses sont connéctées sur le connecteur de base de la carte qui est reliée par des connecteurs au fond de panier du système. après amplification par les buffers (8T95), les seules lignes à décoder étaient un nombre de 4 lignes de poids fort des adresses A_{14} et A_{15} caractérise essentiellement la difference entre carte RAM et ROM.

Le décodeur 74 LS 138 n'est validé que si la sortie d'un $\overline{XOR}$ ($A_{15} \oplus A_{14}$) est à l'état bas, tandis que A_{13},A_{14} et A_{12} arrivent sur le 74 138 pour décoder chaque boitier.

- Bus donnée: Ces 8 lignes sont amplifiées par des buffers bidirectionnels (8 T 26)comme dans la carte RAM. Mais les 8T26 ne sont validés que si $P_1 = P_{15} = 0 = \overline{\overline{cs}}\ \emptyset_2\ R/\overline{w}$

on aura que l'etat de lecture car les EPROM ne sont déstinés qu'à la lecture après avoir logé le programme de géstion du système.



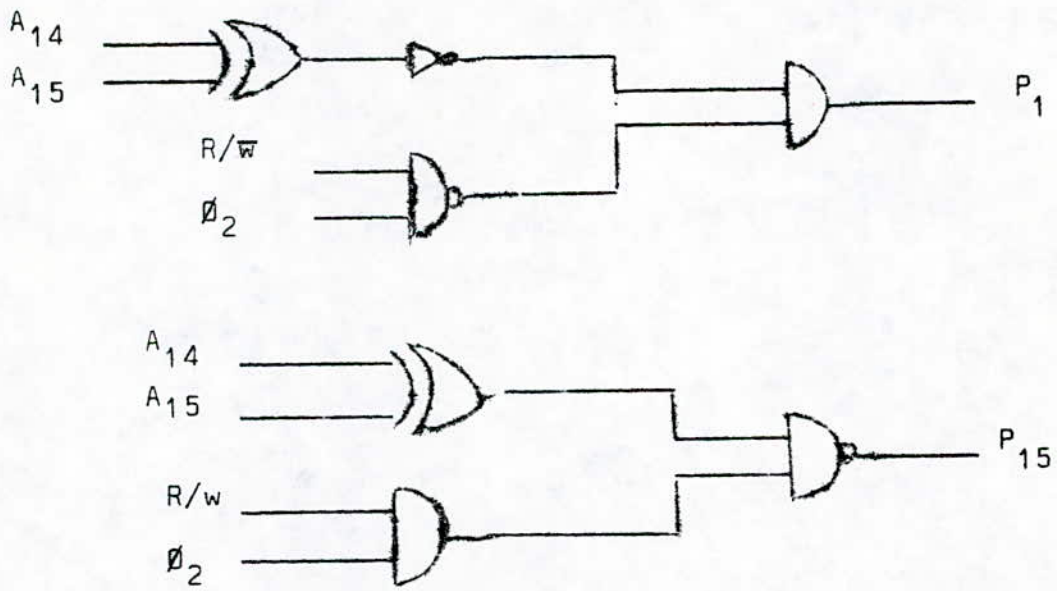
carte MEMOIRE ROM

LABO. E.S.R

C.D.I.A _ CEN

TABLE DE VERITE DES COMMANDES DES 8 T 26

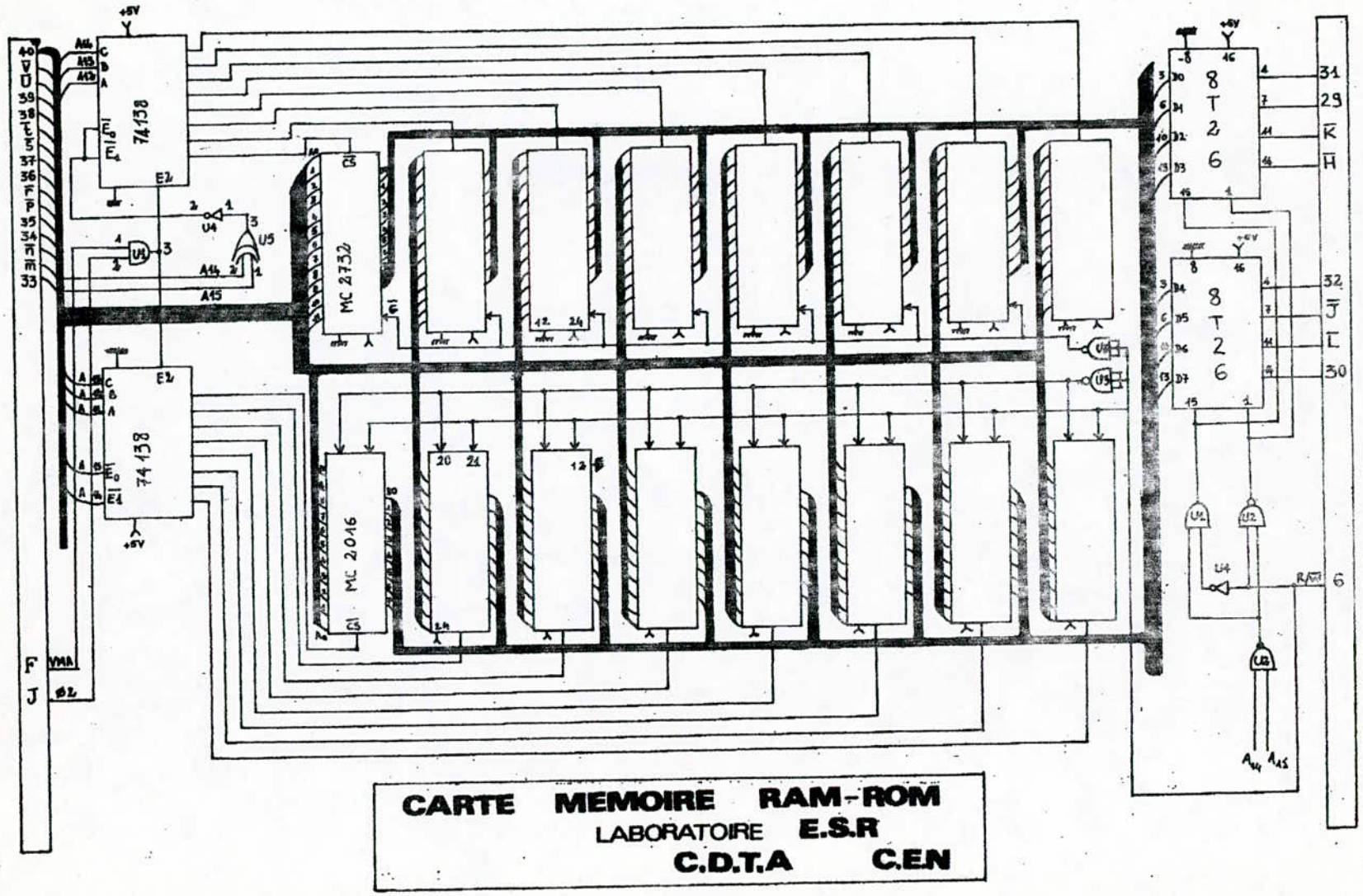
R/w	$\emptyset_2$	A_{14}	A_{15}	CS	$\overline{CS}$	P1	P_{15}
1	1	0	0	0	1	1	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	1	0
0	1	1	1	0	1	1	1



LOGIQUE DE COMMANDE POUR CARTE ROM

REMARQUE / La carte réalisée est une carte qui comporte des boitiers de RAM de 2K type TMM 2016, de champ memoire total de 16K et des boitiers ROM de capacité AK dont le champ mémoire total est de 32K.

Vu le manque de matériel on a réduit la logique de commande en une seule logique qui utilise un adressage complémentaire pour les types de mémoires. Voir figure IV3 .


74138
MC 2732
MC 2016
8T26
+5V
VMA
Ø2
A15
A14
RAM
CARTE MEMOIRE RAM-ROM
LABORATOIRE E.S.R
C.D.T.A C.E.N

CHAPITRE III. APPLICATION SUR LA CARTE RAPIDE

APPLICATION SUR UP 6800.

INTRODUCTION:

La réalisation d'une carte rapide à base de up 6809 n'a pas été réaliser sans nous poser des difficultée divers Cela est dû au manque de systéme de développement.

De plus la non-équivalence et la non-compatibilité totale du logiciel des deux up (6800,6809);

Nous ont permisde réaliser une carte rapide avec L'U.A.R (AM 9511),de tester les différents opérations éffectués par L'AM sur l'exorciser (avec sa carte C.P.U 6800).

Entre autre,cette carte était utilisé pour éxécuter certains sous programmes que nous allons décrire par la suite.

Ces applications sont loin de réfléter les capacités réelles de traitement des autres commandes de calcul(de transfert, de conversion,) de L'AM.

Ces applications concluent le bon fonctionnement de la carte U.A.R.

PRESENTATION DES DIFFERENTS SUBROUTINE :

Chaque sous-programme est présenté sous la forme suivante:

- principe
- algorithme

CARTE RAPIDE

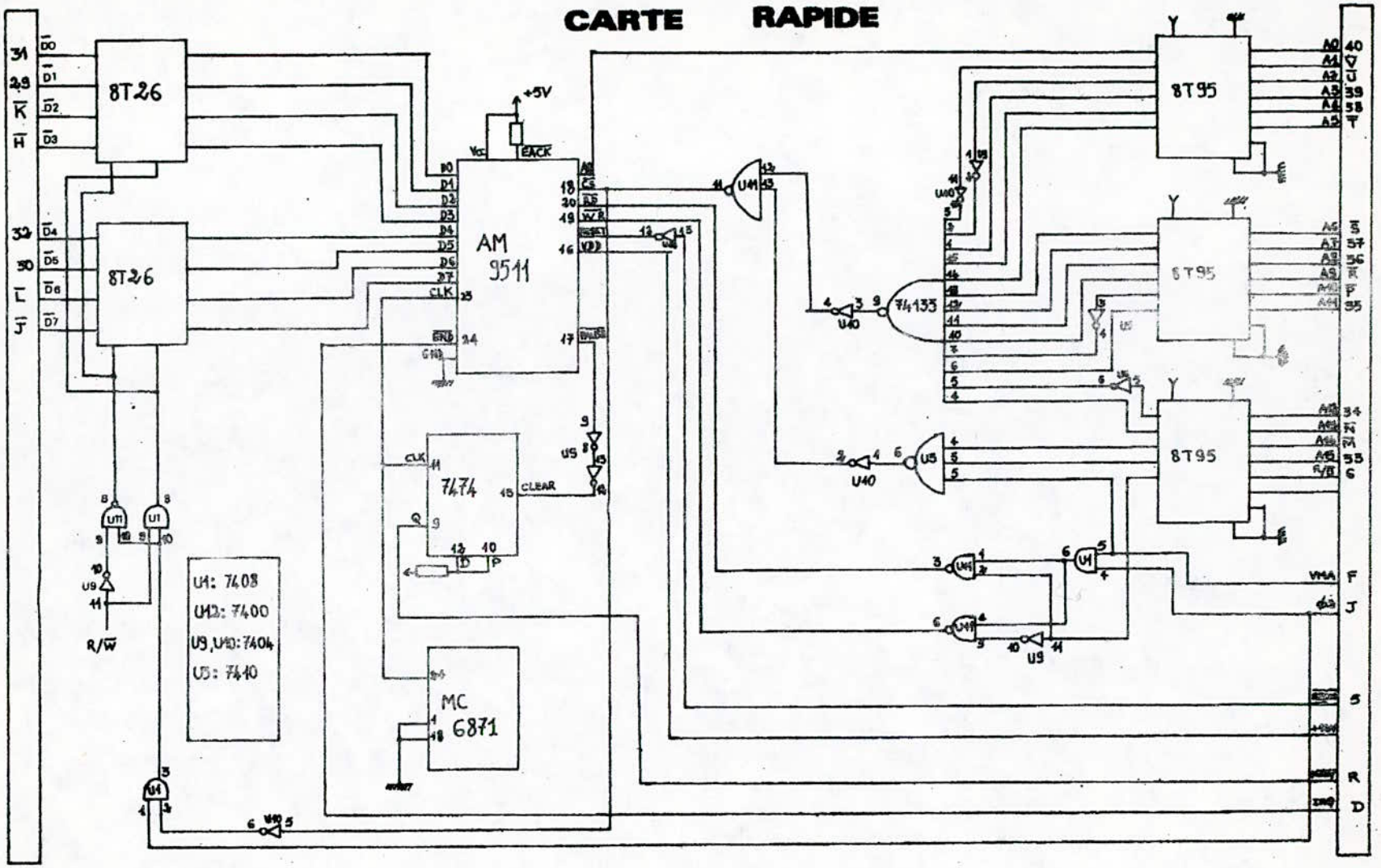


fig V

CDTA LABO. ESR

- organigramme.

N.B: LE LISTING ASSEMBLEUR EST DONNE EN ANNEXE .

1) sous-programme d'initialisation: (INIT)

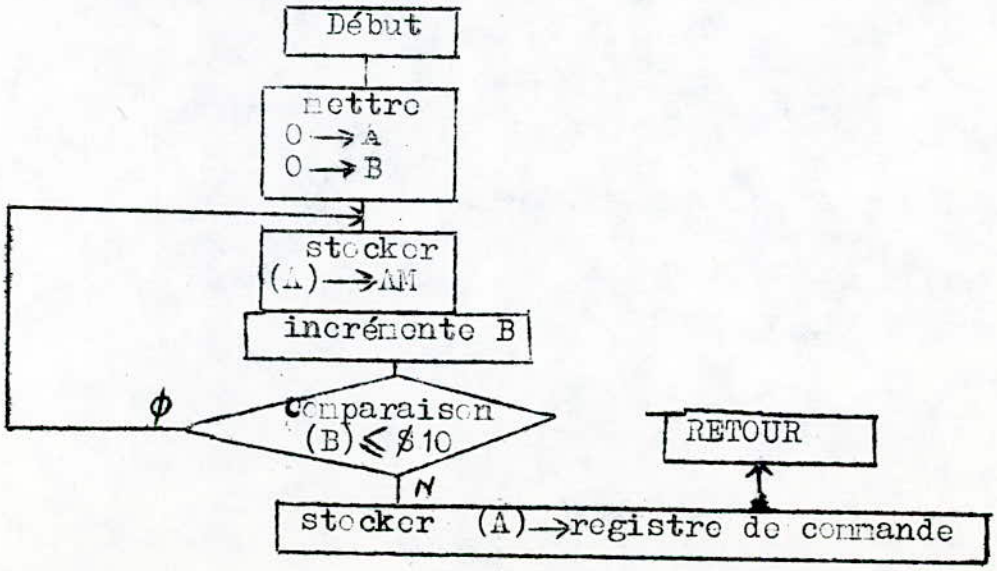
a) principe:

L'opération d'initialisation de l'AM9511 consiste à vider tous les registres internes, ainsi que sa pile de tout opérande, ou commande qui peut se trouver aprés un calcul. (son espace mémoire: ∅0094 à ∅00A3).

b) algorithme:

Mettre "0" dans l'ACC.B qui servira comme comp-teur de boucle. A chaque entrée d'une opérandes "00" dans la pile de l'AM. Il s'incrémente, jusqu'a ce que toute la pile ne renferme que des zéros.

c) organigramme:



2) sous-programme d'envoie de données:(DATA)

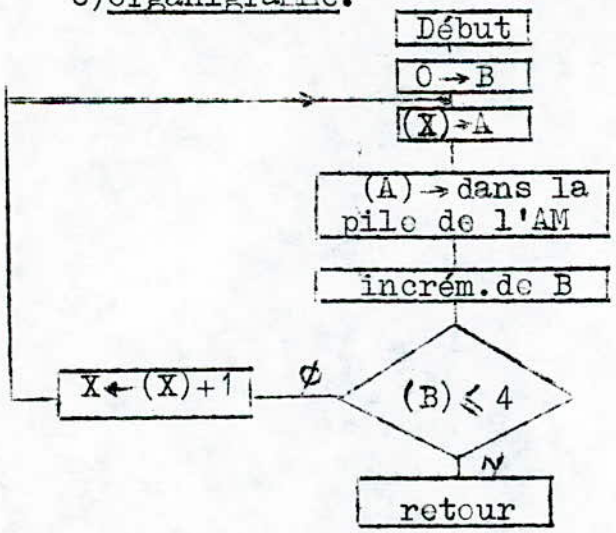
a)principe:

Cette subroutine permet l'envoie de données de 32bits (4 bytes) vers l'AM.Ces données sont logés dans des positions mémoires bien définis.(son éspace mémoire:$02B5 à $02C5.)

b)algorithme:

Pour faire appel à ce sous-programme,il suffit de charger le registre index X(du 6800)par la 1ère adresse du 1er octet(de la 1ère opérande).Cette adresse contient le L.S.B de l'opérande.

c)organigramme.



