9/96

REPUBLIQUE NATIONALE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE D'ALGER

DEPARTEMENT D'ELECTRONIQUE

PROJET DE FIN D'ETUDE

SUJET:

ETUDE & REALISATION D'UNE CARTE FILLE IEEE-488 POUR PC A BASE DU MC 68488

Proposé et dirigé par:
Mr R. SADOUN
Mme M. BEDDEK

Etudié par:
Mr A. OURAHMANE
Mr A. OUKEMOUM

PROMOTION: **Septembre 1996**

REPUBLIQUE NATIONALE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE D'ALGER

DEPARTEMENT D'ELECTRONIQUE

PROJET DE FIN D'ETUDE

SUJET:

ETUDE & REALISATION D'UNE CARTE FILLE IEEE-488 POUR PC A BASE DU MC 68488

Proposé et dirigé par:
Mr R. SADOUN
Mme M. BEDDEK

Etudié par:
Mr A. OURAHMANE
Mr A. OUKEMOUM

PROMOTION: **Septembre 1996**

DEDICACES

- *A mes parents,*
- *A mes frères et soeurs,*
- *A ma femme,*
- *A l'ensemble de mes amis,*
- *Et à tous mes professeurs.*

Amor

- *A mes parents,*
- *A mes frères et à ma soeur,*
- *A tous mes professeurs,*
- *A tous mes amis,*
- Et à la mémoire de mon chat TARZOO.

Ali

REMERCIEMENTS

- Nous tenons à remercier Mr. SAADOUN et Mme BEDDEK pour les connaissances dont ils nous ont fait bénificier et pour le suivi dont notre travail a fait l'objet de leur part.
Nous les remercions aussi pour leur soutien, notamment sur le plan théorique, pour les critiques constructives dont ils nous ont fait part et pour le support qu'ils nous ont apporté sur le plan matériel et en documentation.

- Ces remerciements s'adressent aussi à Mme HAMMAMI pour ses orientations, ses propositions et son apport en documentation.

- Nous tenons aussi à remercier toute l'équipe du labo n° 11 de notre département particulièrement Ahmed, Yazid, Nacerddine, Fayçal, Redha et Abdessamad pour leur participation active dans la création des bonnes conditions de travail en son sein.

- Enfin, nous tenons à remercier nos parents, nos frères et nos amis pour leur soutien moral et financier et en particulier Mourad et Omar.

- Que tous ceux qui ont contribué de près ou de loin à la réalisation de notre projet trouvent ici l'expression de notre profonde gratitude.

SOMMAIRE

CHAPITRE I: ETUDE DE LA NORME IEEE-488:

CHAPITRE I: ETUDE DE LA NORME IEEE-488:

L'incroyable développement de la micro-électronique a ouvert et créé des perspectives dans de très larges domaines. Celui de la mesure profite amplement de ces bouleversements technologiques. Aussi la création des liens de communication entre les appareils de mesure a pour objectif de:

* Pouvoir configurer et déclencher des mesures;

* Collecter les résultats et, éventuellement, reconfigurer les appareils en fonction des résultats obtenus;

* Traiter et stocker les mesures;

* Sortir les résultats sous la forme appropriée.

Tout ceci sans intervention de l'opérateur, mais en suivant des directives programmées dans un système informatique.

Cette manière de procéder possède de multiples avantages:

* La mesure est plus rapide (davantage de mesures, meilleur appréciation des caractéristiques du phénomène étudié...)

* La mesure est plus précise (correction d'une réponse non-linéaire, extraction des mesures douteuses ...)

* La fiabilité est accrue par l'élimination du facteur humain.

L'automatisation des chaines de mesure a conduit à l'élaboration de nombreuses normes dont la plus importante est le bus IEEE-488. L'objectif principal du bus IEEE est de standardiser la mesure programmable. L'intérêt de cette standardisation est de permettre de passer d'une application à une autre, d'un constructeur ou d'un appareil à l'autre, en gardant toujours la même structure centrale.

1)- HISTORIQUE ET APPELATIONS:

En 1972, l'IEC (International Electrotechnical Commitee) publie une norme établissant un principe d'interfaçage par bus entre appareils de mesure. En 1975, l'IEEE (Institute of Electrical and Electronics Engineers) publie la norme IEEE-488-75 définissant une interface pour les instruments programmables. Cette norme est adoptée depuis 1978 par l'IEC (norme IEC-625-1) et par l'ANSI (American National Standards Institute). Elle fait alors l'objet d'une interface appelée **bus IEEE-488** ou encore **GPIB** (General Purpose Instrumentation Bus). Elle a été, en fait, élaborée à partir des propositions de la société Hewlett-Packard, c'est la raison pour laquelle on l'appelle aussi **HPIB** (Hewlett Packard Instrumentation Bus).

A présent les références sont les normes IEEE-488-1978 et IEC-625 qui diffèrent uniquement par les caractéristiques mécaniques du connecteur: IEEE-488 explicite son propre connecteur à 24 contacts alors que IEC-625 utilise le connecteur de la liaison série RS 232 à 25 contacts.

2)- LIMITES DE LA NORME:

La norme IEEE-488 définit trois aspects:

- Mécanique : connecteur et câble.
- Electrique : niveau logique, temps de transition.
- Fonctionnel: protocole et gestion de bus.

Un quatrième point, l'aspect opérationnel, n'est pas défini par la norme. Cela signifie que le format des paramètres envoyés pour configurer un appareil et les résultats de mesure sont extérieurs à la norme.

3)- ORGANISATION GENERALE SUR LE BUS IEEE-488:

On définit sur le bus: Deux modes de fonctionnement:

* Mode de commande.
* Mode de transfert des informations.

Trois types de fonctions:

* Contrôleur.
* Parleur.
* Ecouteur.

a) **La fonction contrôleur:** Assurée souvent par un calculateur, elle consiste à gérer tous les transferts d'informations sur le bus.A travers une liste de commandes bien définie, le contrôleur doit:

- Définir quels appareils doivent participer au transfert des données et quelle fonction chacun d'eux doit accomplir: émetteur ou récepteur.
- Transmettre des instructions et des messages et recevoir des messages provenant des appareils réclamant des services.

Un système donné peut comprendre plusieurs contrôleurs, mais l'un d'entre eux est nécéssairement le contrôleur principal. Seul ce dernier peut activer des circuits de validation et d'invalidation de l'interface.

b) **La fonction parleur:** Elle est attribuée par le contrôleur à un appareil. Il émet alors ses informations sur le bus pour les écouteurs. Un seul parleur est actif à la fois sur le bus.

c) **La fonction écouteur:** Elle est adressée par le contrôleur, un appareil réceptionne alors les informations émises par le parleur. Plusieurs écouteurs peuvent être actifs simultanément sur le bus.

* Les deux modes de fonctionnement sont étroitement liés à ces trois fonctions:

a) **Le mode de commande:** Il définit le cadre dans lequel les informations vont s'échanger. Dans ce mode, le contrôleur est maître du bus. Tous les appareils sont sous ses ordres. Tous les messages transmis sous ce mode sont spécifiés par la norme: le contrôleur adresse les appareils pour leur assigner les fonctions de parleur, d'écouteur ou pour les tester en réponse à une demande de service.

b) **Le mode de transfert d'informations:** Il s'établit une fois qu'une configuration a été déclarée. Le parleur envoie ses informations vers les écouteurs. La fonction contrôleur s'éffface. Les informations sont des messages d'appareils, propres à chaque appareil.Elles ne font pas partie de la norme.

4)- LES ELEMENTS DE CONFIGURATION:

L'ensemble de base comprend:

* Un calculateur: permettant de gérer le bus, de configurer les appareils, de collecter les résultats, de les traiter et de les stocker.

* Des appareils de mesures: configurables soit par le face avant soit par l'intermédiaire du bus IEEE.

* Des unités périphériques: (disque souple, imprimante, table traçante...) pouvant à tout moment donner un compte rendu des mesures transmises par le bus IEEE.

La majorité des appareils possède les deux fonctions écouteur et parleur:

* La fonction écouteur est utilisée pour recevoir les directives de mesure (mode, calibrage, type de mesure...). Des appareils tels que: disque dur, imprimante... reçoivent grâce à cette fonction les résultats des mesures pour stockage ou impression...

* La fonction parleur sert à émettre les résultats des mesures faites.

Le calculateur, quant à lui, possède presque toujours les trois fonctions. Il est très souvent le seul organe intelligent sur le bus. Cette intelligence programmée organise tout le travail du bus en agissant sur les trois fonctions du calculateur:

* Par le contrôleur, elle commande le bus;

* Par le parleur, elle envoie aux différents appareils les directives de mesure;

* Par l'écouteur, elle reçoit les mesures pour les traiter et les centraliser ou elle espionne le bus pour intervenir par le contrôleur à la fin de chaque réception.

D'autres configurations sont possibles, notamment la configuration sans contrôleur. Cependant les appareils, dans ce cas-ci, doivent posséder les fonctions **ton** (talk only) e t **lon** (listen only). Ils communiquent alors entre eux grâce au bus de données et aux 3 lignes de handshake. Les autres lignes sont ignorées. Aucun message à distance n'est envoyé ni reconnu puisque le contrôleur est absent. Les appareils travaillent uniquement en mode de transfert sur le bus. Ce fonctionnement en ton/lon est initialisé par un interrupteur sur chaque instrument. Lorsqu'il est positionné, les rôles sont distribués et le bus est figé.

5)- STRUCTURE DU BUS IEEE-488:

Le bus IEEE-488 est constitué de l'ensemble des élements fonctionnels, électriques et mécaniques d'une interface conforme à la norme IEEE-488. C'est avant tout un câble de liaison comprenant 24 fils: 8 d'entre eux sont des masses et des blindages et les 16 autres sont répartis en 3 groupes:

* transfert de l'information (8 lignes);
* gestion de transfert des informations (3 lignes);
* gestion générale (5 lignes).