3) sous-programme de conversion:(fixe simple precision flottant.)

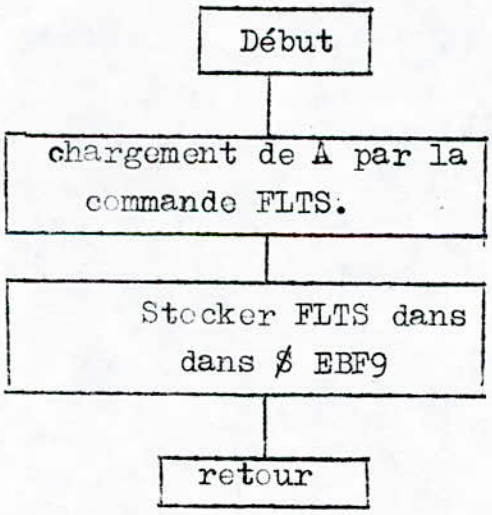
a)principe:

Cette subroutine permet de convertir les nombres du fixe en flottant(32 bits).Par exemple pour calculer le COS. d'un nombre,celui-ci doit-être converti en flottant(La commande COS au niveau de l'AM est en flottant.).

b)algorithme:

Le chargement de l'accu.A par la commande de conversion appropriée (FLTS) du jeux d'instructions de l'AM Puis stocker le contenu de A dans le registre de commande du 9511.

c)organigramme:



4) sous-programme de rangement du résultat:(RANJ)

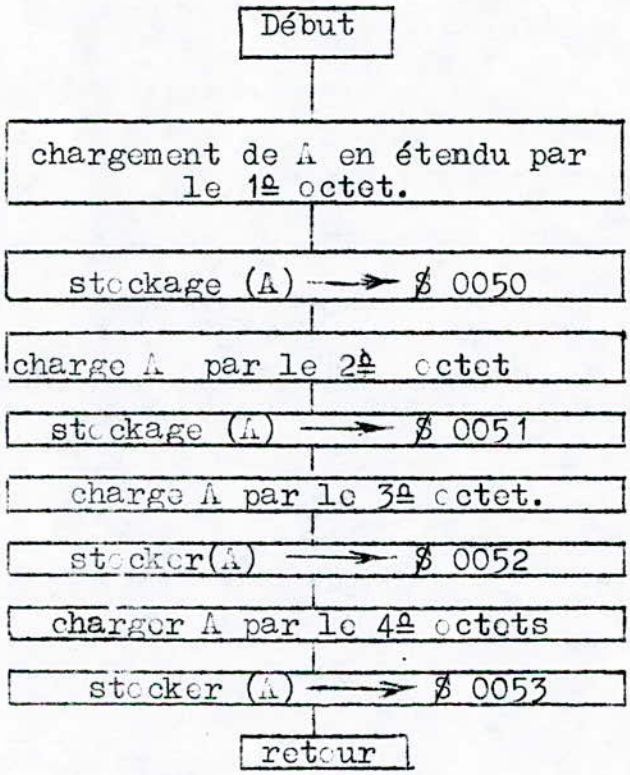
a)principe:

A la fin de calcul d'une opération,le rangement du résultat est nécessaire dans des positions mémoires bien déterminés.(son espace mémoire:∅ 02CC à ∅ 02E7.)

b)algorithme:

Le chargement de l'acc.A par le 1ª octet du résultat (M.S.B).Puis son stockage dans la position $ 0050. Ensuite ,on procède de la même façon pour les 3 autres octets restants dans les positions respectives $ 0051, $ 0052, $ 0053.

c) organigramme:



5) Programme test:

Ce programme test respectivement le S.P.G. DATA le S.P.G. de CONVERSION et le S.P.G RANJ.

a)principe:

La donnée en flottant est plaçée aux adresses $ 0253 à $ 0257.Exemple:$ 26 en binaire 0010 0110 . 0
représentation binaire flottant:0. 1001 1000 0000 0000x2^6
Format binaire flottant de l'AM:

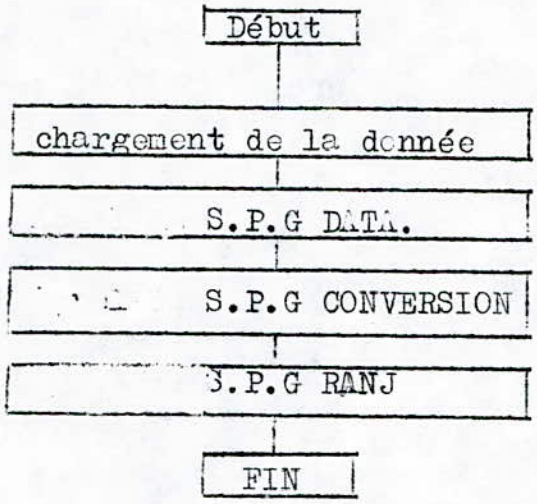
0000 0110 . 1001 1000 0000 0000 0000 0000
0 6 9 8 0 0 0 0 En HEXA.

programme fait la conversion $ 06 98 00 00 ⟶ $ 26

b) algorithme:

Charger la donnée de 4 bytes dans les 4 positions définis en haut. Puis charger respectivement le S.P.G DATA LE S.P.G de CONVERSION (flottant fixe); puis le S.P.G RANJ. Le résultat est plaçé dans les positions $ 0050 , $ 0051.

c) organigramme:



6) sous-programme de calcul de AO:

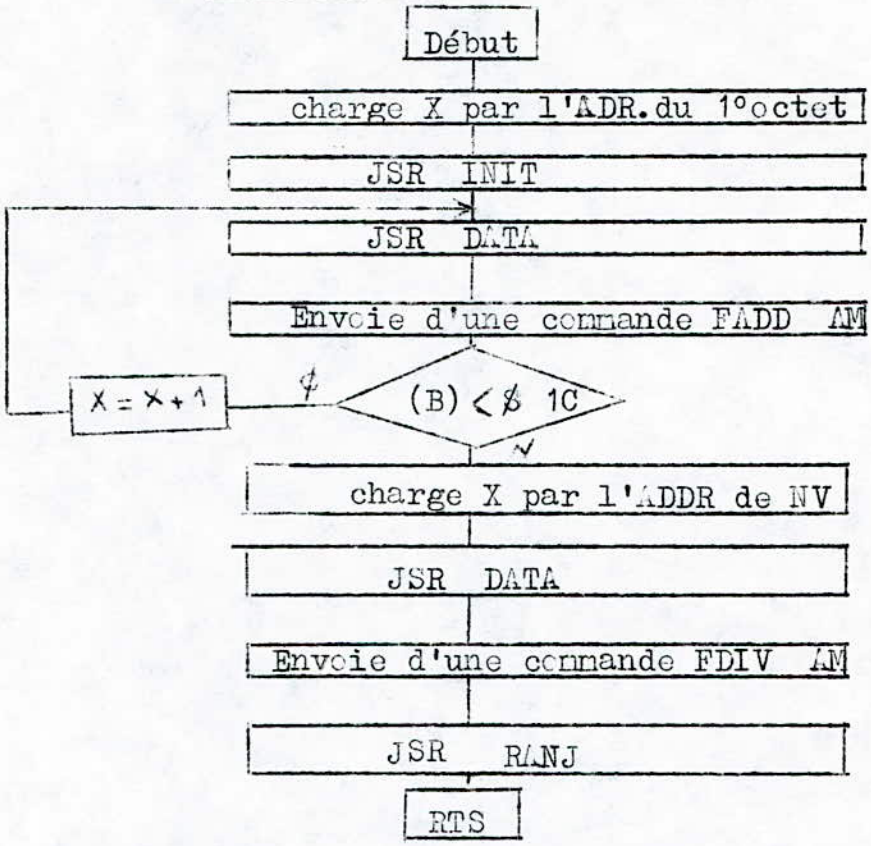
a) principe:

Vider la pile de l'AM par le S.P.G INIT. Faire rentrer la donnée par le S.P.G DATA. Ensuite envoyer une commande d'addition. Le résultat se trouve dans le T.O.S de l'AM. La même opération peut se répéte jusqua ce que tous les opérandes soient prises. Enfin diviser le résultat obtenue par NV.

b) algorithme:

Chargement du registre d'index X par l'adresse du 1ᵉ octet. Puis faire AO=0+X(0), ensuite AO=AO+X(1) et ainsi de suite jusqu'à AO=AO+X(N). Enfin faire AO=AO/NV. Cet algorithme calcul la moyenne de NV valeurs.
remarque: l'acc.B considéré comme un compteur de boucle, pour faire rentrer tous les X(N). Les X(N) sont des opérandes de 32 bits.

c) organigramme:



Remarque: cet organigramme traite un nombre de valeur NV=12.

7)Subroutine COS

Pour calculer COS(2II) on introduit la valeur de donnée en hexadecimal-flottant sur 32 bits, suivi de la commande de donnee PUPI(II). En itroduisant la commande d'operation de multiplication(FMUL), une fois que la multiplication est opérée en introduit la commande COS .

Algorithme:

Les donnees étant introduites dans des positions memoire qui lui sont resery é . On utilise la subroutine DATA (entrée de données dans lapile de L'A.)

L'envoie de la commande de donnée pupi est necessaire pour introduire II dans la pile .

Faire appel ensuite a la commande FMUL en flottant (sr=o) le resultat 2.II en flottant qui se trouve en NOS; Pour prendre son COS. On doit faire intervenir la commande d'echange tos NOS En dernier lieu on introduit la commande COS (avec sr=1) le resultat est ainsi disponible surle bus de donnée.

On fait appel a la subroutine rANJ POUr ranger le resultat .

CONCLUSION:

Rôle de l'unité AM9511

Cette unité permet de libérer le µp de la résolution de fonctions mathématiques complêxes tel que SINUS, COSINUS LOG, e^X,.....mettant en jeu des opérandes à virgule fixe (16bits) ou à virgule flottante (32bits).

Rôle du up

Le µp reçoit les différentes données à partir du clavier ASCII, il convertit ces données du décimal flottant en binaire flottant adaptable avec l'AM9511.
Il véhicule les opérandes stockés dans l'un de ces accumul--ateurs vers l'AM9511. La zône de commandes d'initialisation contenant toutes les opérations à effectuer est initialisée par le µp. Le signalement d'une fin de service de l'AM9511, vers l'entrée IRQ du µp. Ce dernier synchronise les opérations en terminant l'instruction en cours d'execution, et en fin il s'occupe des différentes parties qui composent l'appli--cation telle que:

- calcul des coefficients de Fourier.

CHAPITRE III: Etude théorique de la D.F.T

GENERALITES

Les méthodes numériques sont utilisées pour développer le modéle de simulation qui est une approximation discrète du système réel . Le choix de la méthode numérique, est subordonné à l'exactitude et l'efficacité désirées par l'uti--lisateur.

Considérons x(t) une fonction continue, quand cette fonction est échantillonnée à des intervalles égaux dans le temps t, cette fonction peut être déterminée par la séquence de nombres x(o), x(T),.....x(nT).

Cette série de nombres donne une description limitée de la fonction de temps car les valeurs de x(t) sont seulement connues aux temps o,T,2T,.....nT .Les autres valeurs de x(t), à d'autres valeurs du temps sont trouvées par interpo- lation .

Nous considérons dans la suite un signal (TS) fonction uniquement du temps. Cette fonction qui peut représenter la variation de la température en un point donné du sol, en lui appliquant la méthode de la D.F.T (programme écrit en FORTRON).

III.1. LA TRANSFORMEE DE FOURIER DISCRETE (D.F.T)

La D.F.T EST une représentation de fourier d'une séqu--ence finie. Aprés echantillonnage le signal continu x(t) correspond à une suite de nombre x(n) ainsi l'integrale

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-2\pi jft}\, dt$$

peut être approchée par une somme finie appelée D.F.T

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-2\pi jnK/N}$$

avec K =0, 1,2, ..., N-1

N =nombre de valeurs de x(n)

on pose

$$W^{nK} = e^{-2\pi jnK/N}$$

X(k) est calculée pour N différentes valeurs de K, le nombre d'opération nécessaires pour la D.F.T est N^2 . si N est élevé ce nombre devient prohibitif pour la durée des calculs et pour la capacité de mémoire de la machine de calcul. Si N est très petit on préfère utiliser une D.F.T directe qui est rendu plus efficace et facile à programmer en appliquant à la

quantité W_N^{nk} l'une des propriétés suivantes:

1°/ périodicité $W^{k+\alpha N} = W_N^k$ α. entier

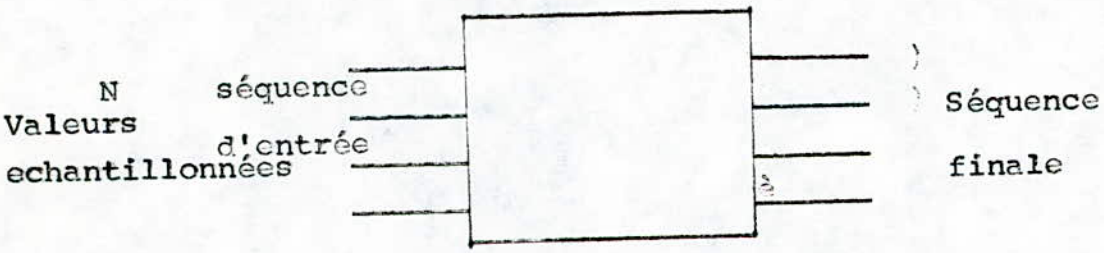
2°/ symétrie par rapport à N/2 $W_N^{k+N/2} = -W^k$

La périodicité de W peut être employé pour réduire plus significativement les opérations qui interviennent dans le calcul de la D.F.T.

III.2. LA TRANSFORMEE DE FOURIER DISCRETE RAPIDE (F.F.T.)

La transformée de Fourier rapide (F.F.T.) "FAST FOURIER TRANSFORM" consiste en une série de procédés de calcul déstinés à réduire le temps d'execution d'une transformation de Fourier discrète

Le processus de la F.F.T. se présente dans un cas général comme suit:



Les résultats de la transformée de Fourrier sont obtenus en ordre de bits inversés suivant N, et une opération simple de réorganisation des résultats est necessaire à la fin de chaque transformée, pour les réordonner. (Voir fig. Graphe de fluence pour N=8)

Le temps de traitement pour le calcul des coefficients de Fourier varie suivant le nbre de valeurs "N" intervenant pour le calcul de coefficient de Fourier.

3.1; La tranformée de Fourier rapide pour N une puissance de 2.

Les algorithmes à base 2 font une économie de temps de calcul trés importante.

Les algorithmes de puissance de (2) sont particulièrement simples à manipuler et souvent dans les applications de manipuler avec des séquences de la forme $N=2^m$

Le principe de la FFT consiste à séparer la séquence initiale de N points en deux séquences plus courtes dont les transformées discrètes de la séquence originale sur N points

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk}$$

avec $W_N^{nk} = e^{-2\Pi j \frac{nk}{N}}$

Les termes W^{nk} forment alors une matrice NxN

Pour N = 4

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_o(o) \\ x_o(1) \\ x_o(2) \\ x_o(3) \end{bmatrix}$$

Cette matrice est décomposée en une matrice NxN en une factorisation de (m) matrice NxN, telle que chaque matrice posséde la propriété de minimiser le nbre d'addition et de multiplication des W_N^{nk} qui existe dans la matrice.

$N = 2^m$ ou $m = \text{Log}_2 N$

m = determine les niveaux d'état de l'algorithme. Donc (m) est le nombre des matrices d'états intermédiaires.

Cette méthode réduit le nbre d'opérations de multiplication à $N \text{ Log } \frac{N}{2}$ et le nbre d'addition à N Log N

Desembrouillage :

Pour la sequence de sortie soit dans un ordre naturel, la sequence d'entrée doit être désordonnée . Quand $N= 2^m$ la séquence d'entrée est rangée de telle sorte que les bits du nbre echantillonnés doivente être inversés afin que la séquence de sortie soit natirellement ordonné.

La définition de l'ordre d'inversion de bits est le suivant

1°) Mettre sous forme matricielle les nombres entiers compris entre 0 et (N-1)

2°) Inverser les formes binaires ainsi obtenues:

3°) Revenir à la notation décimale.

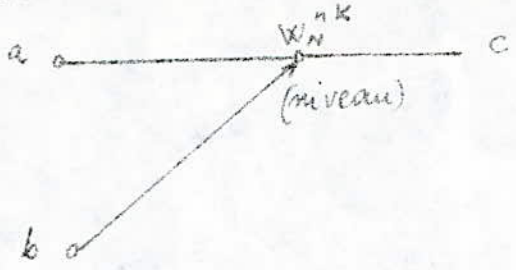
Ex/ $N= 8 = 2^3$

On a 3 digits binaires ($N=2^3$) en inversant les bits bien que cet aménagement puisse apparaitre compliqué, il peut être détérminé de façon relativement simple en suivant l'ordre binaire refléchi, representation symétrique (les nombres donnent l'impression d'être vus dans un miroir) du binaire naturel fig.A

Fig.A- Representation binaire et binaire refléchi des nombres 0 à 7

Valeur	Representation binaire	representation binaire refléchi	Valeur
0	0 0 0	0 0 0	0
1	0 0 1	1 0 0	4
2	0 1 0	0 1 0	2
3	0 1 1	1 1 0	6
4	1 0 0	0 0 1	1
5	1 0 1	1 0 1	5
6	1 1 0	0 1 1	3
7	111 1	1 1 1	7

Convention permettant de lire le graphe de Fluence.



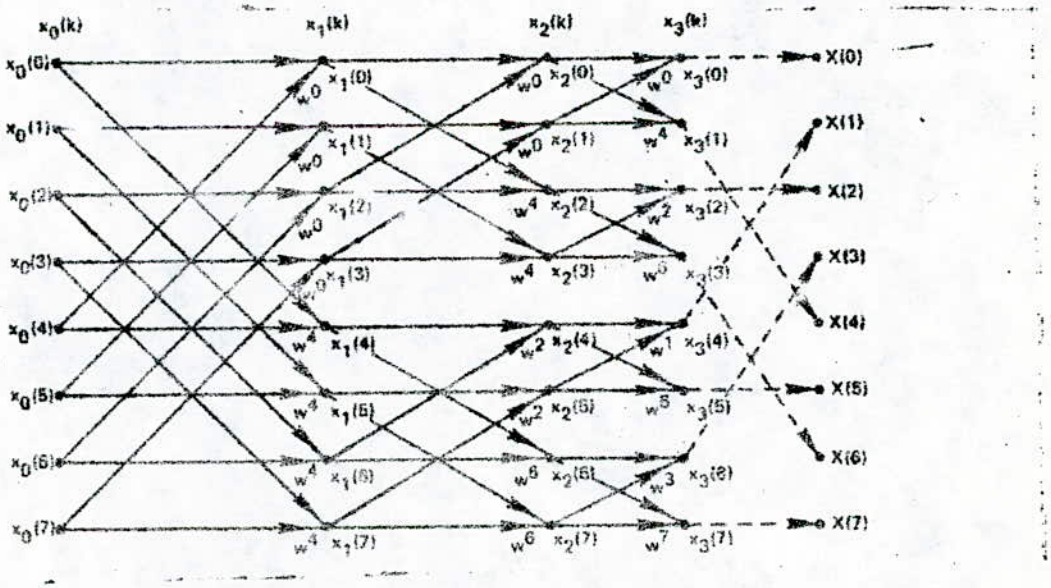
Chaque niveau est obtenu à partir de la combinaison de deux éléments du niveau précédent.

$$C = a + b\,W_N^{nK}$$

$$W = \exp(-2\pi jnK/N)$$

$$W_N^{nK+N/2} = -W_N^{nK}$$

DONNEES D'ENTREES — CALCUL DES NIVEAUX — DESEMBROUILLAGE



GRAPHE DE FLUENCE DE LA F.F.T POUR N= 8.

2°) N : nb paire sans être une puissance de 2.

L'economie de temps significative peut-être obtenue aussi longtemps que N est fortement composite $N = r_1 r_2 \cdots r_m$ r : entier, si $N = r_1 r_2$ la séquence d'entrée peut-être divisée en r_1 séquence de r_2 echantillons, chacune en associant tous les $r_i$eme sous sequences données.

Pour calculer les états intermédiaires, il est necessaire de presenter la sequence d'entrée x(n) et la sequence finale (transformée de Fourier discrète) x(k) sous forme binaire

$$x(k) = x(k_1, k_0) \qquad K_o = 0,1,\ldots.B_1-1$$

$$x(n) = x_o(n_1, n_o) \qquad K_1 = 0,1,2,\ldots.B_2-1$$

$$n_1 = 0,1\ldots\ldots.B_1-1$$

$$n_2 = 0,1\ldots\ldots.B_2-1$$

B_1 et B_2 sont des bases multiples du nb N.

a) Calcul des états intermédiaires.

$$nk = (B_2 K_1 + K_o)(B_1 n_1 + n_o)$$

$$K = (K_1, K_o) = B_1 K_1 + K_o \qquad n = (n_1, n_o) = B_2 n_1 + n_o$$

Tous calculs faits nous obtenons :

$$W_N^{nk} = W_N^{B_1 k o n_1} \; W_N^{(B_2 K_1 + K_o) n_o}$$

La DFT du signal x(n) est :

$$(1)\quad X(K_1, K_o) = \sum_{n_o=0}^{B_1-1} \sum_{n_1=0}^{B_2-1} x_o(n_1, n_o)\, W_N^{B_1 k_o n_1}\, W_N^{(B_2 K_1 + K_o) n_o}$$

En scindant l'expression $X(K_1,K_o)$ en deux parties plus simples, on obtient deux états intermédiaires du signal

$$(2) \qquad X_1(k_o,n_o) = \sum_{n_1=0}^{B_2-1} x_o(n_1,n_o)\ W_N^{B_1 k_o n_1}$$

$$(3) \qquad X_2(K_o,k_1) = \sum_{n_o=0}^{B_1-1} x_1(k_o,n_o)\ W_N^{(B_2K_1+K_o)n_o}$$

b) La séquence finale (transformée de Fourier discrète cherchée)

Cette transformée est obtenue par desembrouillage du second état intermédiaire précedent.

$$(4) \qquad X(K_1,K_o) = x_2\ ((K_o,K_1))$$

c) Graphe de fluence representatif.

Il est constitué par deux niveaux

- Le 1er niveau, representant la relation (2), contient un papillon à base B_2.

- Le 2ème niveau, representant la relation (3), contient B_2 papillons à base B_1.

Ex: $N = 6$

$N = B_1 B_2 = 3 \times 2$

$n_o = 0,1,2$ $\qquad$ $n_1 = 0,1$

$K_o = 0,1$ $\qquad$ $K_1 = 0,1,2$

$$x_1(K_o, n_o) = \sum_{n_1=0}^{1} X_o(n_1, n_o)\, W_N^{B_1 K_o n_1}$$

$$x_2(K_o K_1) = \sum_{n_o=0}^{2} x_1(k_o, n_o)\, W_N^{(B_2 k_{\$} + k_o) n_o}$$

$$X(K_1, K_o) = x_2((K_o, K_1)).$$

Representation matricielle des états intermediaires:

$$\begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(0,2) \\ x_1(1,0) \\ x_1(1,1) \\ x_1(1,2) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & W_N^0 & 0 & 0 \\ 0 & 1 & 0 & 0 & W_N^0 & 0 \\ 0 & 0 & 1 & 0 & 0 & W_N^0 \\ 1 & 0 & 0 & W_N^3 & 0 & 0 \\ 0 & 1 & 0 & 0 & W_N^3 & 0 \\ 0 & 0 & 1 & 0 & 0 & W_N^3 \end{bmatrix} \begin{bmatrix} x_o(0,0) \\ x_o(0,1) \\ x_o(0,2) \\ x_o(1,0) \\ x_o(1,1) \\ x_o(1,2) \end{bmatrix}$$

Calcul du 2ème état intermediaire

Calcul du 2ème état intermediaire

$$\begin{bmatrix} x_2(0,0) \\ x_2(0,1) \\ x_2(0,2) \\ x_2(1,0) \\ x_2(1,1) \\ x_2(1,2) \end{bmatrix} = \begin{bmatrix} 1 & W_N^0 & W_N^0 & 0 & 0 & 0 \\ 1 & W_N^2 & W_N^4 & 0 & 0 & 0 \\ 1 & W_N^4 & -W_N^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & W_N^0 & W_N^2 \\ 0 & 0 & 0 & 1 & W_N^2 & 1 \\ 0 & 0 & 0 & 1 & W_N^4 & -W_N^4 \end{bmatrix} \begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(0,2) \\ x_1(1,0) \\ x_1(1,1) \\ x_1(1,2) \end{bmatrix}$$

La DFT est calculée en B_i niveaux à chaque niveau une nouvelle colonne de N nb est formé à partir de la colonne precedente par une combinaison linéaire des eléments pris 2 à 2. La colonne Bième contient la DFT désirée.

La méthode précedente consiste en définitive à passer par recurrence du signal initial x_o à sa TFD x(n) en faisant intervenir deux signaux intermédiaires x_1 et x_2. Ceux-ci peuvent être considérés comme résultant de substitution efféctuées sur les indices n_1, n_o définissant le signal, par les indices K_o, K_1 définissant sa TFD.

b° La transformée de Fourier discrète cherchée.

La séquence finale est obtenue en renversant les bits du dernier niveau afin d'obtenir un signal ordonné.

$x_2(0,0)$ -----→ X(0)

$x_2(0,1)$ -----→ X(4)

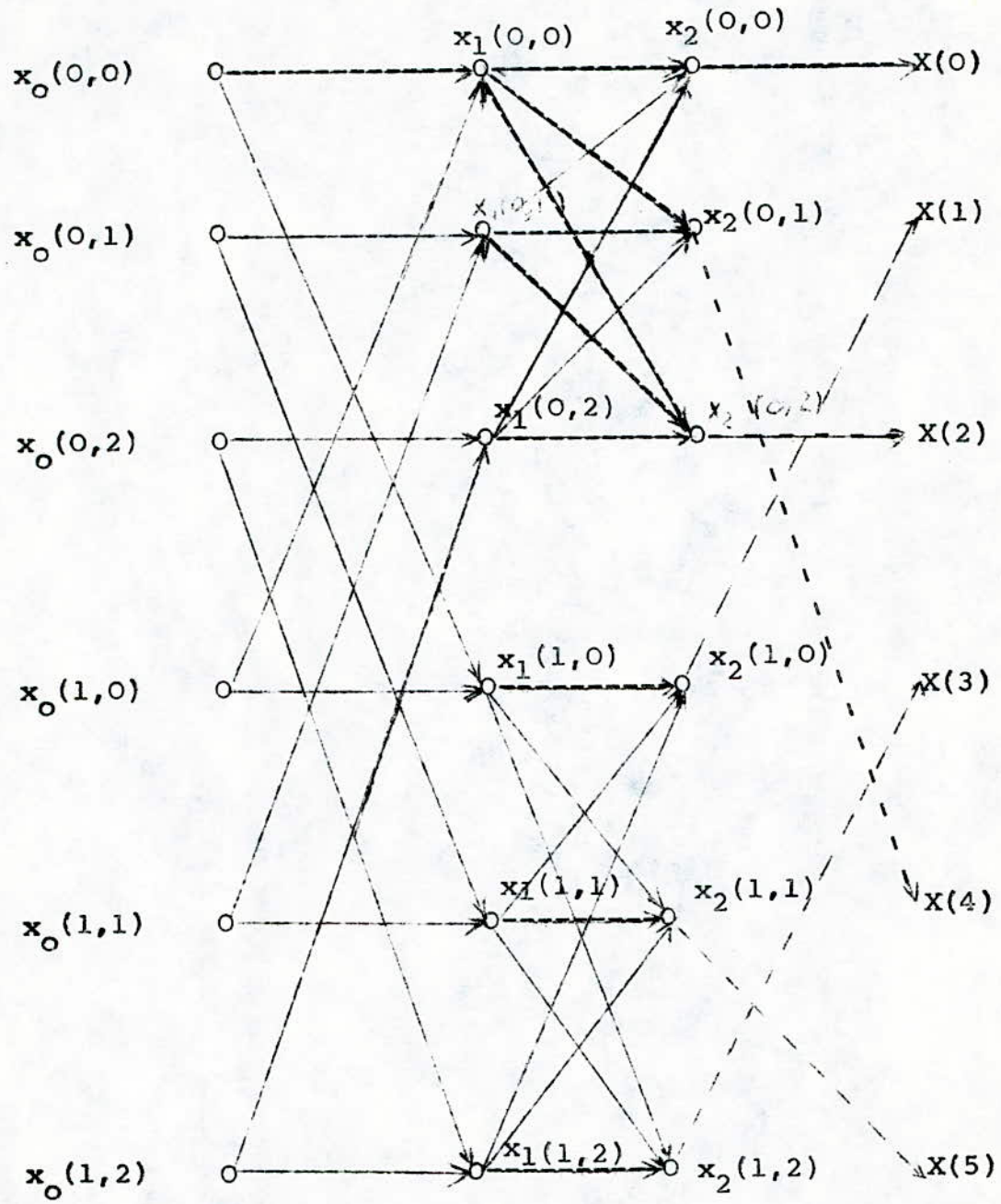
$x_2(0,2)$ -----→ X(2)

$x_2(1,0)$ -----→ X(1)

$x_2(1,1)$ -----→ X(5)

$x_2(1,2)$ -----→ X(3)

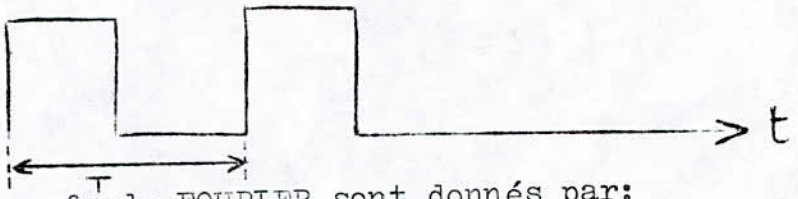
C° Diagramme de fluence N = 6



APPLICATION DE LA TRANSFORME E DE FOURIER DISCRETE:

La représentation spectrale de la D.F.T par ordinateur tels que les résultats sont exactements spécifiés précédemment sont obtenus dans un temps appréciable de calcul, exige une attentive considération et un contrôle des erreurs commises sur les valeurs d'amplitudes du spectre de façon qu'on peut sélectionner judicieusement la précision et l'efficacité de la méthode.