La structure du bus IEEE-488 est représentée par la figure I.1, et la structure fonctionnelle d'un instrument compatible avec le bus est illustrée à la figure I.2.

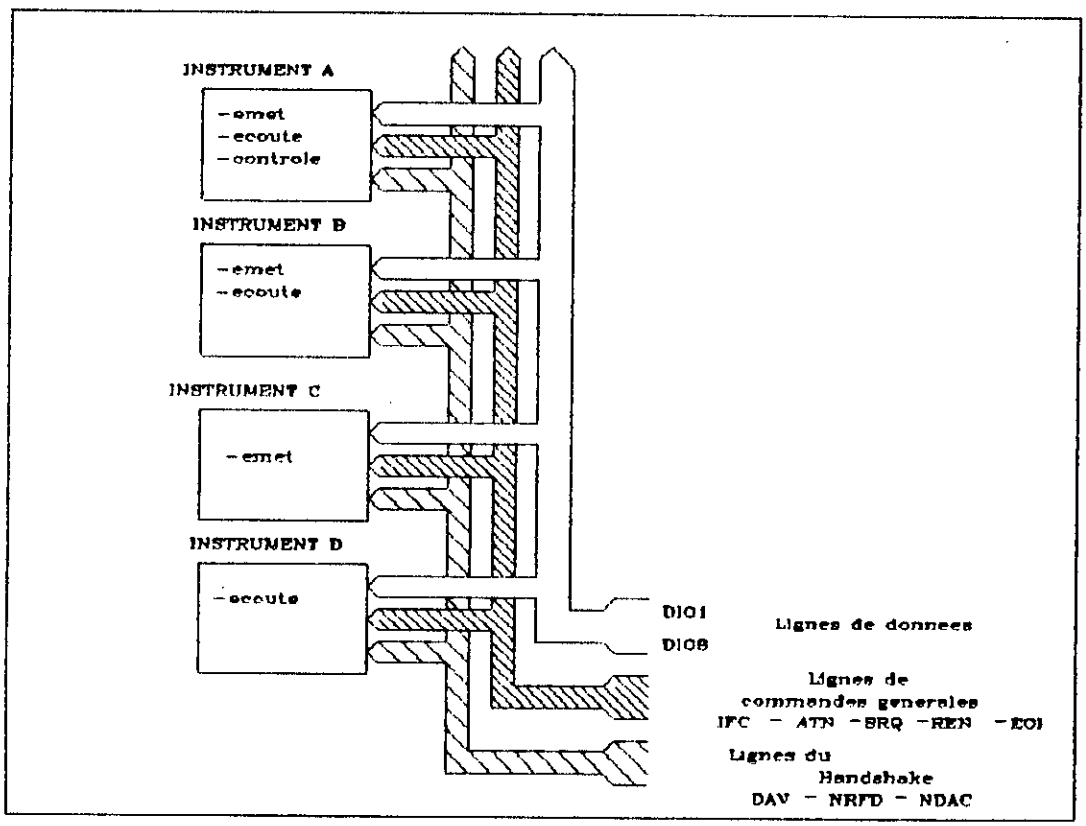


Fig.I.1: **Structure du bus IEEE 488**

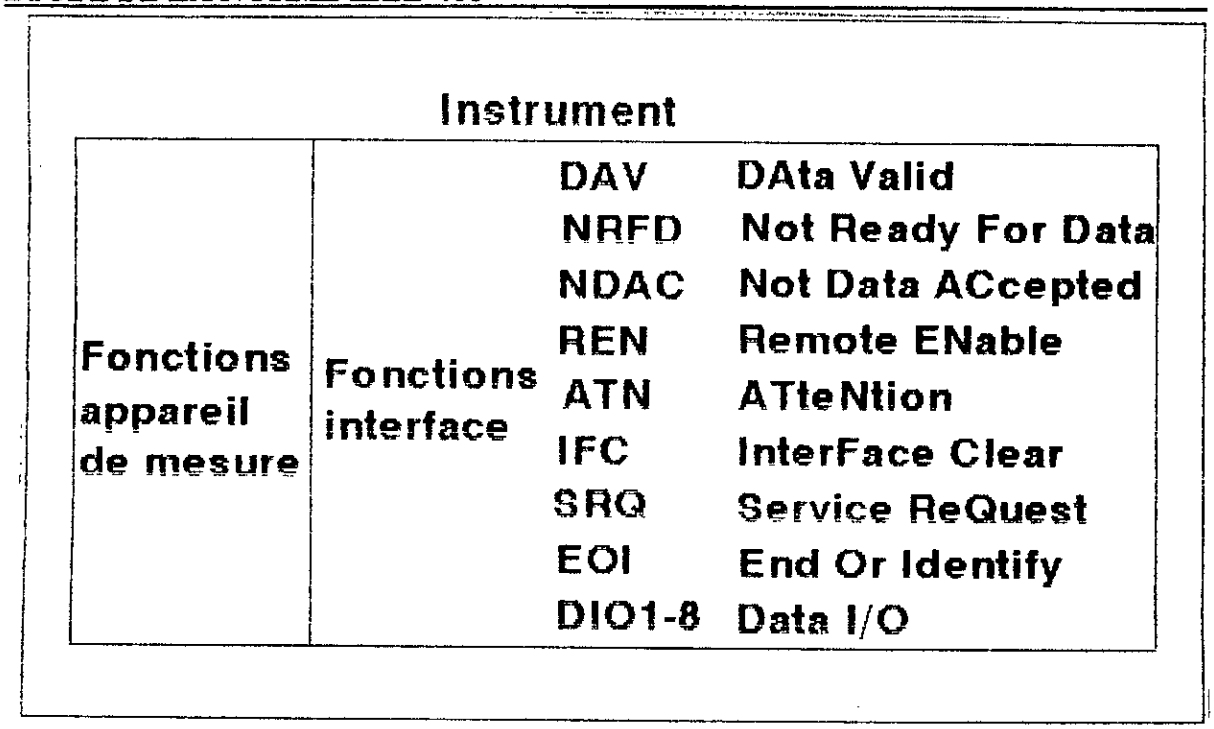


Fig.I.2: **Structure fonctionnelle d'un instrument compatible avec le bus IEEE-488**

***Lignes de transfert de données**: Ces 8 lignes (DIO1-DIO8) sont utilisées pour transmettre les informations entre les appareils sous forme d'octets de 8 bits mis en série. Les transmissions sont bidirectionnelles et du type asynchrone. Deux types de données circulent sur ces lignes suivant le mode de fonctionnement:

En mode de commande: ce sont des adressages ou des ordres de gestion. Ces messages sont définis par la norme et appelés messages à distance multilignes ou messages d'interface.

En mode de transfert: Ce sont des paramètres de mesure pour les appareils ou des résultats de mesure. Ces informations sont codées en ASCII.

***Lignes de gestion de transfert:** Trois lignes bidirectionnelles DAV, NRFD et NDAC sont destinées à gérer le transfert de chaque octet d'un appareil émetteur vers un ou plusieurs appareils récepteurs. Elles forment une poignée de main par laquelle passe l'information.

Les sigles mnémoniques formés par les initiales de leurs codes indiquent le rôle de chaque ligne:

-DAV (DAta Valid: donnée validée): Cette ligne informe les récepteurs que les données présentées sur le bus par un émetteur sont valides. Elle est commandée par l'émetteur (le contrôleur en mode de commande ou le parleur en mode de transfert)

- NRFD (Not Ready For Data: pas prêt pour accepter les données): Cette ligne est actionnée par le contrôleur en mode de commande et par les récepteurs en mode de transfert.

- NDAC(Not Data ACcepted: données non acceptées): Lorsque NDAC = 1, les données ne sont pas acceptées par le récepteur ou, en mode de commande, la commande n'est pas acceptée par l'appareil connecté au bus.

Ces 3 lignes assurent le protocole de handshake toutes les fois où des informations sont transmises sur les 8 lignes du bus de données.

Protocole de handshake:

Supposons que l'appareil source A doit envoyer une information à l'appareil accepteur B:

1- A doit attendre que B soit prêt à recevoir une donnée. Si B est disposé à recevoir, il envoie un message à A signifiant "prêt à recevoir" .

2- Dés qu'il envoie une donnée à B, A doit informer B que la donnée est disponible, le message de A vers B signifie alors "donnée disponible".

3- Lorsque B a reçu la donnée, il le signale à A. Le message de B vers A signifie alors "donnée acceptée".

Au cours de ces transferts, l'émetteur adapte sa propre vitesse sur celle du récepteur le plus lent. C'est pour cette raison qu'on qualifie ces transferts d'asynchrones.

***Lignes de gestion générale:** Chacune des cinq lignes de ce groupe a une fonction de commande spécifique entre le contrôleur et les autres appareils du système.Trois de ces lignes unidirectionnelles, sont sous contrôle exclusif du contrôleur (REN, ATN et IFC). La quatrième, unidirectionnelle aussi, peut être utilisée par n'importe quel appareil (SRQ). La cinquième, bidirectionnelle, est à la disposition soit du contrôleur soit du parleur (EOI).

ATN (ATteNtion): Permet au contrôleur d'indiquer aux instruments que des instructions et adresses (ATN à 1) ou des données (ATN à 0) sont sur le bus. Lorsque ATN est à 1, seuls l'émetteur et les récepteurs adressés sont concernés.

Tous les appareils doivent "regarder" à tout moment cette ligne et, dés qu'il y a un changement sur cette ligne, ils doivent répondre dans un delai de 200 ns.

IFC (InterFace Clear): Cette ligne peut être ramenée à 1 uniquement par le contrôleur afin de mettre les interfaces connectées au bus dans un état inactif. Toutes les opérations en cours sont alors arrêtées pour permettre de repartir d'une situation neutre et uniforme avant toute autre opération. IFC ne sert pas à réinitialiser l'appareil même; ceci est réalisée par la fonction d'interface DC.