Pour voir l'éfficacité de l'une de ces deux méthodes, on procède par l'étude d'une séquence périodique (de période N=16) d'un signal rectangulaire.

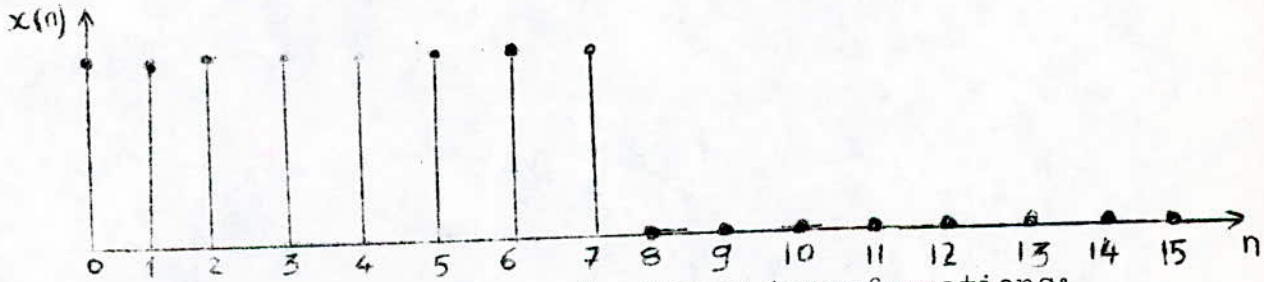


Les coefs. de FOURIER sont donnés par:

$X(K)=\sum_{0}^{} x(n)\, W_N^{nK}$ (1) avec $W_N^{nK}= e^{-j2\Pi/16 \,.\, nK}$

La représentation discrète du signal est:

x(n)=1 pour n=0à7 et x(n)=0 pour n=8à15



La formule (1) peut s'écrire aprés divers transformations:

$$X(k)= e^{-j7\Pi/16} \cdot \frac{SIN(\Pi k/2)}{SIN(\Pi k/16)} \qquad (2)$$

En considérant le module de (2):

$$|X(k)|= \left|\frac{SIN(\Pi k/2)}{SIN(\Pi k/16)}\right| \qquad (3)$$

On obtient ainsi le graphe du spectre d'amplitude X(K)

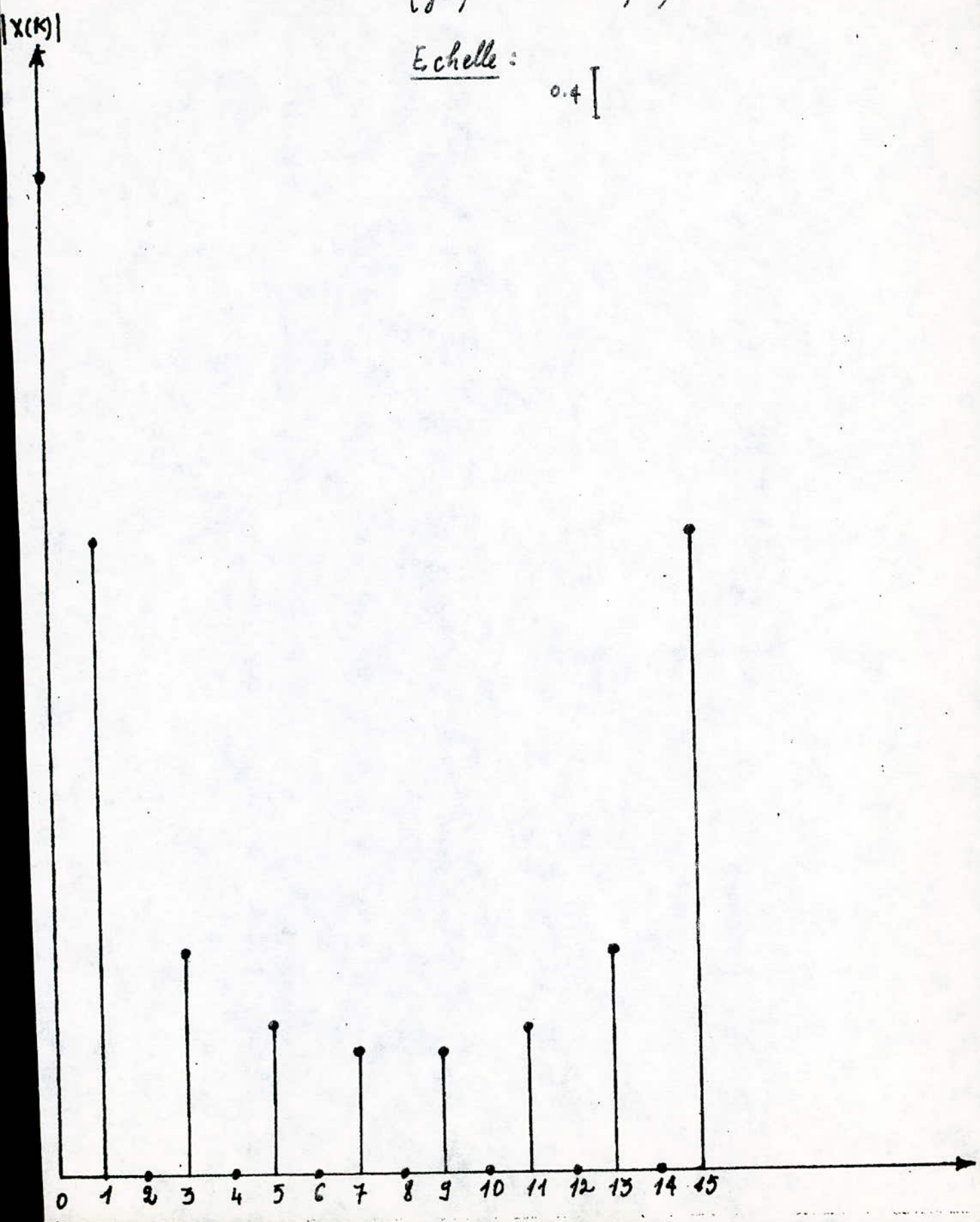
X(0)=8 pour K=0 ; pour K≠0 on applique (3)

Tableau de valeur:

K	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
|X(K)|	8	5.1	0	1.8	0	1.2	0	1	0	1	0	1.2	0	1.8	0	5.1	

Le graphe est représenté en fig. 3

Fig 3: Graphe d'Amplitude des coefficients de Fourier de x(n)
(graphe théorique)



ETUDE COMPARATIVE

La différence du nombre d'opération entre la DFT directe et la FFT n'est pas importante pour de petite valeurs de N.

La supériorité de la DFT augmente avec N,comme le montre le tableau ci-dessous.

NBRE DE POINT N	FFT nbre d'opération $N(r_1 + r_2 + \cdots + r_m) \simeq \frac{Nm}{2}$	DFT $\frac{N^2}{2} + \frac{N}{2}(N-1)$	RAPPORT $\frac{Nm}{2}$
12	24	72	3
16	32	128	4
32	80	512	6,4
48	120	1152	9
64	192	2048	10
128	448	8192	19
1024	5120	524288	102

Pour $N=2^m$ l'algorithme de la FFT est un procédé simple, qui minimise le nombre de multiplications et d'additions.

Si nous prenons l'exemple de N=8 ,la FFT aura 12 opérations de multiplication ($\frac{Nm}{2} = 8.\frac{3}{2} = 12$),et 24 opérations d'addition (Nm = 8x3 = 24),par contre la méthode directe demande $\frac{N^2}{2} = \frac{64}{2} = 32$ multiplications et $\frac{N}{2}(\frac{N}{2} - 1) = 12$ additions (en lui appliquant la symétrie).

Considérons le rapport approché du nombre de multiplication des deux méthodes, nous obtenons : $\frac{N}{Nm} = \frac{2N}{m}$.

Le graphe pris de l'ouvrage (The fast fourier transform) fait une comparaison entre le nombre de multiplication nécessaire par l'algorithme de la FFT et celui utilisé par la DFT directe. (fig. A)

COMPARAISON DES GRAPHES OBTENU

Pour l'utilisation du programme de la FFT l'erreur a été évalué : $/\Delta X(K)/ = \pm 1{,}03$

L'erreur de la DFT a été évalué par : $/\Delta X(K)/ = \pm 0{,}02$

Remarque:

Les programmes de la F.F.T et de la D.F.T ainsi que les valeurs de X(K) sont données en ANNEXE.

COMPARAISON DES MULTIPLICATIONS FAITES
PAR LE CALCUL DIRECT ET L'ALGORITHME F.F.T.

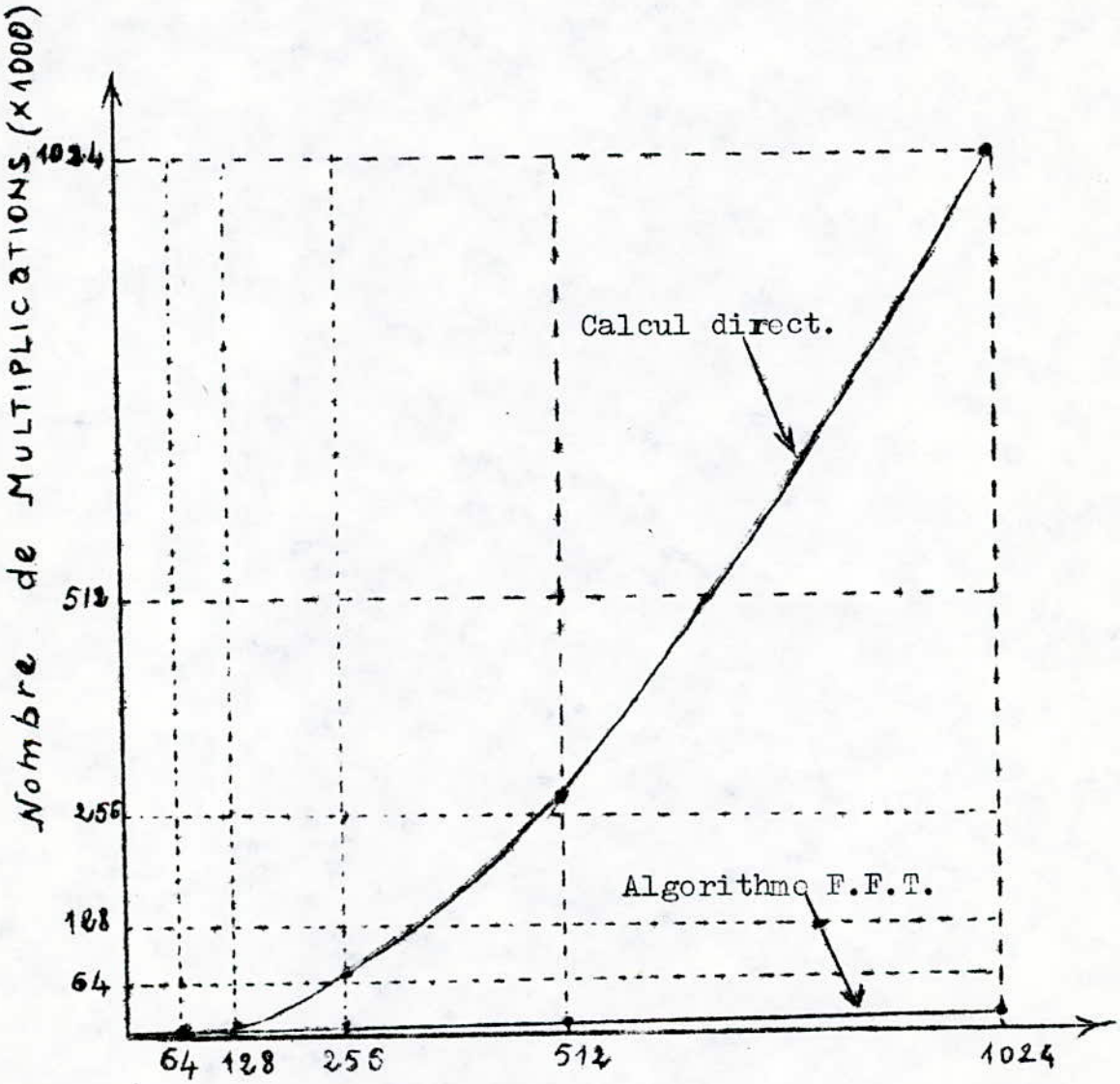


Fig. A

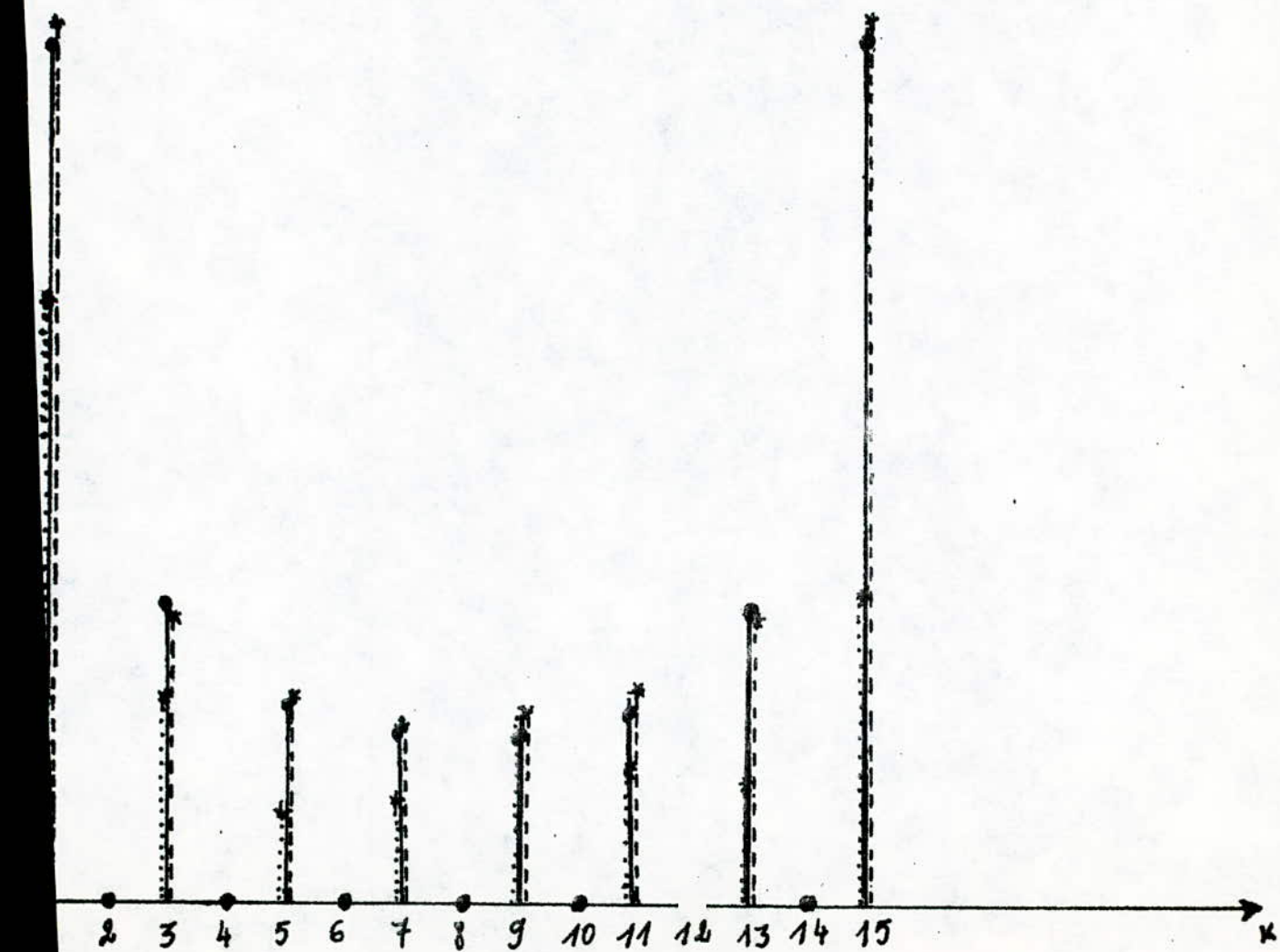
|C(k)|

Echelle :

0,4

Fig 4 : Graphes de comparaison de spectre d'amplitude

———	amplitude de la transformée de Fourier (théorique)
- - - - -	amplitude de la TFD direct
.....	amplitude de la FFT



CONCLUSION :

On remarque que les résultats donnés par la méthode de la DFT directe sont très proches des résultats theoriques,par contre ceux donnés par la méthode algorithmique de la DFT rapide pour un nombre de point petit(N) sont loin des valeurs exactes, ceci a été confirmé par la représentation spéctrale des deux méthodes.

AVANTAGE ET INCONVENIENT DE LA DFT DIRECTE

A- AVANTAGE :

Cette méthode présente l'avantage de la facilité de programmation dans le calcul des W_N^{nK},et n'utilise pas des ittérations.

B- INCONVENIENT :

L'inconvenient de cette méthode se situe comme suit: Plus le nombre N d'échantillons est élevé,plus le programme de résolution se complique,et cela rend le temps d'execution long.

AVANTAGE ET INCONVENIENT DE LA FFT DIRECTE

A- AVANTAGE :

Il résulte des algorithmes de puissance de 2 par leurs simplicités de manipulation, et une économie de temps de calcul important.

B- INCONVENIENT :

La FFT étant plus prérationnelle pour des puissances de 2,les algorithmes FFT deviennent complexe si les dimensions des transformées ne sont pas des puissances de 2, posant ainsi une contrainte à son utilisation. Un algorithme rapide appliqué à des nombres non puissance 2 introduit une complexité de calcul supérieur à celui de la transformée de Fourier directe et nécessite beaucoup de position mémoire.

C O N C L U S I O N

Le travail que nous avons réalisé comprend deux parties essentielles simulation de la décomposition en D.F.T et réalisation d'une carte rapide ainsi que la carte mémoire.

L'utilisation de l'ordinateur nous a permis de comparer précisement les deux méthodes de la D.F.T (D.F.T directe et F.F.T) et de mettre en évidence l'intérêt que peut apporter la D.F.T directe pour un nombre points réduit . Comme nous avons pu le remarquer dans les résultats obtenus, la méthode de la F.F.T n'est pas efficace pour un nombre(N) trés petit.

La réalisation des cartes citées auparavent n'a pas été réalisée sans nous poser de difficultés, d'autant plus que l'unité rapide AM9511 n'a jamais été utilisé dans le cadre des activités du laboratoire ETUDE SPATIALE DE RAYONNEMENT du C.E.N .

Lors de réalisation, de notre travail certains problème ont étés rencontrés. Le manque d'un système de développement à base du 6809 pour développer notre système; nous avons été contraints d'utiliser la méthode de test décrite au chapitre II à l'aide d'une EPROM d'un PIA et des switches sans compter le manque de composants.

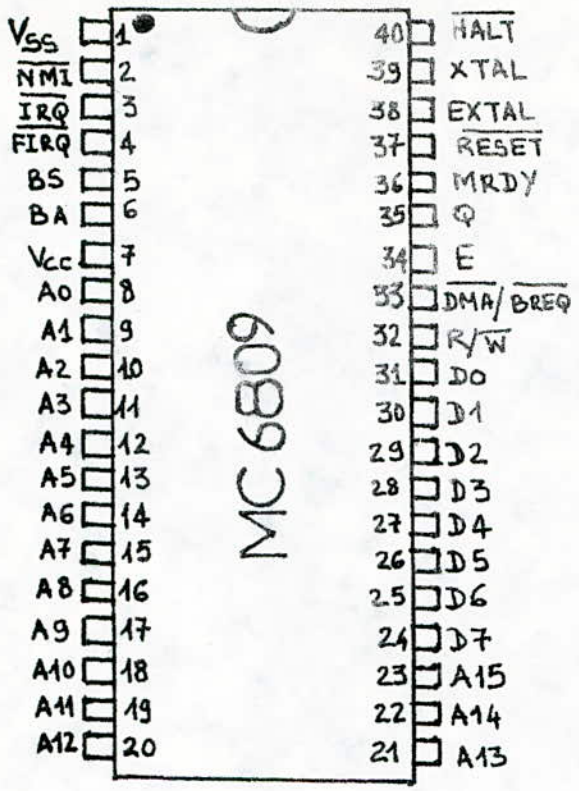
Cependant nous avons utilisé seulement la carte rapide et la carte mémoire sur l'EXOR 6800 pour traiter les différentes opérations qui interviennent dans la décomposition en série de FOURIER , ceci nous permet entre autre de tester le bon fonctionnement de cette carte.

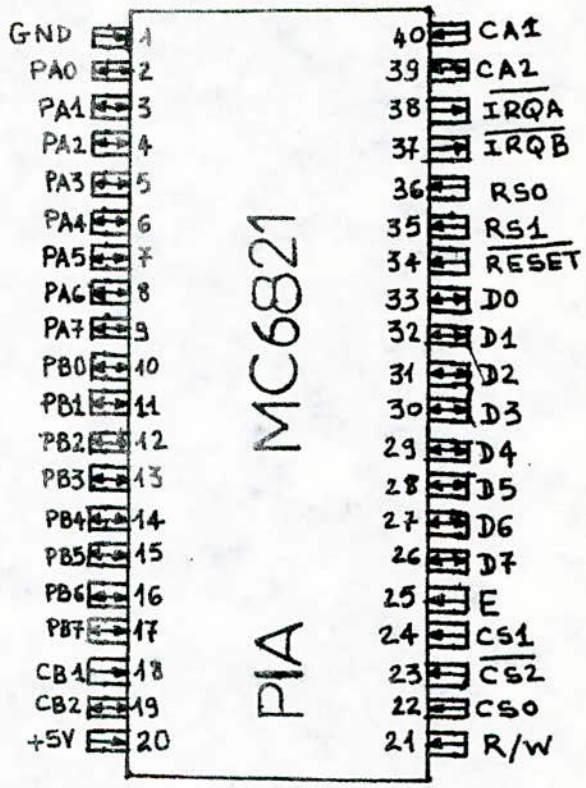
En dernier lieu nous proposons à ce que l'application (Décomposition en Série de Fourier) se fasse à l'aide de l'extension des E/S:

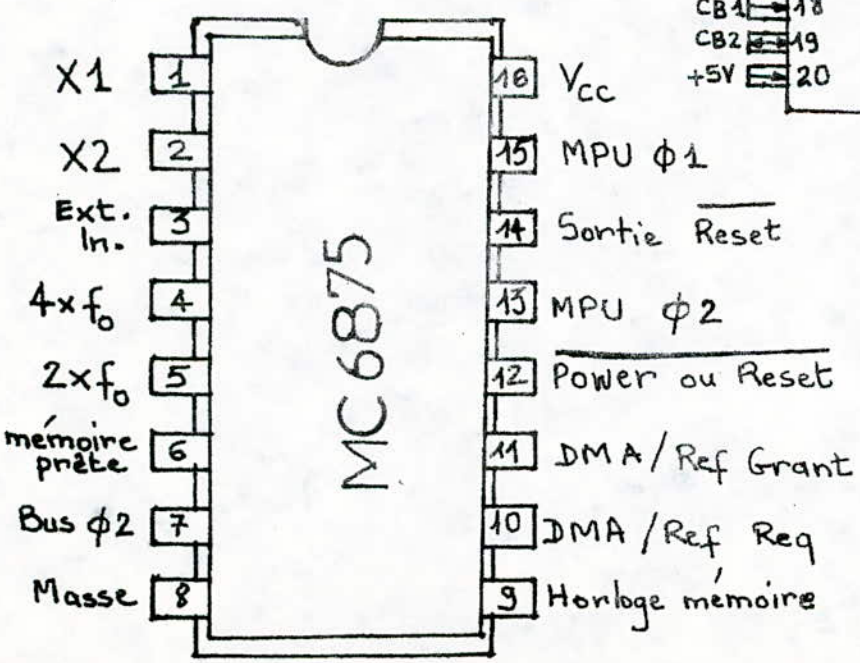
- entrées de données: clavier hexadécimale-binaire.
- sortie de résultats: de connecter une interface de visualisatio sur la carte CPU pour afficher les résultats sur une TELETYPE.

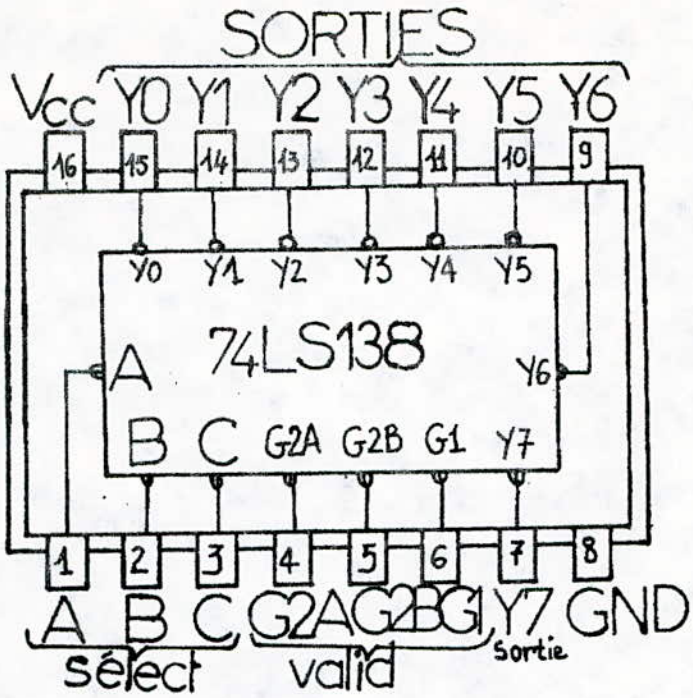
ANNEXE

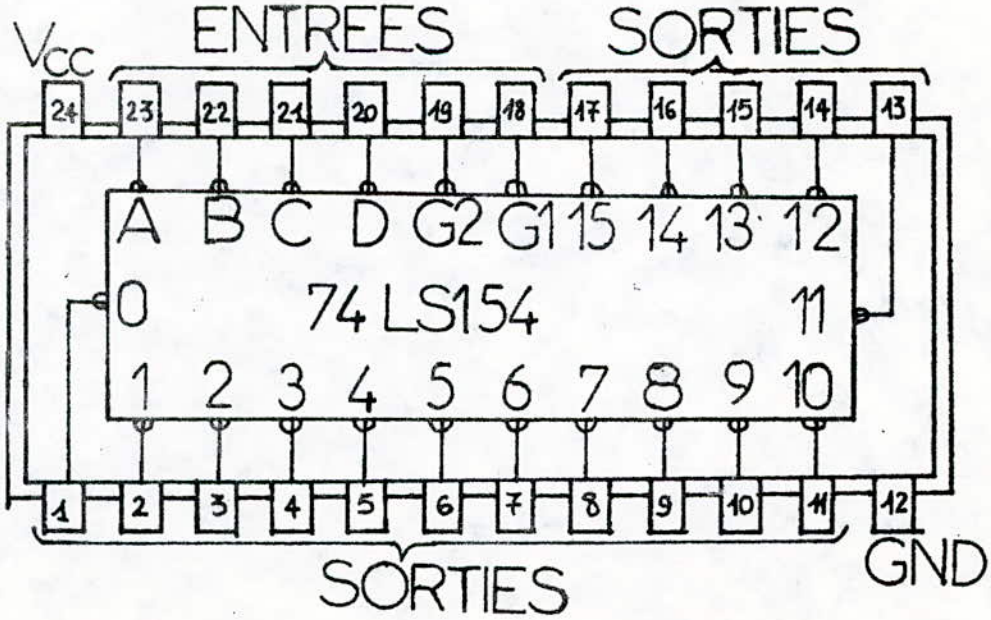
SOMMAIRE DE L'ANNEXE

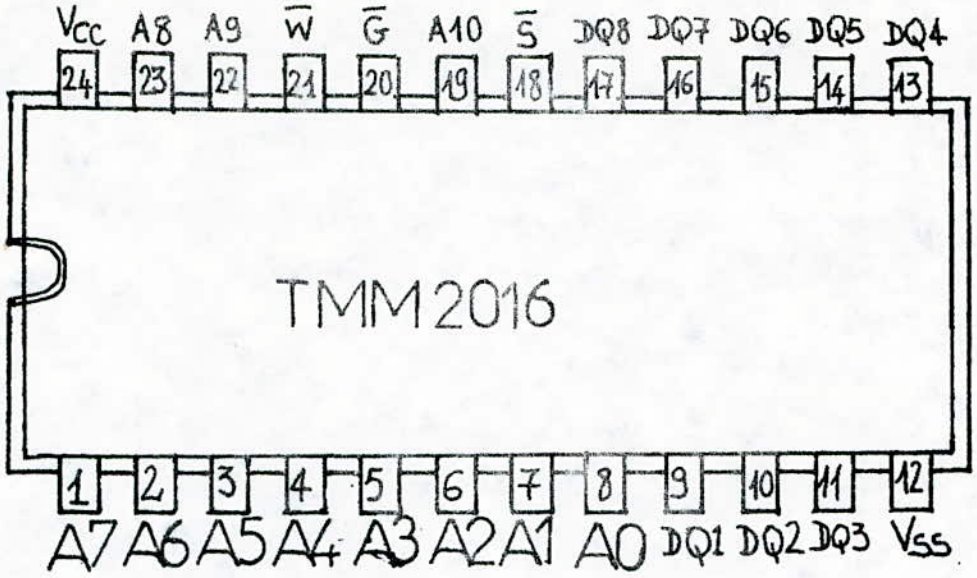

MC 6809
Vss 1
NMI 2
IRQ 3
FIRQ 4
BS 5
BA 6
Vcc 7
A0 8
A1 9
A2 10
A3 11
A4 12
A5 13
A6 14
A7 15
A8 16
A9 17
A10 18
A11 19
A12 20
40 HALT
39 XTAL
38 EXTAL
37 RESET
36 MRDY
35 Q
34 E
33 DMA/BREQ
32 R/W
31 D0
30 D1
29 D2
28 D3
27 D4
26 D5
25 D6
24 D7
23 A15
22 A14
21 A13


PIA MC6821
GND 1
PA0 2
PA1 3
PA2 4
PA3 5
PA4 6
PA5 7
PA6 8
PA7 9
PB0 10
PB1 11
PB2 12
PB3 13
PB4 14
PB5 15
PB6 16
PB7 17
CB1 18
CB2 19
+5V 20
40 CA1
39 CA2
38 IRQA
37 IRQB
36 RS0
35 RS1
34 RESET
33 D0
32 D1
31 D2
30 D3
29 D4
28 D5
27 D6
26 D7
25 E
24 CS1
23 CS2
22 CS0
21 R/W


MC 6875
X1 1
X2 2
Ext. In. 3
4×fo 4
2×fo 5
mémoire prête 6
Bus φ2 7
Masse 8
16 Vcc
15 MPU φ1
14 Sortie Reset
13 MPU φ2
12 Power ou Reset
11 DMA / Ref Grant
10 DMA / Ref Req
9 Horloge mémoire


SORTIES
Vcc Y0 Y1 Y2 Y3 Y4 Y5 Y6
16 15 14 13 12 11 10 9
Y0 Y1 Y2 Y3 Y4 Y5
74LS138
A Y6
B C G2A G2B G1 Y7
1 2 3 4 5 6 7 8
A B C G2A G2B G1 Y7 GND
sélect
valid
Sortie


Vcc
ENTREES
SORTIES
24 23 22 21 20 19 18 17 16 15 14 13
A B C D G2 G1 15 14 13 12
0
74 LS154
11
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10 11 12
SORTIES
GND