SRQ **(Service ReQuest)**: Cette ligne est activée par tout appareil ayant un service à demander au contrôleur. Cette demande peut interrompre une opération en cours. Lorsque SRQ est demandé par plusieurs appareils au même moment, le contrôleur doit éffectuer une recherche pour déterminer les appareils en question et la nature du service demandé. Cette recherche peut s'éffectuer soit en série, soit en parallèle.

EOI **(End Or Identify)**: Si ATN = 0 (mode de transfert), cette ligne, activée par un émetteur, signifie que l'octet en cours est le dernier transmis pour l'opération.
Si ATN = 1 (mode de commande), c'est le contrôleur qui active la ligne EOI dans une recherche parallèle.

REN **(Remote ENable)**:Cette ligne est activée uniquement par le contrôleur pour commuter un appareil de la position "commande locale" à la position "commande à distance". Lorsque REN=0, l'appareil retourne en mode de commande locale.

6)- DOMAINE D'APPLICATION DU BUS IEEE-488:

Le bus est conçu pour travailler dans un contexte peu perturbé électriquement. Ses propriétés sont:

* 15 appareils au maximum;
* Longueur totale d'interconnexion 20 m;
* Reconfiguration simple par mise en parallèle des appareils;
* Transmission numérique asynchrone sur 8 bits;
* Vitesse maximum de transmission pour une ligne, 1 M bit/s.

7)- ASPECTS OPERATIONNELS DU BUS IEEE-488:

On doit distinguer deux types de messages:

* Les commandes issues du contrôleur et destinées à l'interface incorporée aux appareils: ce sont des messages interfaces.

* Les données destinées aux appareils de mesure afin de les placer dans un état particulier ou de recevoir le résultat découlant d'une action de mesure. Les données sont émises par un émetteur qui peut être le contrôleur: Ce sont des messages appareils.

Le schéma (fig.I.3) représente ces deux types de messages et délimite ainsi la zone d'intervention de la norme IEEE-488.

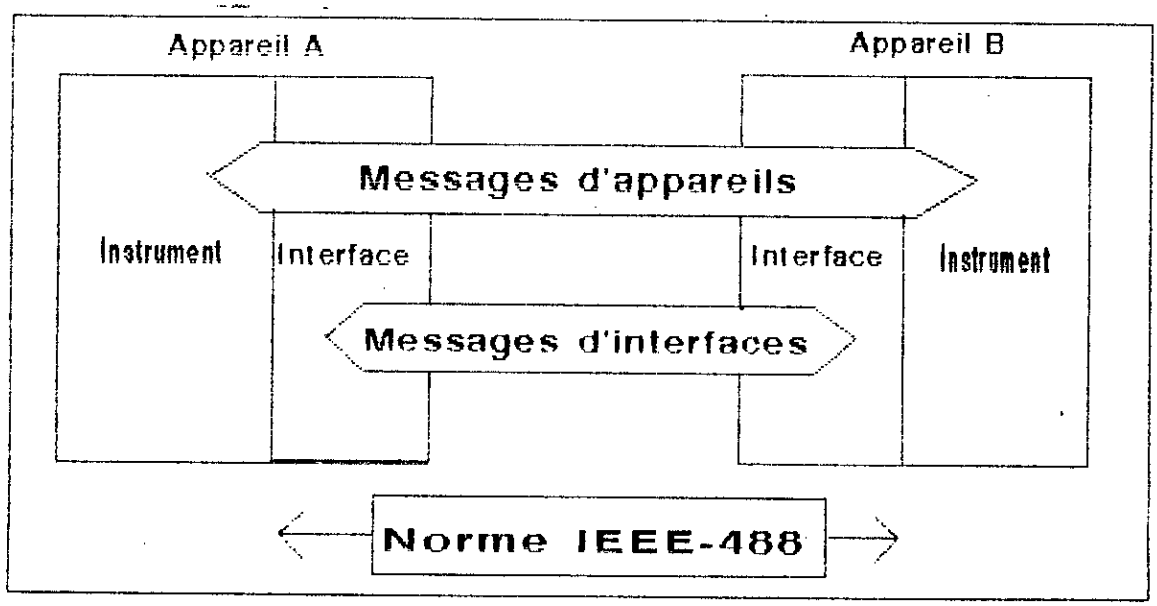


Fig.I.3: **Messages interfaces et messages appareils**

a- **Messages appareils:** Un message sur les huit lignes DIO est un message appareil (ou données) si la ligne ATN est simultanément à l'état faux (ATN = 0). Ces données sont émises par l'appareil adressé comme parleur et reçues par les appareils adressés comme écouteurs sous le contrôle du handshake. Elle peuvent être:

* Des données d'entrée:

- Données de commande (instructions de programme pour un appareil particulier).
- Données pour l'affichage ou le stockage.

* Des données de sortie:

- Données d'un résultat de mesure.
- Informations d'état de l'appareil.

Il doit exister entre les appareils en communication des conventions sur le codage des données. La norme définit seulement le mode de transfert des données mais non leur contenu. Ce dernier dépend de l'appareil utilisé.

Il semble ainsi que dans ce domaine, il peut y avoir des évolutions de normalisation dans l'avenir. A l'heure actuelle, la convention la plus répandue est le code ASCII.

En plus des messages appareils multifilaires que nous venons de définir, il faut ajouter deux autres messages appareils unifilaires: SRQ et EOI (déjà définis le paragraphe précédent).

b- Messages interfaces: On distingue, comme dans les messages appareils, les messages interfaces multifilaires et unifilaires. Les messages unifilaires (ATN, IFC et REN)qui ordonnent certaines fonctions aux appareils connectés sur le bus, ont été déjà définis précédemment. On a bien souligné qu'il s'agit de commandes émises par le contrôleur aux interfaces des appareils.

Un message multifilaire sur les lignes DIO est considéré comme une commande si la ligne ATN est simultanément à l'état vrai (ATN = 1). Il est codé par 7 bits sur 7 lignes (de DIO1 à DIO7).

* **Adressage des appareils :** Chaque appareil est identifié par une ou plusieurs adresses. C'est le contrôleur qui désigne la fonction de l'appareil en lui envoyant une notification spéciale au moyen de deux bits portés par les lignes DIO6 et DIO7:

ATN	DIO7	DIO6	L'appareil sélectionné doit:
1	0	1	recevoir des données (MLA)
1	1	0	émettre des données (MTA)

Le récepteur est noté MLA (My Listen Address), il correspond à *une valeur binaire adresse +32.* L'émetteur est noté MTA (My Talk Address), il correspond à *une valeur binaire adresse + 64.*

L'adresse proprement dite de l'appareil est codée sur 5 bits portés par les lignes DIO1 à DIO5. Elle est déterminée par 5 commutateurs (Fig.I. 4) positionnés par l'utilisateur: A_1, A_2, A_3, A_4 et A_5. Par conséquent l'adresse d'un appareil s'écrit alors: A $A_4A_3A_2A_1$.Cependant la valeur $A_5A_4A_3A_2A_1$ = 11111 n'est pas utilisable pour adresser un appareil, cette valeur particulière est réservée pour ramener les appareils à l'état passif.

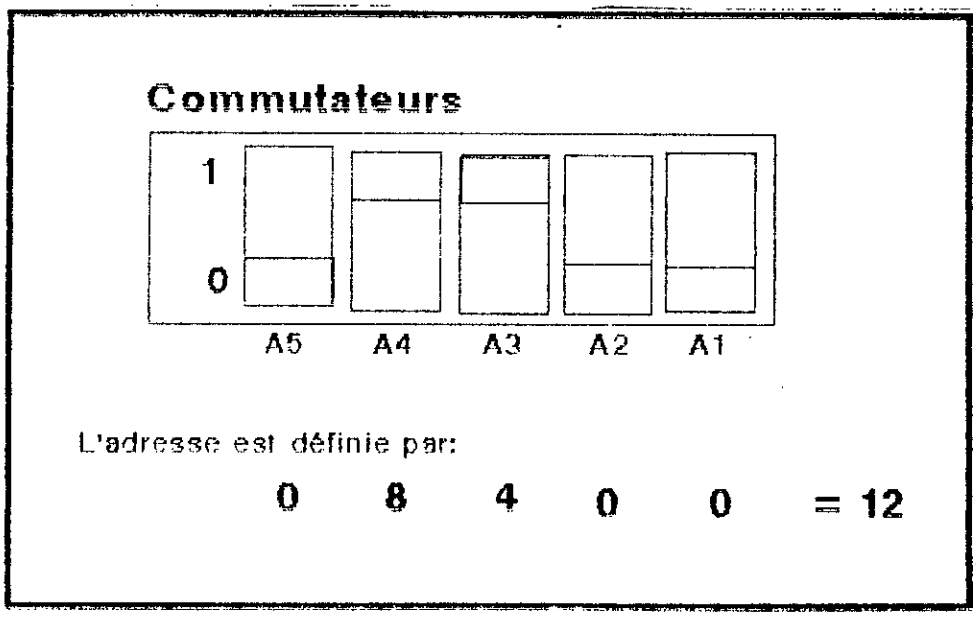


Fig.I.4: **Commutateurs internes définissant l'adresse d'un appareil**

Avec 5 bits, théoriquement on peut définir 32 adresses (0 à 31). Cependant comme nous venons de réserver l'adresse 31 au message de désadressage et que l'adresse 00000 n'est pas autorisée, il reste donc 30 adresses possibles. Ce nombre d'adresses est, dans la pratique, très largement supérieur au nombre d'appareils réèllement connectés sur le bus, car la vitesse de transfert maximale est garantie seulement pour un nombre d'appareils inférieur à 15.