Vcc A8 A9 W̄ Ḡ A10 S̄ DQ8 DQ7 DQ6 DQ5 DQ4
24 23 22 21 20 19 18 17 16 15 14 13
TMM2016
1 2 3 4 5 6 7 8 9 10 11 12
A7 A6 A5 A4 A3 A2 A1 A0 DQ1 DQ2 DQ3 Vss

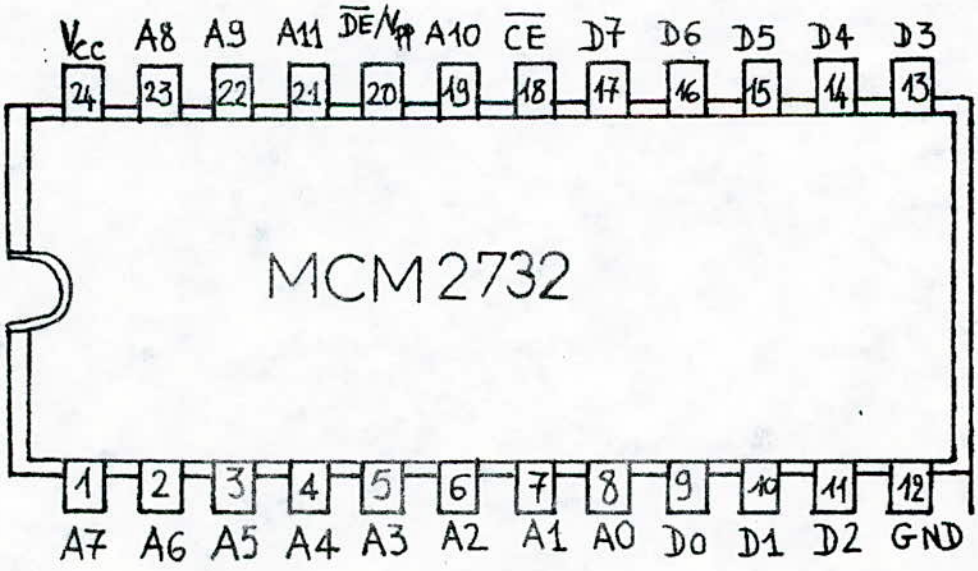

Vcc A8 A9 A11 D̄E̅/V̄p̄p̄ A10 C̄E̅ D7 D6 D5 D4 D3
24 23 22 21 20 19 18 17 16 15 14 13
MCM2732
1 2 3 4 5 6 7 8 9 10 11 12
A7 A6 A5 A4 A3 A2 A1 A0 Do D1 D2 GND

FIGURE 1 – READ DATA FROM MEMORY OR PERIPHERALS

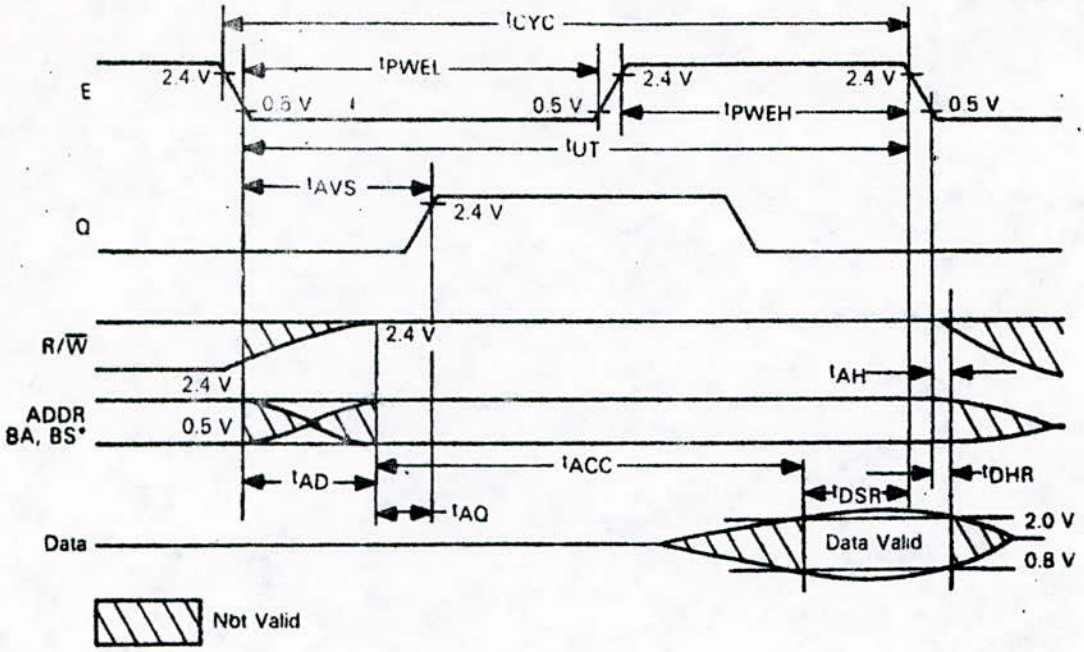
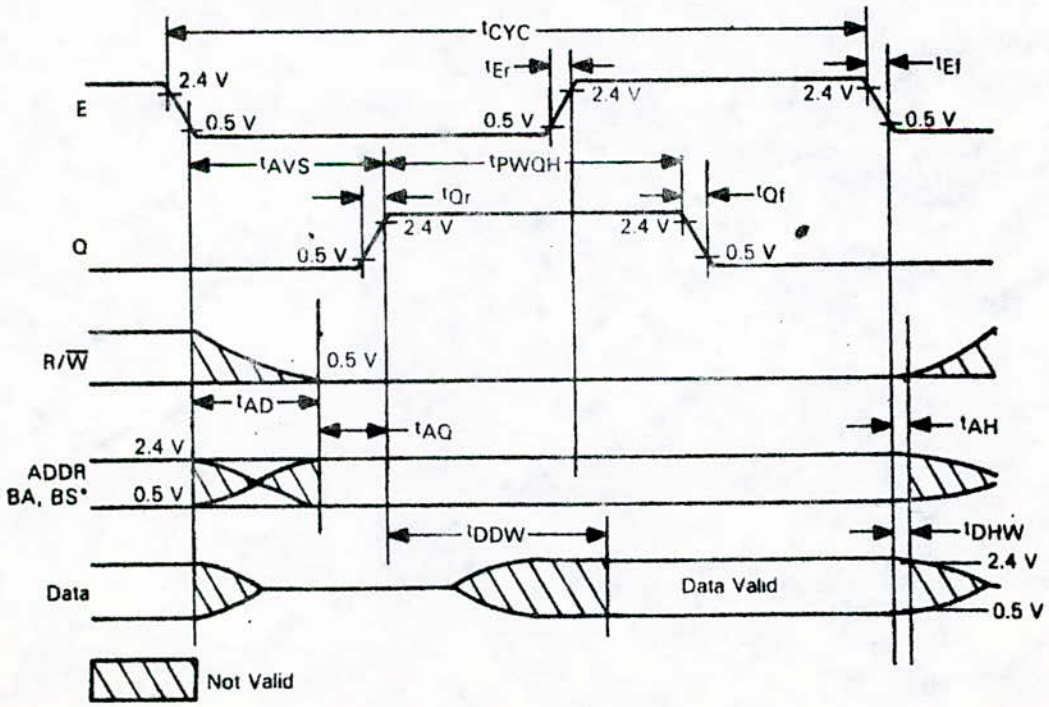
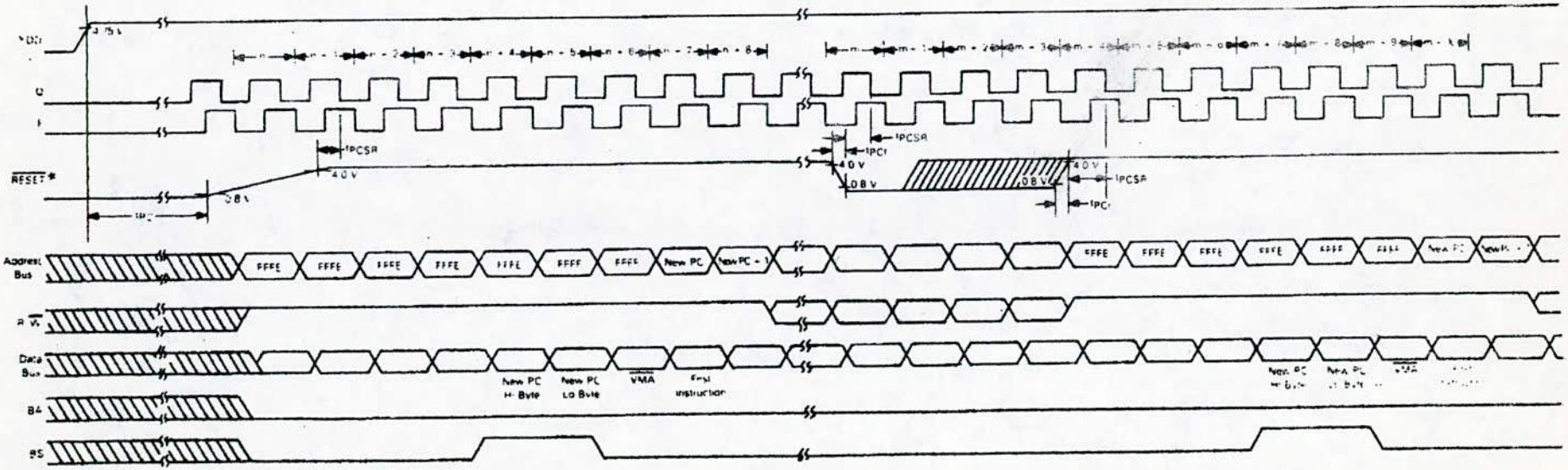


FIGURE 2 – WRITE DATA TO MEMORY OR PERIPHERALS



*Hold time for BA, BS not specified

FIGURE 7 — $\overline{\text{RESET}}$ TIMING



*Note: Parts with data codes prefixed by 7F will come out of RESET one cycle sooner than shown.

FIGURE 10 — $\overline{IRQ}$ AND $\overline{NMI}$ INTERRUPT TIMING

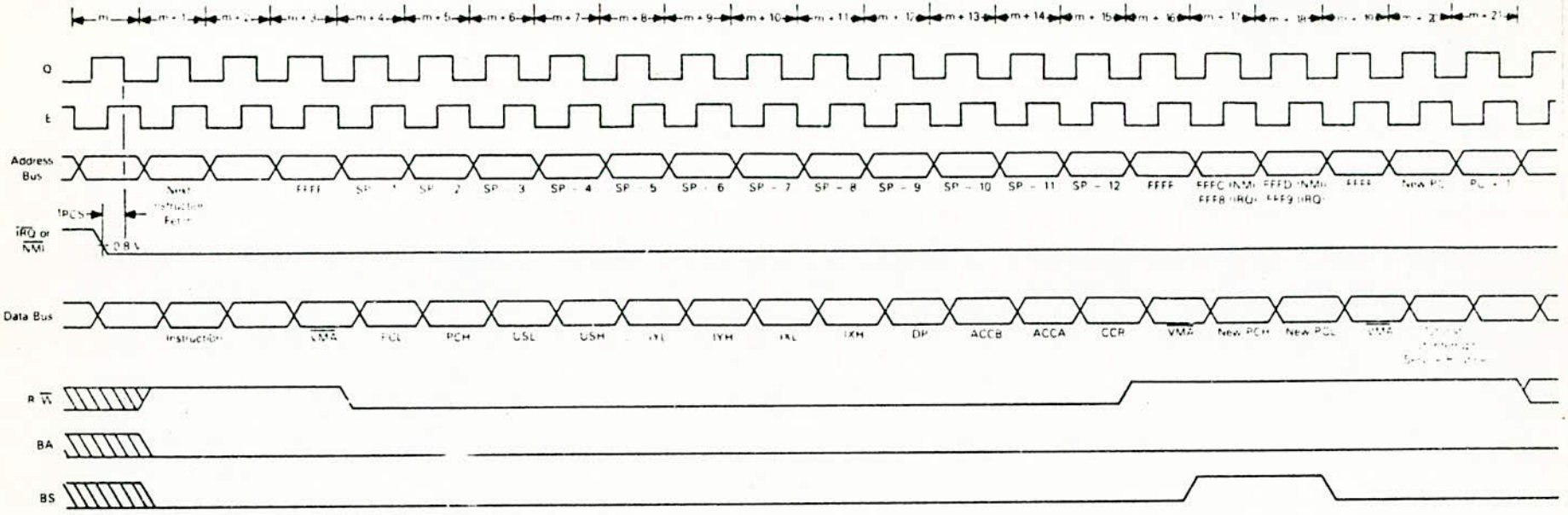


FIGURE 11 — $\overline{FIRQ}$ INTERRUPT TIMING

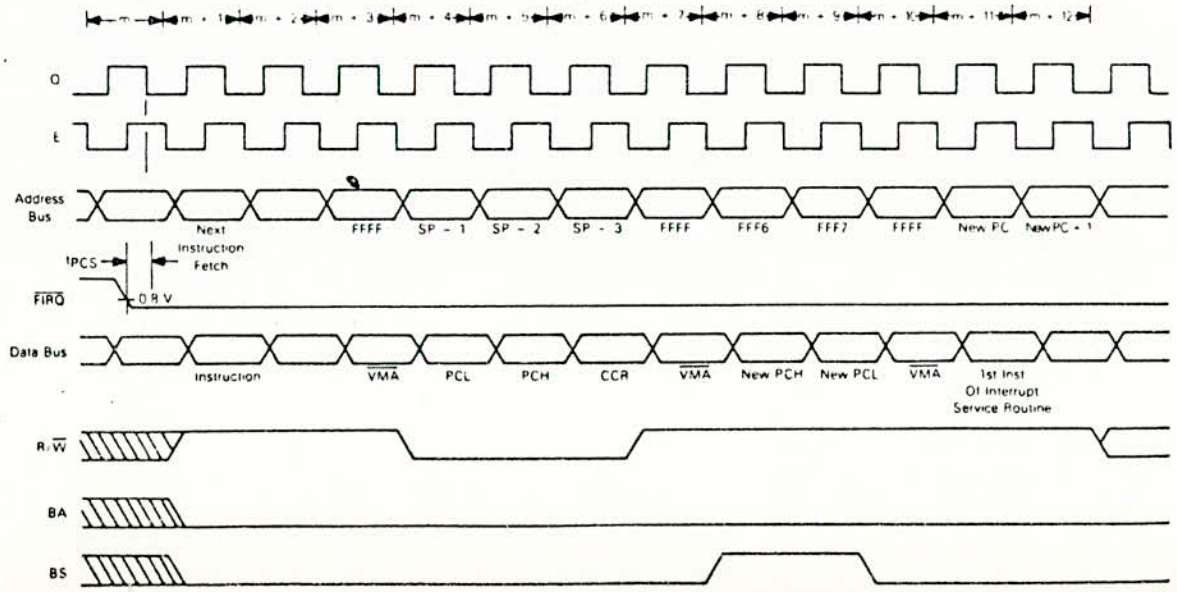


TABLE 4 — 8-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumulator or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

NOTE: A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

TABLE 5 — 16-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

NOTE: D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

TABLE 6 — INDEX REGISTER/STACK POINTER INSTRUCTIONS

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from hardware stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