Le tableau I.1 résume le mode d'adressage des appareils:

ATN	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	Fonction
1	X	0	1	A5	A4	A3	A2	A1	MLA(écouteur)
1	X	1	0	A5	A4	A3	A2	A1	MTA(parleur)
1	X	0	1	1	1	1	1	1	UNL
1	X	1	0	1	1	1	1	1	UNT

Tableau I.1: **Adressage des appareils**

* D'autres messages interfaces permettent de faire exécuter une tâche particuliere aux instruments, ce sont:

b-1- **Les commandes universelles:** qui concernent toutes les interfaces présentes sur le bus. Elles sont groupées dans le tableau I.2.

Symbole	Commande	ASCII	Décimal	octal	Hex
LLO	Local LOckout	DC1	17	21	11
DCL	Device CLear	DC4	20	24	14
PPU	Parallel Poll Unconfigure	NAK	21	25	15
SPE	Serial Poll Enable	CAN	24	30	18
SPD	Serial Poll Disable	EM	25	31	19
UNL	UNListen	3	63	77	3F
UNT	UNTalk	<--	95	137	5F

Tableau I.2: **Commandes universelles multifilaires**

* **PPU**: permet d'inhiber la recherche parallèle.

* **SPE:** établit l'état de recherche série. Les appareils émetteurs adressés doivent répondre en plaçant un mot d'état sur le bus. La ligne DIO7 permet de reconnaitre si l'appareil a agit sur la ligne SRQ du bus.

* **SPD:** permet de terminer l'état de recherche série.

* **DCL:** met les appareils dans l'état initial particulier propre à chacun.

* **LLO:** permet d'inhiber la possibilité de retour en mode commande locale de façon manuelle.

b-2- Les commandes adressées: qui ne concernent que les appareils adressés par cette action. Elles sont groupées dans le tableau I.3. Elles sont toutes multifilaires.

	Commande	ASCII	Décimal	octal	Hex
GTL	Go To Local	SOH	1	1	1
SDC	Select Device Clear	EOT	4	4	4
PPC	Parallel Poll Configure	ENQ	5	5	5
GET	Group Execute Trigger	BS	8	10	8
TCT	Take ConTrol	HT	9	11	9

Tableau I.3 : **Commandes adressées**

* **GET:** sert de synchronisation aux appareils adressés écouteurs pour permettre d'éxécuter ensemble une tâche programmée.

* **PPC:** permet de configurer les récepteurs pour que ceux-ci prennent en compte la recherche parallèle PPE.

* SDC: est la commande DCL destinée uniquement aux appareils sélectionnés écouteurs.

* GTL: ramène les appareils sélectionnés écouteurs à l'état de commande locale.

* TCT: confie le contrôle à un autre contrôleur préalablement adressé écouteur.

b-3- **Les commandes secondaires:** Elles sont émises à la suite d'une adresse (MLA ou MTA), d'une commande universelle ou d'une commande adressée; elles permettent d'étendre ces commandes.

Par exemple: l'adresse primaire est utilisée pour sélectionner un appareil (exemple: un multimètre); l'adresse secondaire est destinée à choisir la fonction (Voltmètre, Ampèrmètre, Ohmmètre...) ou une gamme de mesure (volt, millivolt...).

Sur le bus DIO, l'adresse secondaire est précédée de DIO6 = 0 et DIO7 =1.

ATN	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
1	X	1	1	A_5	A_4	A_3	A_2	A_1

Remarque: Les différentes commandes primaires et secondaires sont résumées dans le tableau I.4 avec leur code ISO.

Le groupe de commandes secondaires (SCG: Secondary Command Group) se trouve à la dernière colonne.

Le groupe de commandes primaires est constitué des:

* Commandes adressées (ACG: Addressed Command Group).

* Commandes universelles (UCG: Universal Command Group).

* Adresses des écouteurs (LAG: Listen Address Group).

* Adresses des parleurs (TAG: Talk Address Group).

b7 → b6 → b5 →	0 0 0			0 0 1			0 1 0			0 1 1			1 0 0			1 0 1			1 1 0			1 1 1		
ISO b4 b3 b2 b1 ↓ ↓ ↓ ↓	ISO 7 bits	Déci-mal	ATN = 1	ISO 7 bits	Déci-mal	ATN = 1	ISO 7 bits	Déci-mal	ATN = 1	ISO 7 bits	Déci-mal	ATN = 1	ISO 7 bits	Déci-mal	ATN = 1	ISO 7 bits	Déci-mal	ATN = 1	ISO 7 bits	Déci-mal	ATN = 1	ISO 7 bits	Déci-mal	ATN = 1
0 0 0 0	NUL	0		DLE	16		SP	32	0	0	48	16	@	64	0	P	80	16		96	0	p	112	16
0 0 0 1	SOH	1	GTL	DC1	17	LLO	!	33	1	1	49	17	A	65	1	Q	81	17	a	97	1	q	113	17
0 0 1 0	STX	2		DC2	18		"	34	2	2	50	18	B	66	2	R	82	18	b	98	2	r	114	18
0 0 1 1	ETX	3		DC3	19		#	35	3	3	51	19	C	67	3	S	83	19	c	99	3	s	115	19
0 1 0 0	EOT	4	SDC	DC4	20	DCL	$	36	4	4	52	20	D	68	4	T	84	20	d	100	4	t	116	20
0 1 0 1	ENQ	5	PPC	NAK	21	PPU	%	37	5	5	53	21	E	69	5	U	85	21	e	101	5	u	117	21
0 1 1 0	ACK	6		SYN	22		&	38	6	6	54	22	F	70	6	V	86	22	f	102	6	v	118	22
0 1 1 1	BEL	7		ETB	23		'	39	7	7	55	23	G	71	7	W	87	23	g	103	7	w	119	23
1 0 0 0	BS	8	GET	CAN	24	SPE	(	40	8	8	56	24	H	72	8	X	88	24	h	104	8	x	120	24
1 0 0 1	HT	9	TCT	EM	25	SPD	)	41	9	9	57	25	I	73	9	Y	89	25	i	105	9	y	121	25
1 0 1 0	LF	10		SUB	26		*	42	10	:	58	26	J	74	10	Z	90	26	j	106	10	z	122	26
1 0 1 1	VT	11		ESC	27		+	43	11	;	59	27	K	75	11	[	91	27	k	107	11	{	123	27
1 1 0 0	FF	12		FS	28		,	44	12	<	60	28	L	76	12	\	92	28	l	108	12	|	124	28
1 1 0 1	CR	13		GS	29		—	45	13	=	61	29	M	77	13	]	93	29	m	109	13	}	125	29
1 1 1 0	SO	14		RS	30		.	46	14	>	62	30	N	78	14	^	94	30	n	110	14	~	126	30
1 1 1 1	SI	15		US	31		/	47	15	?	63	UNT	O	79	15	_	95	UNT	o	111	15	DEL	127	31
	Addressed command group			*Universal command group*			*Listen address group*						*Talk address group*						*Secondary command group*					

Tableau I.4: **Codes des commandes et adresses des écouteurs et parleurs**

Transfert d'un octet de données: Le transfert d'un octet de données sur le bus DIO est assuré par le protocole de handshake qui est explicité par le diagramme temporel de la figure I.5 dans lequel on suppose un transfert de données issues d'un parleur vers plusieurs écouteurs.

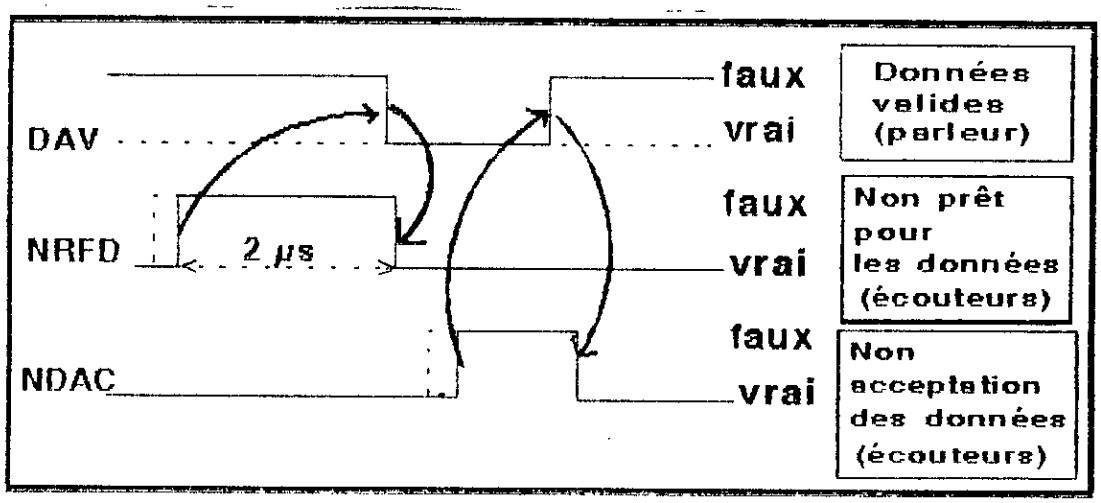


Fig.I.5: **Protocole de handshake entre un appareil parleur et des appareils écouteurs**

Les états succésifs des lignes DAV, NRFD et NDAC, au cours du transfert à l'intérieur de l'appareil parleur et des appareils écouteurs, sont représentés par l'organigramme de la figure ci-dessous:

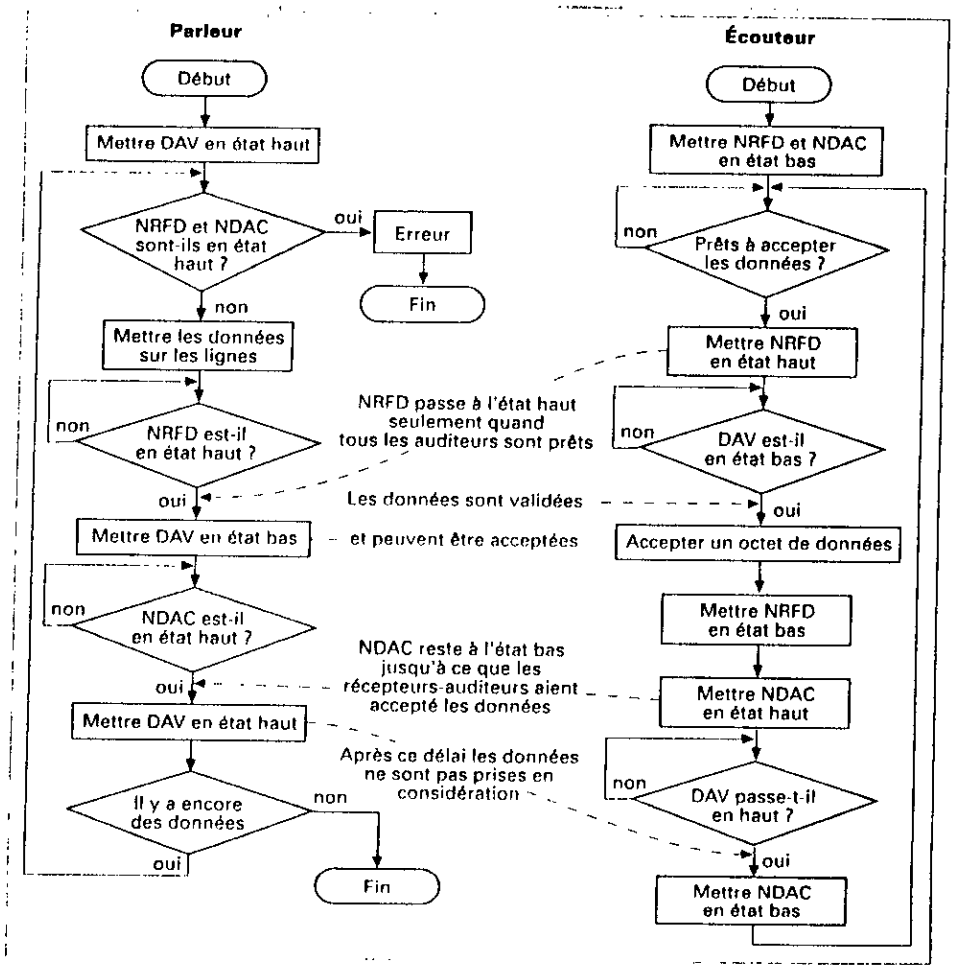


Fig. I.6: **Organigramme des états successifs de lignes DAV, NRFD et NDAC dans le parleur et les écouteurs**.

: La séquence de transfert d'un octet de données est résumée par le diagramme de la figure I.7 dans lequel on observe les instants suivants:

-à t_0 : la ligne DAV est mise à l'état haut par le parleur (cela signifie que les données ne sont pas validées). Les lignes NRFD et NDAC sont à l'état bas (vrai = 1), les écouteurs ne sont pas prêts à recevoir un nouveau mot de données (NRFD = 1) et les données ne sont pas acceptées (NDAC = 1);

-à t_1 : le parleur présente les premières données sur le bus DIO;

-à t_2 : les écouteurs adressés signalent qu'ils sont prêts à recevoir les données; à t_3 l'appareil le plus lent est enfin prêt et NRFD devient alors faux (NRFD = 0);

-à t_3 : le parleur peut alors déclarer valides les données qu'il présente sur le bus (DAV=1);

-à t_4 : l'écouteur le plus rapide ramène la ligne NRFD à l'état vrai (NRFD = 1) pour signaler au parleur qu'il y a au moins un des écouteurs qui n'est pas prêt à recevoir de nouvelles données;

-à t_5 : les écouteurs mettent leur sortie NDAC à l'état haut (NDAC), le plus lent des écouteurs a enfin accepté les données;

-à t_6 : le parleur remet DAV à l'état faux (DAV = 0) afin de changer l'octet qu'il présente sur le bus;

-à t_7 : les écouteurs remettent NDAC à l'état vrai (NDAC = 1) pour recommencer un nouveau cycle;

-à t_8 : les écouteurs sont de nouveau prêts à recevoir l'octet suivant et ainsi de suite... .

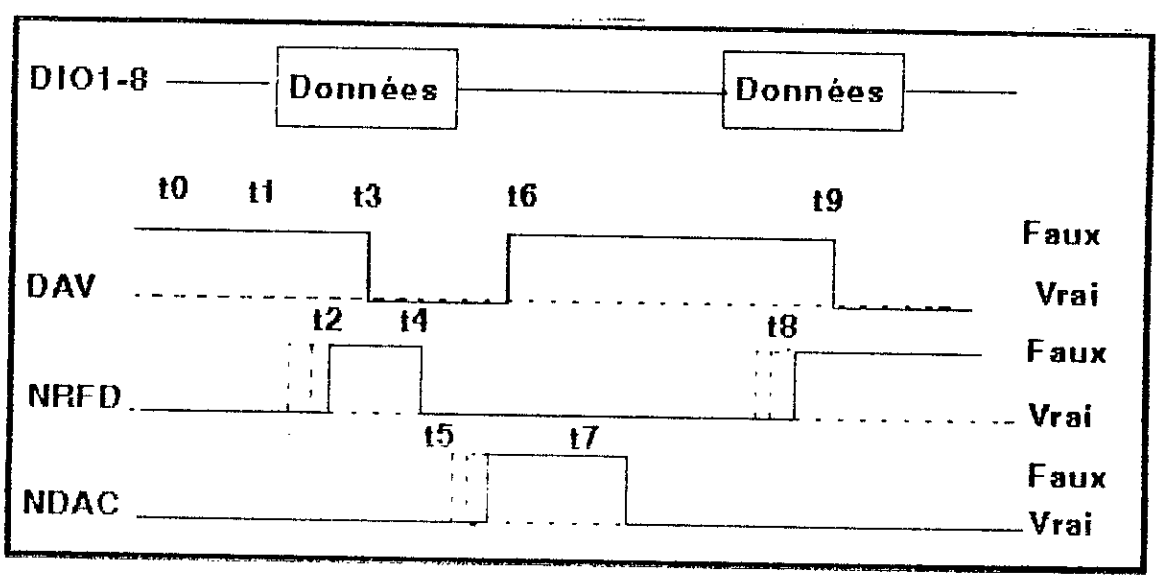


Fig.I.7: **Diagramme temporel de transfert d'un octet**

Transfert d'un octet d'adresse: Le diagramme temporel de la figure I.8 représente une séquence de transfert d'un message.

-à t_1 : ATN passe en mode commande(ATN = 1);

-à t_2 : NDAC se place à l'état vrai (NDAC = 1);

-à t_3 : l'adresse est placée sur les lignes DIO par le contrôleur, DAV passe à l'état vrai (DAV = 1);

-à t_4 : NRFD se place alors à l'état vrai (NRFD = 1), cela signifie que les appareils ne sont pas disposés à recevoir de nouvelles données;

-à t_5 : lorsque l'adresse est acceptée par un appareil, NDAC retourne à l'état faux (NDAC = 0);

-à t_6 : l'octet d'adresse sur les lignes DIO cesse d'être valide, DAV est remis à l'état faux (DAV = 0) par le contrôleur;

-à t_7 : les appareils placent NDAC à l'état vrai (NDAC = 1);

-à t_8 : NRFD revient à son état initial (ATN = 0) pour annoncer qu'il est prêt à recommencer;

-à t_9 : ATN repasse à l'état faux (ATN = 0) pour arrêter le mode de commande;

-à t_{10} : NDAC bascule à l'état faux (NDAC = 0) pour attendre le début de la séquence suivante.

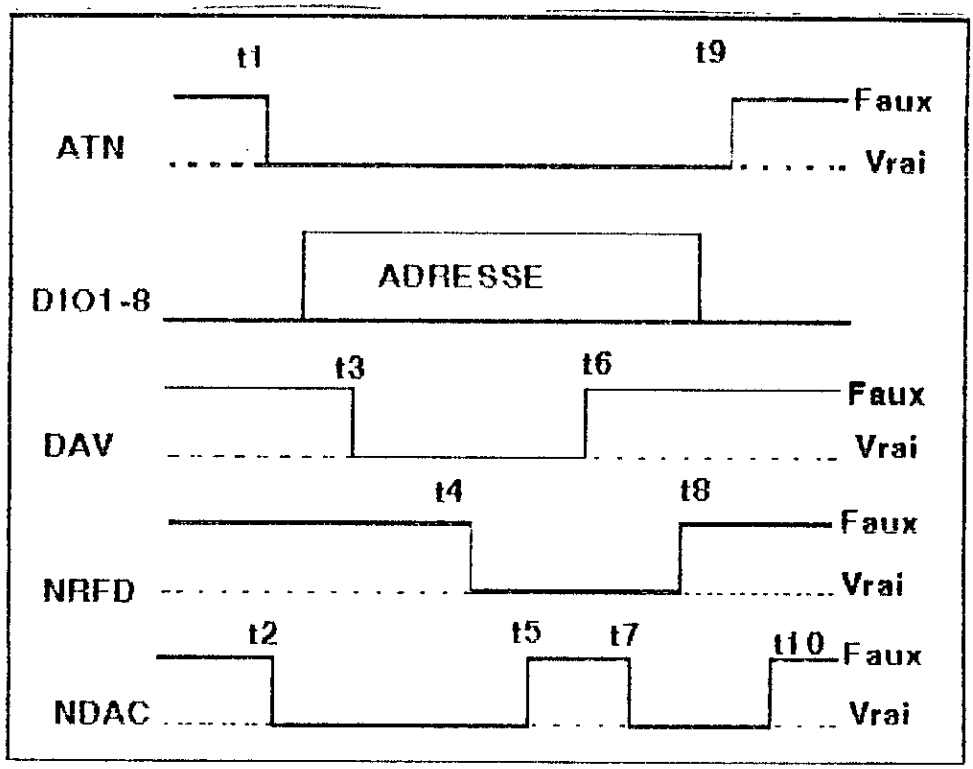


Fig.I.8: **Diagramme temporel du transfert d'un octet d'adresse**

8)-CARACTERISTIQUES MECANIQUES:

Le raccordement au bus utilise un connecteur de 24 broches (Fig.I.9). Le connecteur femelle est sur l'appareil IEEE.

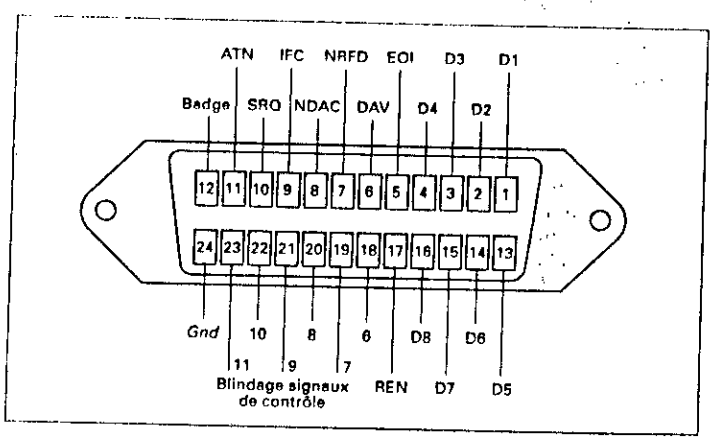


Fig.I.9: **Désignation des contacts du connecteur**

Le câble est passif et possède à chacune de ses extrémités une prise mâle, une prise femelle et deux vis dont la tête est identique aux écrous de la prise châssis ce qui rend aisé le raccordement en parallèle des instruments.

Il peut être constitué de la manière suivante:

- Le blindage (shield) est une tresse.
- Au coeur, les lignes de contrôle, chacune torsadée avec sa ligne de masse
- A la périphérie: les lignes de données.

La longueur maximale du câble ne doit pas dépasser 20 mètres au total et 2 mètres au maximum par instrument. Celle-ci est limitée de telle sorte que les temps de propagation des signaux ne puissent pas entrainer la désynchronisation des communications.

9)-CARACTERISTIQUES ELECTRIQUES:

* La logique employée est négative:

- Le niveau logique 1, actif, est rendu par une tension inférieure à 0,4 V.

- Le niveau logique 0, inactif, est traduit par un niveau de tension haut: 2 à 5 V.

* Les niveaux de tension et de courant fournis ou reçus par les émetteurs-récepteurs sont indiqués par le tableau suivant:

Entrée GPIB récepteur	Sortie GPIB émetteur	Vrai ou faux	Etat logique	Minimum	Maximum
Vbas		vrai	1	-0.6 V	+0.8 V
Vhaut		faux	0	+2.0 V	+5.5 V
	V bas	vrai	1	0.0 V	+0.4V
	V haut	faux	0	+2.4 V	+5.0 V
I bas		vrai	1	-1.6 mA	
I haut		faux	0		+50µA
	I bas	vrai	1		+48 mA
	I haut	faux	0		-5.2 mA

Tableau I.5: **Niveaux de tension et de courant.**

* Des circuits à collecteur ouvert sont utilisés pour les lignes SRQ, NRFD, NDAC, ATN, IFC, REN, EOI, DAV et DIO.

Pour les vitesses de transmission élevées (1 MO/s), les sorties ATN, IFC, REN, EOI, DAV et DIO doivent être constituées de circuits à 3 états. Pour les appareils dotés de la possibilité d'une réponse à la recherche parallèle, les lignes DIO sont obligatoirement en collecteur ouvert.

Les sorties à collecteur ouvert sont les plus répandues. Bien que plus lentes que les sorties trois états, elles ont l'avantage de supporter sans problème les court-circuits à la masse.

* Les temps de transition aussi sont spécifiés par la norme. On les retrouve dans les diagrammes d'état:

T: temporisation pendant laquelle l'état doit être stable,et c'est après ce laps de temps seulement qu'on peut transiter de l'état vers un autre.

t: temps maximal autorisé par un transition.

T_1	> 2 µs	Stabilisation des messages multilignes
t_2	< 200 ns	Réponse à ATN
T_3		Acceptation d'un message d'interface
t_4	< 100 µs	Réponse à IFC ou REN faux
t_5	<= 200 ns	Réponse à ATN ^ EOI
T_6	>= 2µs	Exécution de la reconnaissance parallèle
T_7	> 500 n s	Temporisation contrôleur pour permettre au parleur de recevoir ATN
T_8	> 100 µs	Durée de IFC et REN faux
T_9	>= 1.5 ns	Temporisation pour EOI

Tableau I.6: **Temps de transition**

10)-CARACTERISTIQUES FONCTIONNELLES:

Pour pouvoir réaliser sur le bus les séquences et processus prévus par la norme, chaque appareil ou calculateur doit posséder, outre ses propres fonctions, un ensemble électronique spécifique qui a pour but de le rendre compatible à la norme.

D'un point de vue fonctionnel, on peut décomposer cette interface en dix sous-ensembles; chaque partie synthétise une fonction particulière sur le bus.

La plupart des appareils connectés sur le bus ne comprenne qu'une partie de ces dix fonctions. Ces dernières sont assez souvent décrites par une phrase imprimée sur la face arrière de l'appareil près du connecteur et qui indiquent précisément ce qu'ils sont capables de réaliser. Toutes les fonctions existent sous plusieurs variantes, affectées d'un numéro.

1- La fonction SH: (Source Handshake)

Cette fonction a trait uniquement à l'envoi d'un mot sur le bus . L'émetteur peut être:

- Un contrôleur envoyant des commandes;
- Un parleur transférant des paramètres de mesure ou des résultats;
- Un appareil lors d'une reconnaissance série.

2- La fonction AH:(Acceptor Handshake)

Cette fonction est la duale de la fonction SH;elle est associée à tout appareil ou calculateur réceptionnant des mots de 8 bits. Le récepteur peut être:

- Un appareil recevant une adresse ou un message de gestion;
- Un appareil ayant été adressé écouteur;
-Un contrôleur recevant le mot d'état dans une reconnaissance série.

3- La fonction L: (Listener)

La fonction écouteur s'accompagne toujours de la fonction AH. Son rôle est d'ouvrir à l'appareil lui même l'accés au bus et de faire le lien avec les fonctions de Handshake pour synchroniser l'émission.

4- La fonction T: (Talker)

La fonction parleur s'apparente à l'écouteur. Elle s'accompagne obligatoirement de la fonction SH et elle n'a de sens que s'il existe dans la configuration un écouteur.

*** La reconnaissance série:**

Un appareil peut désirer attirer l'attention du contrôleur. Il active alors la ligne SRQ. En réponse, le contrôleur va chercher à savoir qui le demande et pourquoi; il pourra alors en connaissance de cause intervenir.

La séquence reconnaissance série est prévue pour cette recherche. Son objectif est que le contrôleur reçoive de chaque appareil un mot de 8 bits, le renseignant sur l'état courant des instruments: STB ou mot d'état. Le bit 7 du mot d'état reflète obligatoirement la validation ou non de la ligne SRQ par l'appareil. Les autres bits sont laissés à la disposition des constructeurs. Ils sont en général utilisés pour décrire la cause et la nature du service demandé.

Le contrôleur déclare d'abord que la reconnaissance va commencer: en mode de commande, il envoie SPE. Puis, il adresse à tour de rôle chacun des appareils susceptibles de demander service. Il adresse le premier en parleur puis il relâche ATN. L'appareil désigné, en mode de transfert, envoie son STB au contrôleur (adressé en écouteur) qui le réceptionne puis réactive ATN.

Ayant analysé le STB reçu, il décide soit de continuer sa recherche, soit de l'arrêter. Pour ce faire, il envoie SPD .

5- **La fonction SR:**(Service Request)

Cette fonction est liée à la reconnaisance série. Elle caratérise l'envoi de SRQ. A tout moment et d'une manière asynchrone, l'appareil peut valider la ligne SRQ. Par contre suivant la programmation, le contrôleur ignore la demande ou la prend en compte après un laps de temps. Sa réponse est la reconnaissance série. Dés que l'appareil transmet son STB, il relâche la ligne SRQ.

6- **La fonction PP:** (Parallel Poll)

En recherche parallèle, plusieurs instruments peuvent ensemble envoyer une information au contrôleur. Son déroulement est plus rapide que celui de la recherche série:

- Les appareils sont initialisés une seule fois; on affecte à chacun une ligne parmi huit sur laquelle il répondra.

- Par la suite, le contrôleur déclenche une reconnaissance parallèle en activant ensemble ATN et EOI.

- Aussitôt, tous les appareils, participant à la recherche, répondent chacun sur sa ligne DIOn.

- Sans protocole de handshake, le contrôleur lit l'octet DIO1-8, puis relâche EOI. La recherche est terminée.

La distribution des lignes entre les instruments est accomplie par le bus. Le contrôleur adresse à tour de rôle chaque instrument en écouteur, puis lui envoie PPC, PPEn.

- L'octet PPC a un codage fixé, il introduit PPEn.

- PPE réalise l'affectation: PPE = X 1 1 0 S $P_3 P_2 P_1$:

  * Le bit S détermine la logique employée: positive ou négative.

  * Les bits P3, P2 et P1 représentent un nombre binaire de 0 à 7 qui correspond à une ligne DIO.

- Pour redéfinir une nouvelle configuration de test, il est nécessaire d'invalider la précédente: PPD désactive sélectivement des appareils alors que PPU annule toute la configuration de reconnaissance.

7- **La fontion RL:** (Remote Local)

Cette fonction fait la connexion entre les fonctions propres à l'appareil et l'interface. Elle élabore le choix entre les deux modes de programmation: local ou à distance.