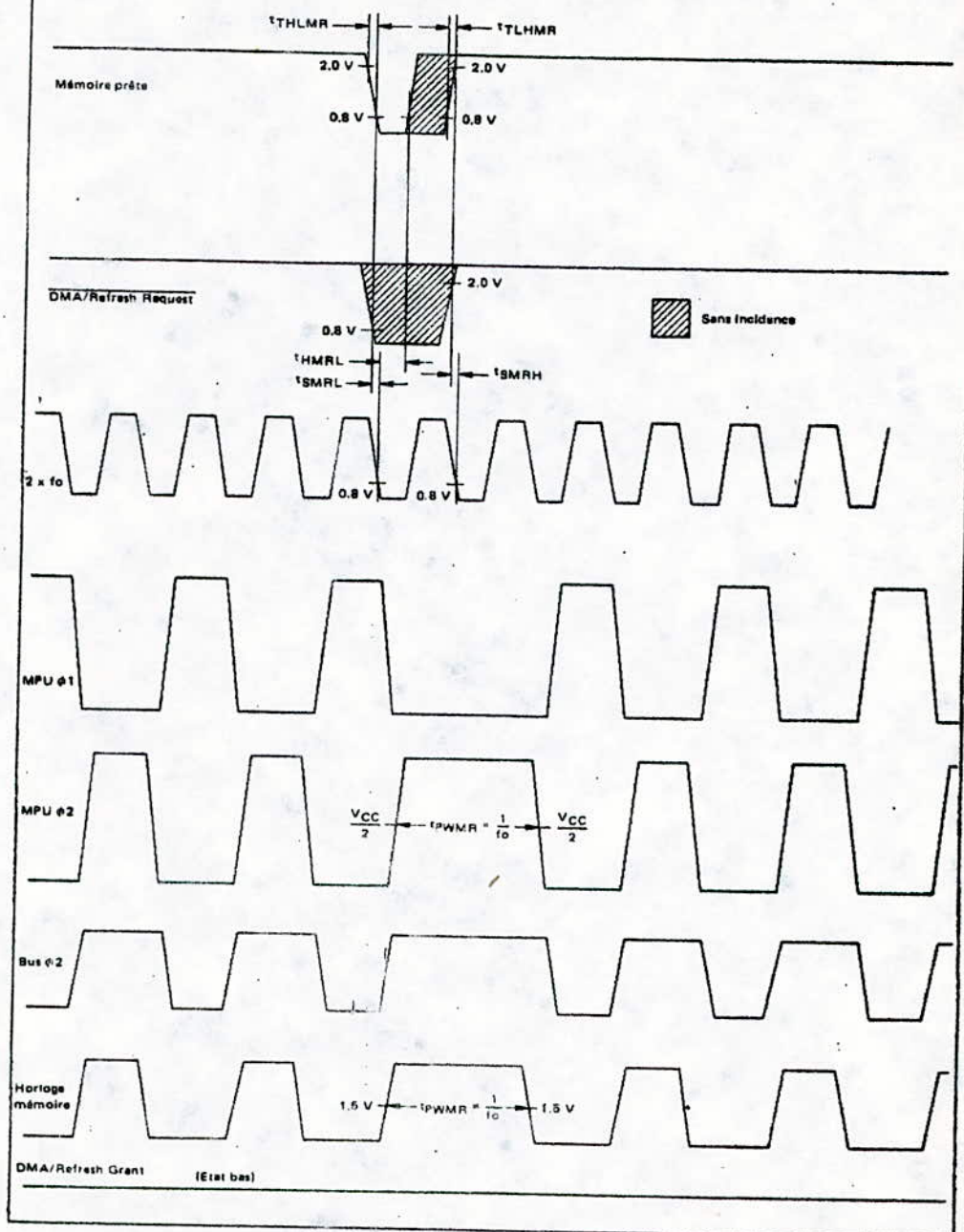
TABLE 7 — BRANCH INSTRUCTIONS

Mnemonic(s)	Operation
	SIMPLE BRANCHES
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
	SIGNED BRANCHES
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ; LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
	UNSIGNED BRANCHES
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	OTHER BRANCHES
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

TABLE 8 — MISCELLANEOUS INSTRUCTIONS

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

FIGURE 5 – DIAGRAMME DES TEMPS EN FONCTIONNEMENT RALENTI
(UTILISATION D'UNE MÉMOIRE LENTE)
(La figure correspond à l'allongement minimum des temps)
Tension d'entrée : 3,0 à 0 V, $t_{THLMR} = t_{TLHMR} = 5{,}0$ ns



COMMAND SUMMARY

Command Code								Command Mnemonic	Command Description
7	6	5	4	3	2	1	0		
FIXED-POINT 16-BIT									
sr	1	1	0	1	1	0	0	SADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	1	1	0	1	1	0	1	SSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	1	1	0	1	1	1	0	SMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.
sr	1	1	1	0	1	1	0	SMUU	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.
sr	1	1	0	1	1	1	1	SDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
FIXED-POINT 32-BIT									
sr	0	1	0	1	1	0	0	DADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	0	1	0	1	1	0	1	DSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	0	1	0	1	1	1	0	DMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.
sr	0	1	1	0	1	1	0	DMUU	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.
sr	0	1	0	1	1	1	1	DDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
FLOATING-POINT 32-BIT									
sr	0	0	1	0	0	0	0	FADD	Add TOS to NOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	0	1	FSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	1	0	FMUL	Multiply NOS by TOS. Result to NOS. Pop Stack.
sr	0	0	1	0	0	1	1	FDIV	Divide NOS by TOS. Result to NOS. Pop Stack.
DERIVED FLOATING-POINT FUNCTIONS									
sr	0	0	0	0	0	0	1	SQRT	Square Root of TOS. Result in TOS.
sr	0	0	0	0	0	1	0	SIN	Sine of TOS. Result in TOS.
sr	0	0	0	0	0	1	1	COS	Cosine of TOS. Result in TOS.
sr	0	0	0	0	1	0	0	TAN	Tangent of TOS. Result in TOS.
sr	0	0	0	0	1	0	1	ASIN	Inverse Sine of TOS. Result in TOS.
sr	0	0	0	0	1	1	0	ACOS	Inverse Cosine of TOS. Result in TOS.
sr	0	0	0	0	1	1	1	ATAN	Inverse Tangent of TOS. Result in TOS.
sr	0	0	0	1	0	0	0	LOG	Common Logarithm (base 10) of TOS. Result in TOS.
sr	0	0	0	1	0	0	1	LN	Natural Logarithm (base e) of TOS. Result in TOS.
sr	0	0	0	1	0	1	0	EXP	Exponential (e^X) of TOS. Result in TOS.
sr	0	0	0	1	0	1	1	PWR	NOS raised to the power in TOS. Result in NOS. Pop Stack.
DATA MANIPULATION COMMANDS									
sr	0	0	0	0	0	0	0	NOP	No Operation
sr	0	0	1	1	1	1	1	FIXS	Convert TOS from floating point to 16-bit fixed point format.
sr	0	0	1	1	1	1	0	FIXD	Convert TOS from floating point to 32-bit fixed point format.
sr	0	0	1	1	1	0	1	FLTS	Convert TOS from 16-bit fixed point to floating point format.
sr	0	0	1	1	1	0	0	FLTD	Convert TOS from 32-bit fixed point to floating point format.
sr	1	1	1	0	1	0	0	CHSS	Change sign of 16-bit fixed point operand on TOS.
sr	0	1	1	0	1	0	0	CHSD	Change sign of 32-bit fixed point operand on TOS.
sr	0	0	1	0	1	0	1	CHSF	Change sign of floating point operand on TOS.
sr	1	1	1	0	1	1	1	PTOS	Push 16-bit fixed point operand on TOS to NOS (Copy)
sr	0	1	1	0	1	1	1	PTOD	Push 32-bit fixed point operand on TOS to NOS. (Copy)
sr	0	0	1	0	1	1	1	PTOF	Push floating point operand on TOS to NOS. (Copy)
sr	1	1	1	1	0	0	0	POPS	Pop 16-bit fixed point operand from TOS. NOS becomes TOS.
sr	0	1	1	1	0	0	0	POPD	Pop 32-bit fixed point operand from TOS. NOS becomes TOS.
sr	0	0	1	1	0	0	0	POPF	Pop floating point operand from TOS. NOS becomes TOS.
sr	1	1	1	1	0	0	1	XCHS	Exchange 16-bit fixed point operands TOS and NOS.
sr	0	1	1	1	0	0	1	XCHD	Exchange 32-bit fixed point operands TOS and NOS.
sr	0	0	1	1	0	0	1	XCHF	Exchange floating point operands TOS and NOS.
sr	0	0	1	1	0	1	0	PUPI	Push floating point constant "π" onto TOS. Previous TOS becomes NOS.

NOTES:

1. TOS means Top of Stack. NOS means Next on Stack.
2. AMD Application Brief "Algorithm Details for the Am9511A APU" provides detailed descriptions of each command function, including data ranges, accuracies, stack configurations, etc.
3. Many commands destroy one stack location (bottom of stack) during development of the result. The derived functions may destroy several stack locations. See Application Brief for details.
4. The trigonometric functions handle angles in radians, not degrees.
5. No remainder is available for the fixed-point divide functions.
6. Results will be undefined for any combination of command coding bits not specified in this table.

Parameters	Description		Am9511A Min	Am9511A Max	Am9511A-1 Min	Am9511A-1 Max	Am9511A-4 Min	Am9511A-4 Max	Units
TAPW	$\overline{EACK}$ LOW Pulse Width		100		75		50		ns
TCDR	C/$\overline{D}$ to $\overline{RD}$ LOW Set-up Time		0		0		0		ns
TCDW	C/$\overline{D}$ to $\overline{WR}$ LOW Set-up Time		0		0		0		ns
TCPH	Clock Pulse HIGH Width		200		140		100		ns
TCPL	Clock Pulse LOW Width		240		160		120		ns
TCSR	$\overline{CS}$ LOW to $\overline{RD}$ LOW Set-up Time		0		0		0		ns
TCSW	$\overline{CS}$ LOW to $\overline{WR}$ LOW Set-up Time		0		0		0		ns
TCY	Clock Period		480	5000	320	3300	250	2500	ns
TDW	Data Bus Stable to $\overline{WR}$ HIGH Set-up Time		150		100 (Note 9)		100		ns
TEAE	$\overline{EACK}$ LOW to $\overline{END}$ HIGH Delay			200		175		150	ns
TEPW	$\overline{END}$ LOW Pulse Width (Note 4)		400		300		200		ns
TOP	Data Bus Output Valid to $\overline{PAUSE}$ HIGH Delay		0		0		0		ns
TPPWR	$\overline{PAUSE}$ LOW Pulse Width Read (Note 5)	Data	3.5TCY+50	5.5TCY+300	3.5TCY+50	5.5TCY+200	3.5TCY+50	5.5TCY+200	ns
		Status	1.5TCY+50	3.5TCY+300	1.5TCY+50	3.5TCY+200	1.5TCY+50	3.5TCY+200	
TPPWW	$\overline{PAUSE}$ LOW Pulse Width Write (Note 8)			50		50		50	ns
TPR	$\overline{PAUSE}$ HIGH to $\overline{RD}$ HIGH Hold Time		0		0		0		ns
TPW	$\overline{PAUSE}$ HIGH to $\overline{WR}$ HIGH Hold Time		0		0		0		ns
TRCD	$\overline{RD}$ HIGH to C/$\overline{D}$ Hold Time		0		0		0		ns
TRCS	$\overline{RD}$ HIGH to $\overline{CS}$ HIGH Hold Time		0		0		0		ns
TRO	$\overline{RD}$ LOW to Data Bus ON Delay		50		50		25		ns
TRP	$\overline{RD}$ LOW to $\overline{PAUSE}$ LOW Delay (Note 6)			150		100 (Note 9)		100	ns
TRZ	$\overline{RD}$ HIGH to Data Bus OFF Delay		50	200	50	150	25	100	ns
TSAPW	$\overline{SVACK}$ LOW Pulse Width		100		75		50		ns
TSAR	$\overline{SVACK}$ LOW to SVREQ LOW Delay			300		200		150	ns
TWCD	$\overline{WR}$ HIGH to C/$\overline{D}$ Hold Time		60		30		30		ns
TWCS	$\overline{WR}$ HIGH to $\overline{CS}$ HIGH Hold Time		60		30		30		ns
TWD	$\overline{WR}$ HIGH to Data Bus Hold Time		20		20		20		ns
TWI	Write Inactive Time	Command	3TCY		3TCY		3TCY		ns
		Data	4TCY		4TCY		4TCY		
TWP	$\overline{WR}$ LOW to $\overline{PAUSE}$ LOW Delay (Note 6)			150		100 (Note 9)		100	ns

Notes:

1. Typical values are for T_A = 25°C, nominal supply voltages and nominal processing parameters.
2. Switching parameters are listed in alphabetical order.
3. Test conditions assume transition times of 20ns or less, output loading of one TTL gate plus 100pF and timing reference levels of 0.8V and 2.0V.
4. $\overline{END}$ low pulse width is specified for $\overline{EACK}$ tied to VSS. Otherwise TEAE applies.
5. Minimum values shown assume no previously entered command is being executed for the data access. If a previously entered command is being executed, $\overline{PAUSE}$ LOW Pulse Width is the time to complete execution plus the time shown. Status may be read at any time without exceeding the time shown.
6. $\overline{PAUSE}$ is pulled low for both command and data operations.
7. TEX is the execution time of the current command (see the Command Execution Times table).
8. $\overline{PAUSE}$ low pulse width is less than 50ns when writing into the data port or the control port as long as the duty requirement (TWI) is observed and no previous command is being executed. TWI may be safely violated up to 500ns as long as the extended TPPWW that results is observed. If a previously entered command is being executed, $\overline{PAUSE}$ LOW Pulse Width is the time to complete execution plus the time shown.
9. 150ns for the Am9511A-1DM.

1) subroutine INIT:

```
0094   4F          CLRA
0095   5F          CLRB
0096   B7 EBF8     STAA  $ EBF8
0099   5C          INCB
009A   C1 10       CMPB ##$ 10
009C   2F F8       BLE  % F8
009E   B7 EBF9     STAA $ EBF9
00A1   39          RTS
```

2) subroutine DATA:

```
02B5   C6 00       LDA B ## 00
02B7   A6 00       LDA A    0,X
02B9   B7 EBF8     STAA     EBF8
02BC   5C          INC B
02BD   C1 04       CMPB ##04
02BF   27 04       BEQ  04
02C1   08          INX
02C2   7E 02B7     JMP $ 02B7
02C5   39          RTS
```

3) subroutine CONVERSION

```
02C6   86 9F       LDA A ## 9F   *FLTS=9F
02C8   B7 EBF9     STA A    EBF9
02CB   39          RTS
```

4) subroutine RANJ

```
02CC    6 EBF8     LDA A    EBF8
02CF   B6 EBF8     LDA A    EBF8
02D2   B7 0050     STA A    $0050
02D5   B6 EBF8     LDA A    EBF8
02D8   B7 0051     STA A    $0051
02DB   B6 EBF8     LDA A    EBF8
02DE   B7 0052     STA A    $0052
```

```
02E1   B6  EBF8        LDA A  $ EBF8
02E4   B7  0053        STA A  $ 0053
02E7   39              RTS
```

5) programme test:

```
0253   00              + chargement de la
0254   00              table de données.
0255   98
0256   06
0257   3F

02B2   CE  0253        LDX ## $ 0253
02B5   C6  00          LDA B ## 00
02B7   A6  00          LDA A    0,X
02B9   B7  EBF8        STA A  $ EBF8
02BC   5C              INCB
02BD   C1  04          CMPB ## 04
02BF   27  04          BEQ      04
02C1   08              INX
02C2   7E  02B7        JMP    $ 02B7
02C5   3F              SWI
02C6   86  9F          LDA A ## $ 9F
02C8   B7  EBF9        STA A  $ EBF9
02CB   3F              SWI
02CC   B6  EBF8        LDA A $ EBF8
02CF   B6  EBF8        LDA A $ EBF8
02D2   B7  0050        STA A $ 0050
02D5   B6  EBF8        LDA A $ EBF8
02D8   B7  0051        STA A $ 0051
02DB   B6  EBF8        LDA A $ EBF8
02DE   B7  0052        STA A $ 0052
```

Exemple : COS 2II

7) Sous programme de COS

```
0253  00              Chargement de la table de donnée
0254  00
0255  80
0256  02
0257  3F
0258  BD  02B2   JSR $02B2     DATA
025B  3F           SWI
$0100  86  1A      LDAA += 1A          PUPI= 1A
0102  B7  EBF9     STAA $ EBF9
0105  3F           SWI
0106  86  12
0107  B7  EBF9     LDAA=$12            FMUL=12
010A  3F           SWI
010C  86  19       LDAA= $19           XCHF=19
010E  B7  EBF9     STAA  EBF9
011   3F           SWI
012   86  83       LDAA=$83            CO=83
0114  B7  EBF9     STAA  EBF9
0117  3F
0118  BD  02CC     JSR $02CC           SUBROU.RANJ
011B  3F           SWI
RESULTAT
0050/01
0051/80       EN FLOTTANT HEXA
0052/00            01  80  00  00  ----> 0.1000 .2^1
0053/00                en decimal ----> 1
```

Remarque:

on peut avoir le SIN , en changeant simplement la commande de COS par la commande SIN .

```
      DIMENSION IVECT(100)
      DIMENSION  TS(100),A(48),B(48)
      DATA IPLANC,IASTER,IPLUS/1H  ,1H*,1H+/
      READ (105,15)  ITER
      PRINT  200, ITER
      READ (105,15) NV
      READ (105,100) (TS(L),L=1,NV)
      PRINT 87
      PRINT  200,NV
      PRINT 300,(TS(L),L=1,NV)
      CALL  CALCUL (TS,A,B,NV)
      PRINT 204
204   FORMAT (20X,72(1H*),/,20X,1H*,'GRAPHE DE LA TRANSFORME
     *DE FOURIER DE F(X)=X(T)',/,20X,72(1H*),//)
      PRINT 2040
2040  FORMAT (1H1)
      DO 400 I=1,100
      IVECT(I)=IPLUS
400   CONTINUE
      WRITE (108,201) (IVECT(J),J=1,100)
201   FORMAT (///,10X,100(A1),/)
      DO   11   I=1,100
      IVECT(I)=IPLANC
11    CONTINUE
      DO 202 K=1,16
      IVECT(20)=IPLUS
      T=B(K)*B(K)+A(K)*A(K)
      T=10*(T**0.5)
      J=T+20
      IVECT(J)=IASTER
      WRITE (108,203) IVECT
203   FORMAT (10X,100(A1),/)
      IVECT(J)=IPLANC
202   CONTINUE
      STOP
15    FORMAT(I2)
87    FORMAT (1X,////)
100   FORMAT(16(F2))
200   FORMAT (10X,10(1H*),I3)
300   FORMAT(2((10X,16F2),/))
      END

      SUBROUTINE CALCUL (TS,A,B,NV)
      DIMENSION TS(1),A(1),B(1)
      PI=2.*3.14159265
      DO 70 K=1,NV
      KK=K-1
      A(1)=0.
      B(1)=0.
      DO  65  I=1,NV
      JJ=I-1
      ANG=PI*FLOAT(JJ)*FLOAT(KK)/FLOAT(NV)
      A(K)=A(K)+TS(I)*COS(ANG)
      B(K)=B(K)+TS(I)*SIN(ANG)
65    CONTINUE
      A(K)=A(K)
      B(K)=B(K)
70    PRINT 160,K,A(K),B(K)
160   FORMAT (10X,I3,2(5X,F5.2))
      RETURN
      END
```

Echelle :

0,4

COEFS. REELLE.	COEFS. IMAG.
8.00	.00
1.00	5.03
:00	.00
1.00	1.50
.00	.00
1.00	.67
.00	.00
1.00	.20
.00	.00
1.00	- .20
0:00	.00
1.00	-.67
.00	.00
1.00	-1.50
.00	.00
1.00	-5.03

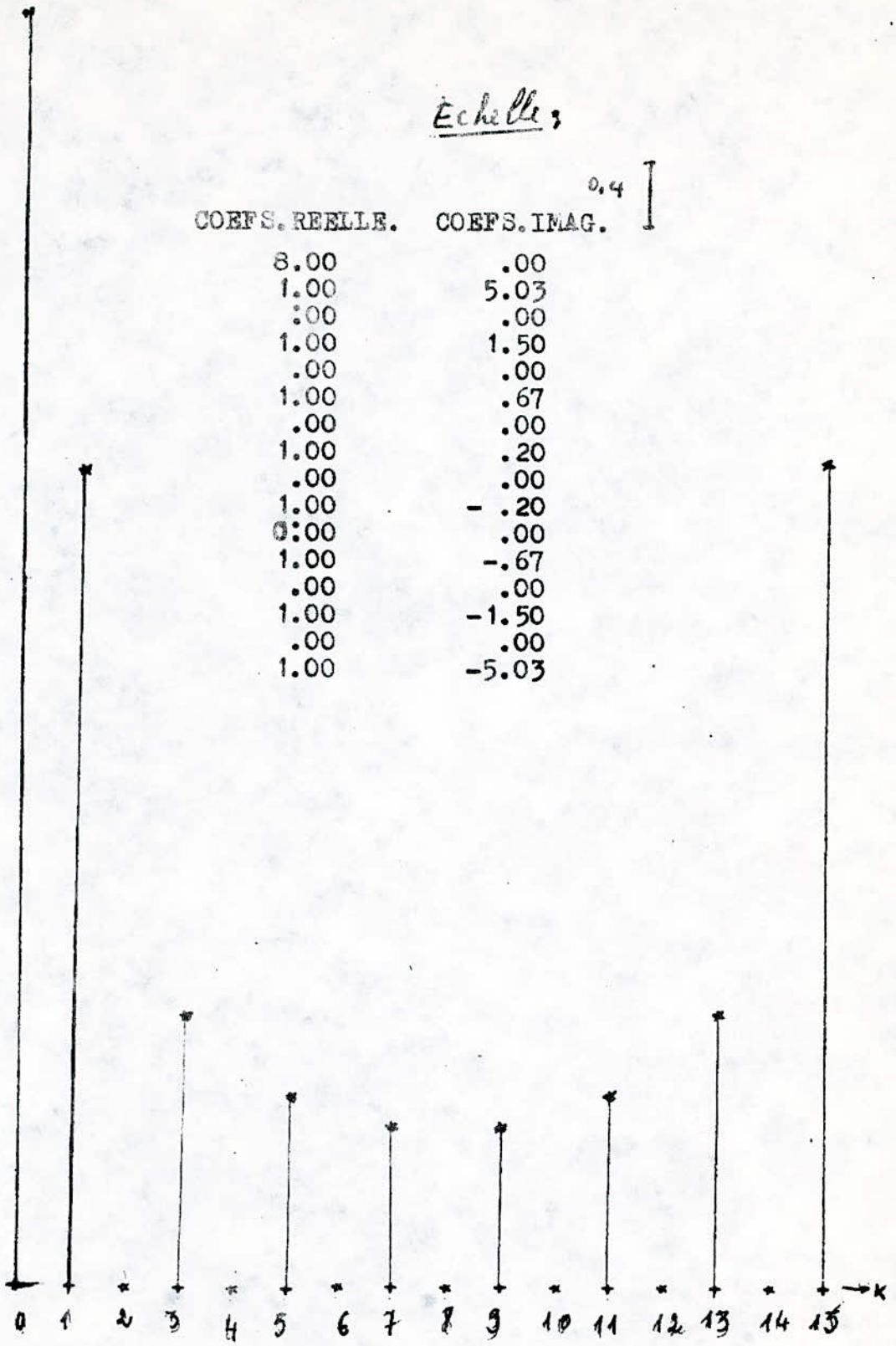


Fig. 1A

```
      DIMENSION [illegible](100)
      DIMENSION  AR(16),AI(16)
      DIMENSION X(16)
      INTEGER X
      DATA IBLANC,TASTER,IPLUS/1H  ,1H*,1H+/
      M=4
      N=2**M
      NU2=N/2
      NM1=N-1
      READ (105,11) (X(I),I=1,16)
11    FORMAT(16(I2))
      DO 04  K=1,16
      J=X(K)
      AR(K)=J
      AI(K)=0
04    CONTINUE
      I=0
      J=0
30    IF(I.GE.J) GO TO 40
      TR=AR(J+1)
      AR(J+1)=AR(I+1)
      AR(I+1)=TR
      TI=AI(J+1)
      AI(J+1)=AI(I+1)
      AI(I+1)=TI
40    K=NU2
50    IF(K.GT.J) GO TO 60
      J=J-K
      K=K/2
      GO TO 50
60     J=J+K
      I=I+1
      IF(I.LT.NM1)  GO TO 30
      J=J+K
      PI=3.14
      DO 100  L=1,M
      LE=2**L
      LE1=LE/2
      UI=0
      UR=1
      WR=COS(PI/LE1)
      WI=SIN(PI/LE1)
      DO 80  J=1,LE1
      DO 75  I=J,N,LE
      IP=I+LE1
      TR=AR(IP)*UR-AI(IP)*UI
      TI=AR(IP)*UI+AI(IP)*UR
      AR(IP)=AR(I)-TR
      AI(IP)=AI(I)-TI
      AR(I)=AR(I)+TR
      AI(I)=AI(J)+TI
75    CONTINUE
      UR=UR*WR-UI*WI
      UI=UR*WI+UI*WR
80    CONTINUE
100   CONTINUE
      WRITE (108,105)AR
105   FORMAT(20X,1H*,38X,'PARTIE REELLE DES COEFS DE FOURIER',
     *37X,1H*,/,20X,1H*,16(F5.2,2X),1H*,/)
```

```
      WRITE (108,110) AI
110   FORMAT(20X,1H*,20X,'PARTIE IMAGINAIRE DES COEFS DE FOURIER',
     *52X,1H*,/,20X,1H*,16(F5.2,2X),1H*,/)
      PRINT 32
32    FORMAT (20X,112(1H*),1H1)
      DO 200 I=1,100
      IVECT(I)=IPLUS
200   CONTINUE
      PRINT 2040
2040  FORMAT (1H1)
      PRINT 204
204   FORMAT (20X,72(1H*),/,20X,1H*,'GRAPHE DE LA TRANSFORME RAPIDE
     *DE FOURIER DE F(X)=X(T)',/,20X,72(1H*),//)
      WRITE (108,201) (IVECT(J),J=1,100)
201   FORMAT (///,10X,100(A1),/)
      DO 2021 I=1,100
      IVECT(I)=IPLANC
2021  CONTINUE
      DO 202 K=1,16
      IVECT(20)=IPLUS
      T=AI(K)*AI(K)+AR(K)*AR(K)
      T=10*(T**0.5)
      J=T+20
      IVECT(J)=IASTER
      WRITE (108,203) IVECT
203   FORMAT (10X,100(A1),/)
      IVECT(J)=IPLANC
202   CONTINUE
      STOP
      END
```

TABLEAU DES COEFFICIENTS DE FOURIER.

COEFS;DES REELLE;	COEFS. DES IMAGINAIRE
8.00	.00
2.93	2.03
.00	.00
.88	.88
.00	.00
.55	-.25
.00	.00
.64	.14
.00	.00
.48	-1.03
.00	.00
.68	-.54
.00	.00
.04	-.76
.00	.00
1.80	-.48

KMS. .

```
**************************************************************************
*GRAPHE DE LA TRANSFORME RAPIDE     DE FOURIER DE F(X)=X(T)
**************************************************************************
```

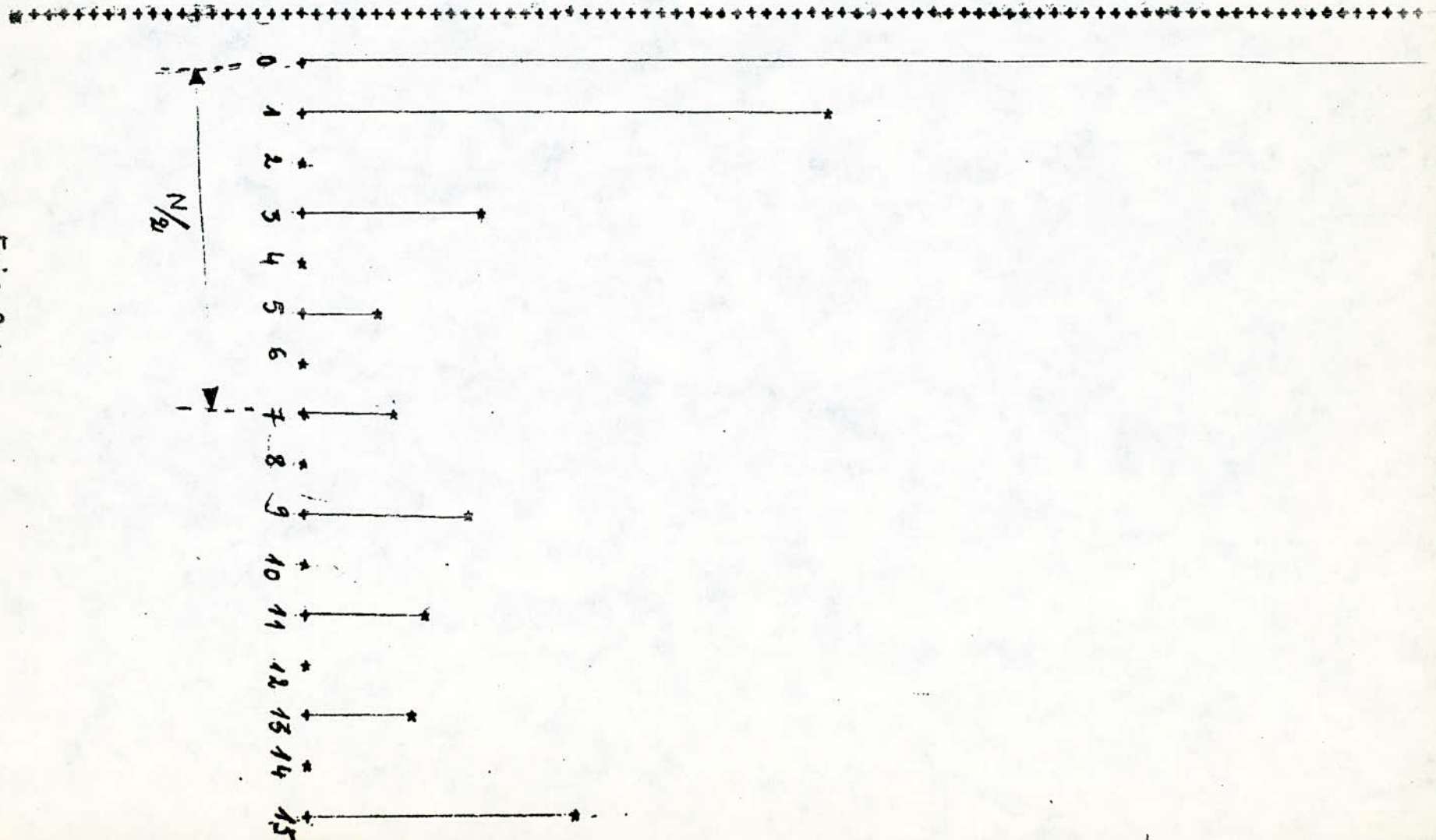


Fig. 8A

BIBLIOGRAPHIE.

THESES:

-Résolution de l'équation du bilan énérgetique
C.ERI janvier 82.

-Etude du up 6809 Juin 83

REVUES:

-Haut Parleur

-Minis et micro n°92 et 96

- Electronics

- IEE AUDIO 67

-Rapport de séminaire de A. ABDELLAOUI.
année 1977.

OUVRAGES:

-Emploie des microsystèmes (AUMIAUX)

-Les systèmes à up (AUMIAUX)

-Le up 6809 (DARDANE)

-Les méthodes rapides de transformation du signal. (LIFERMANN)

-The Fast Fourier Transform (BRIGHAM)

-Digital Signal Processing
(RONALD SHAFER.)

Fiche Technique fournie par A.M.D (de l'AM 9511)