Tant que la ligne REN n'est pas validée par le contrôleur, l'appareil est toujours en mode local. Pour le faire passer en mode à distance, le contrôleur doit valider REN puis envoyer l'adresse écouteur de l'appareil.

Remarque:

Certains appareils possèdent un bouton poussoir rtl, qui permet à l'utilisateur de reprendre la main à la face avant alors que REN est actif.

La commande LLO envoyée par le contrôleur inhibe cette possibilité rtl. Elle est une protection contre une éventuelle mauvaise manipulation .

8- **La fonction DC:** (Device Clear)

La fonction DC sert à initialiser les fonctions propres à l'appareil c'est à dire le remettre dans l'état où il se trouve après une mise sous tension . Elle est équivalente au bouton de reset de l'appareil.

Cette réinitialisation peut être faite pour tous les appareils à la fois à l'aide de la commande DCL ou d'une manière sélective à l'aide de la commande SDC. Dans ce dernier cas seuls les instruments écouteurs sont remis à zéro.

9- **La fonction DT:**(Device Trigger)

La fonction DT est utilisée pour déclencher une mesure à un moment précis. En mode de commande, tous les appareils concernés sont adressés écouteurs, puis le contrôleur envoie GET. A sa réception tous les appareils adressés déclenchent une mesure.

L'intérêt majeur de cette fonction est de pouvoir ainsi déclencher simultanément une mesure sur plusieurs appareils.

10- **La fonction C:** (Controller)

La fonction contrôleur est la plus importante mais aussi la plus complexe de l'ensemble. Le contrôleur entre en jeu dans presque toutes les autres fonctions.

Semblable à la fonction parleur, un seul contrôleur est actif sur le bus. Il peut déléguer son pouvoir à un autre contrôleur en lui envoyant la commande TCT.Seul le contrôleur système valide les lignes REN et IFC.

11)- CIRCUITS D'IMPLEMENTATION DE LA NORME:

- L'interface entre le bus d'un microprocesseur et le bus IEEE demande un très petit nombre de circuits intégrés. Elle est facilement réalisable sur une seule carte. Aussi quasiment toutes les familles l'ont dans leurs catalogues.

- Elle contient souvent deux produits:

* La carte elle-même, avec le circuit Ecouteur/Parleur/Contrôleur.

* Un logiciel, en mémoire morte, qui accomplit les tâches fastidieuses de gestion du circuit d'interface.

- Pour chaque famille de microprocesseurs un circuit d'interface spécialisé pour le bus IEEE a été étudié:

* Le 68488 pour la famille 6800 de MOTOROLA, FAIRCHILD,...

* Le 8291-8292 chez INTEL pour les MPU 8080, 8085, 8086 ...

* Le 9914 de Texas Instruments pour ses familles 9900, 99000

* D'autres circuits sont moins liés à une famille: le 7210 de NEC, le 4738 de RTC.

- Ces différents circuits sont accompagnés de boitiers buffers: TEXAS SN 75160/161, MOTOROLA 3341-3346/7/8, INTEL 8293.

CHAPITRE II: DESCRIPTION DU CONNECTEUR D'EXTENSION:

CHAPITRE II: DESCRIPTION DU CONNECTEUR D'EXTENSION:

Nous nous référerons dans la suite de ce chapitre au PC disponible au labo n° 11 de notre département et qui dispose d'un processeur INTEL 486 DX 4 TM.

1)- LE BUS D'EXTENSION:

Le bus d'extension s'étend sur une rangée de 7 connecteurs (de X1 à X7). Chaque broche est séparement câblée en parallèle avec toutes les autres broches du même numéro.

La figure ci-dessous représente un bloc diagramme de ces connecteurs:

- X1, X2, X3, et X4 sont des connecteurs à 62 broches pour des dispositifs 8 bits.
- X5, X6 et X7 sont des connecteurs à 98 broches pour des dispositifs 16 bits.

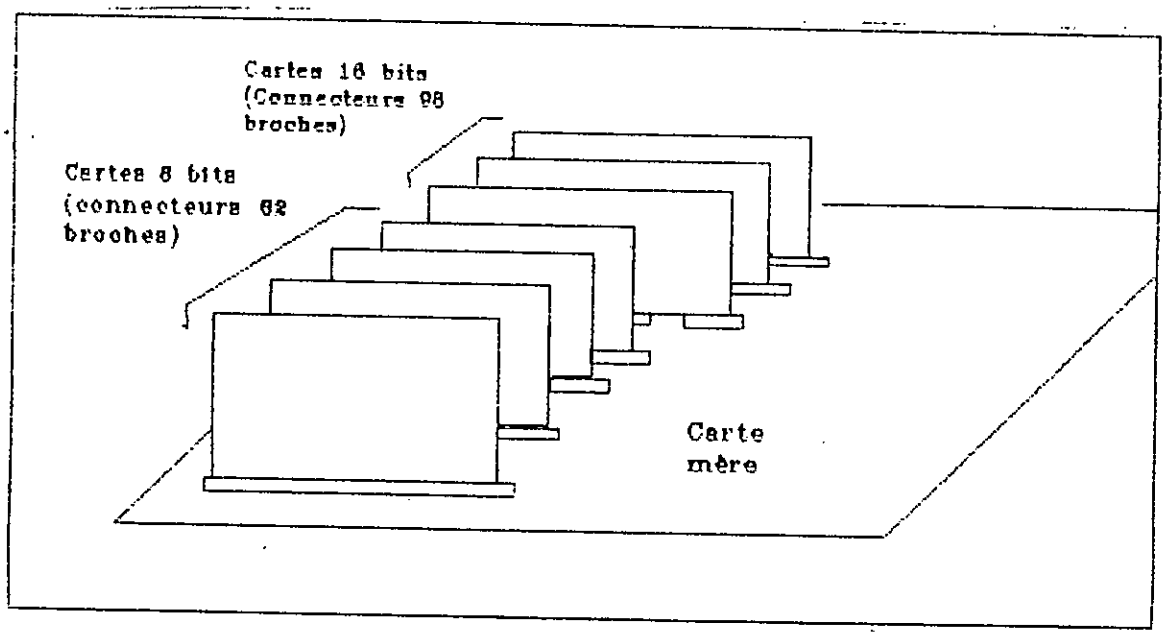


Fig.II.1: **Le bus d'extension**

2)- LES CONNECTEURS 8 BITS:

Tous les connecteurs à 8 bits (de X1 à X7) sont capables de supporter des dispositifs 8 bits étant donné que les 62 broches sont cablées en parallèle. Les numéros des broches de ces connecteurs figurent sur le schéma ci-dessus:

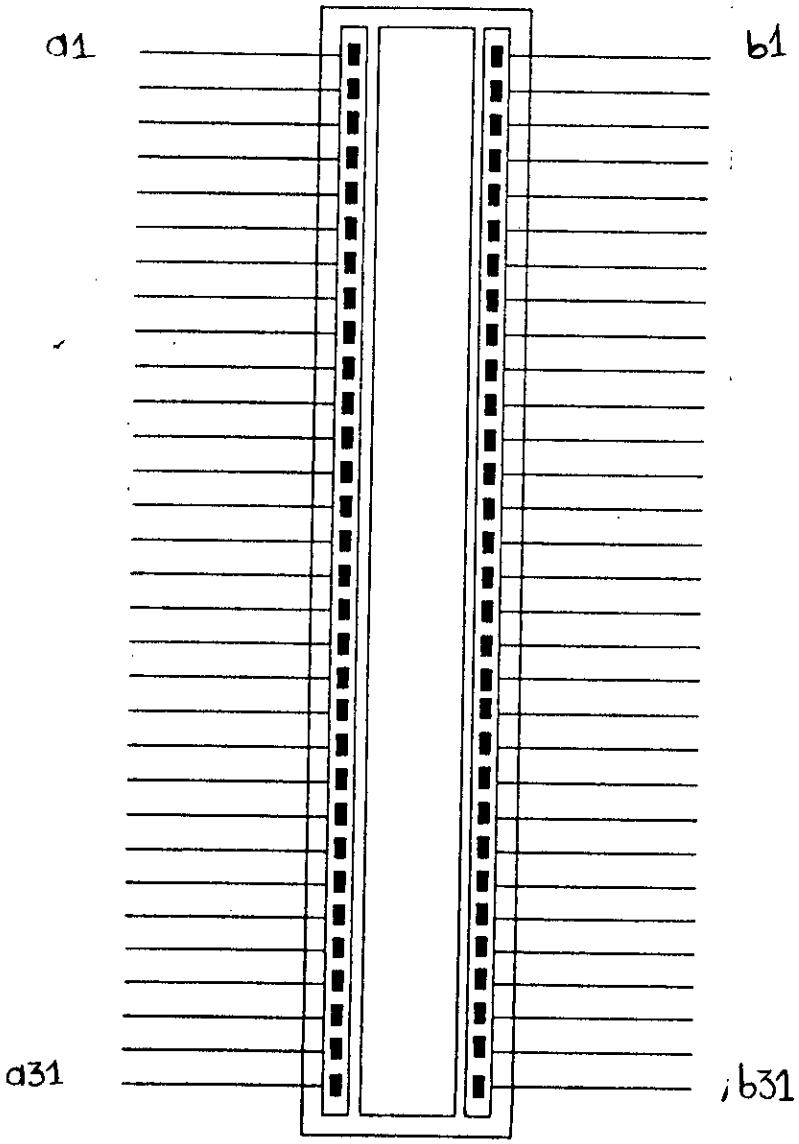


Fig.II.2: **Connecteur 8 bits: Numérotation des broches**

Chaque côté de ce connecteur possède 62 broches. Vu du côté des composants, le côté (a) est sur la droite alors que le côté (b) est sur la gauche.

Le tableau ci-dessous donne les noms des différents signaux correspondants aux 62 broches ainsi que leurs types:

DESCRIPTION DU CONNECTEUR D'EXTENSION

PIN	NOM	TYPE	PIN	NOM	TYPE
a1	CHCK\	Entrée	b1	GND	Masse
a2	D07	E/S	b2	RESET	Sortie
a3	D06	E/S	b3	+ 5 V	Alimentation
a4	D05	E/S	b4	IRQ9	Entrée
a5	D04	E/S	b5	- 5 V	Alimentation
a6	D03	E/S	b6	DRQ2	Entrée
a7	D02	E/S	b7	- 12 V	Alimentation
a8	D01	E/S	b8	0WS	Entrée
a9	D00	Entrée	b9	+ 12 V	Alimentation
a10	RDY	Sortie	b10	GND	Masse
a11	AEN	E/S	b11	SMEMW\	Sortie
a12	A19	E/S	b12	SMEMR\	Sortie
a13	A18	E/S	b13	IOW\	E/S
a14	A17	E/S	b14	IOR\	E/S
a15	A16	E/S	b15	DACK3\	Sortie
a16	A15	E/S	b16	DRQ3	Entrée
a17	A14	E/S	b17	DACK1\	Sortie
a18	A13	E/S	b18	DRQ1	Entrée
a19	A12	E/S	b19	REFRESH\	E/S
a20	A11	E/S	b20	CLK	Sortie
a21	A10	E/S	b21	IRQ7	Entrée
a22	A09	E/S	b22	IRQ6	Entrée
a23	A08	E/S	b23	IRQ5	Entrée
a24	A07	E/S	b24	IRQ4	Entrée
a25	A06	E/S	b25	IRQ3	Entrée
a26	A05	E/S	b26	DACK2\	Sortie
a27	A04	E/S	b27	T/C	Sortie

a28	A03	E/S	b28	BALE	Sortie
a29	A02	E/S	b29	+ 5 V	Alimentation
a30	A01	E/S	b30	OSC	Sortie
a31	A00	E/S	b31	GND	Masse

Tableau II.1: **Affectation des broches**

3)- LES CONNECTEURS 16 BITS:

Les connecteurs X5, X6 et X7 peuvent également être utilisés pour connecter des dispositifs 16 bits du fait que des connecteurs additionnels de 36 broches sont fournis par le constructeur.

La figure II.3 montre la numérotation des 36 broches de ces connecteurs.

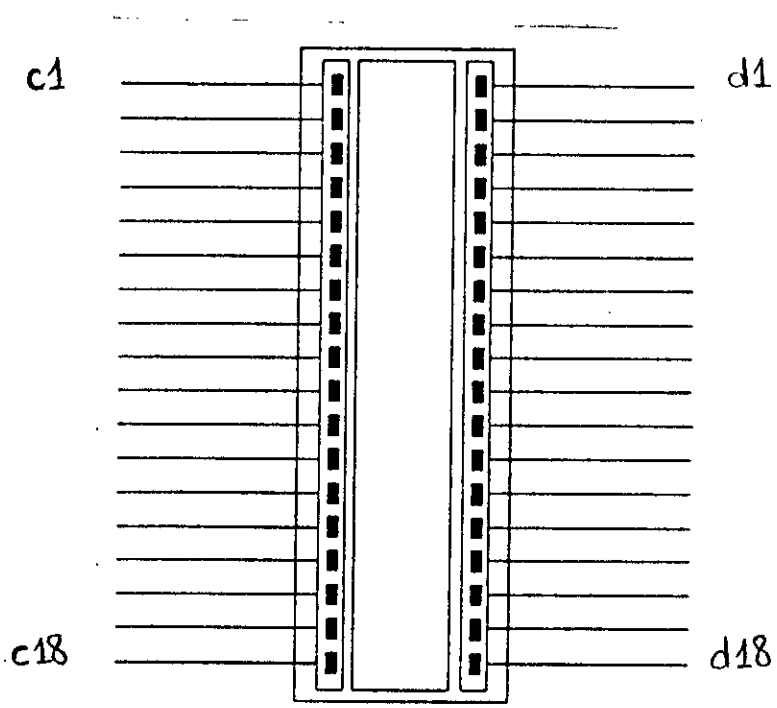


Fig.II.3: **Le connecteur additionnel du bus d'extension.**

DESCRIPTION DU CONNECTEUR D'EXTENSION

Chaque côté contient 18 broches. Vu du côté des composants, le côté (c) est sur la droite tandis que le côté (d) est sur la gauche.

Ces connecteurs sont labelés X12, X13 et X14 pour les distinguer des connecteurs 8 bits à 62 broches. En réalité, tous les deux ne forment qu'un seul connecteur contigu; ils sont décrits comme s'ils étaient deux connecteurs séparés uniquement pour distinguer entre les dispositifs à 8 bits et ceux à 16 bits.

Le tableau ci-dessous donne les différents signaux correspondants aux 36 broches de ce connecteur ainsi que leurs types:

PIN	NOM	Type	PIN	NOM	Type
c1	SBHE\	Sortie	d1	MEMCS16\	Entrée
c2	LA23	E/S	d2	I/O CS16\	Entrée
c3	LA22	E/S	d3	IRQ10	Entrée
c4	LA21	E/S	d4	IRQ11	Entrée
c5	LA20	E/S	d5	IRQ12	Entrée
c6	LA19	E/S	d6	IRQ15	Entrée
c7	LA18	E/S	d7	IRQ14	Entrée
c8	LA17	E/S	d8	DACK0\	Sortie
c9	MEMR\	E/S	d9	DRQ0	Entrée
c10	MEMW\	E/S	d10	DACK5\	Sortie
c11	D08	E/S	d11	DRQ5	Entrée
c12	D09	E/S	d12	ACK6\	Sortie
c13	D10	E/S	d13	DRQ6	Entrée
c14	D11	E/S	d14	DACK7\	Sortie
c15	D12	E/S	d15	DRQ7	Entrée
c16	D13	E/S	d16	+ 5 V	Alimentation
c17	D14	E/S	d17	MASTER\	Entrée
c18	D15	E/S	d18	GND	Masse

Tableau II.2: **Affectation des broches du connecteur additionnel**

4)- DESCRIPTION DES SIGNAUX DU BUS D'EXTENSION:

Les connecteurs du bus d'extension décrits précédemment alimentent les cartes d'extension et leurs délivrent les signaux d'interface. Tous ces signaux sont compatibles TTL avec une charge maximum de deux charges TTL-LS par carte.

Notre carte d'entrée/sortie est un dispositif à 8 bits, c'est pourquoi nous ne décrirons, dans ce qui suit, que les signaux nécessaires à sa réalisation, en omettant la description des autres signaux.

* **D00-D15:** Les lignes de données D00 à D15 sont utilisées pour échanger des données entre le CPU/DMA controler, les mémoires et les entrées/sorties. Les cartes 8 bits utilisent les lignes D00-D07, quand aux cartes 16 bits, elles utilisent les lignes D00-D15.
L'accés d'un processus 16 bits à une carte 8 bits se fait en deux accés de 8 bits; pour l'accés de l'octet haut, les lignes D08-D15 sont switchées à travers un buffer (swap) à D00-D07 du bus de données.

* **RDY:** Cette ligne, normalement à l'état haut,est mise à l'état bas par le dispositif connecté au bus d'extension pour allonger les cycles d'entrée/sortie. Ce signal asynchrone allonge les temps d'accés standardisés sur la carte mère.
Les cartes lentes génèrent ce signal à partir de l'adresse de la carte et le signal de lecture ou d'écriture. Quand il est à "0" (non prêt), le processeur (ou le contrôleur DMA) insère des cycles d'attente.
Ce signal doit être utilisé seulement lorsque le temps d'accés de 500 ns est insuffisant. Il ne doit pas être maintenu à l'état bas plus de 2,5 µs car ceci peut amener à la perte de données dûe à l'absence du rafraichissement. Il doit être généré par une sortie à collecteur ouvert ou à 3 états

* **CLK\:** C'est un signal d'horloge, carré, de fréquence 8 MHz et qui est utilisé pour synchroniser les opérations.

DESCRIPTION DU CONNECTEUR D'EXTENSION

* **AEN:** Cette ligne est utilisée pour permettre aux transferts DMA d'avoir lieu. Elle est nécessaire pour le décodage d'adresse de l'entrée/sortie. Lorsque cette ligne est active (à l'état haut), le contrôleur DMA a le contrôle du bus d'adresse, du bus de données et les lignes de commande de lecture/écriture pour accéder aux mémoires et aux entrées/sorties.

* **A00-A19:** Ces lignes sont utilisées pour adresser les mémoires et les dispositifs d'entrée/sortie. Comme signaux de sortie, ils sont générés par le MPU ou par le contrôleur DMA et comme signaux d'entrée, ils le sont par les cartes master du bus.
Ces lignes sont contrôlées soit par le processuer principal, soit par le contrôleur DMA ou par le processeur périphérique connecté au bus d'extension.

* **RESET :** Cette ligne est utilisée pour initialiser la logique de contrôle sur la carte d'extension. Elle est active au niveau haut.

* **IRQ 03-07, 09-12, 14, 15:** Ces lignes sont utilisées pour signaler au processeur de contrôle qu'un dispositif d'entrée/sortie demande de l'attention.
Elles obeïssent à une priorité avec IRQ09-15 ayant la priorité la plus haute (IRQ 09 est la plus prioritaire) et IRQ03-07 ayant la priorité la plus basse (IRQ07 est la moins prioritaire).
Une demande d'interruption est générée en basculant la ligne IRQ (transition: bas -> haut) et en la maintenant haute jusqu'à ce qu'elle soit reçue par le processeur durant une routine d'interruption. Pour notre part, on utilise la ligne IRQ05.

* **IOW\:** Ce signal, actif à l'état bas, indique au dispositif d'entrée/sortie adressé de lire les données se trouvant sur le bus de données. Il est piloté par le MPU\ DMA controller.

* **IOR\:** Ce signal, actif à l'état bas, indique au dispositif d'entrée/sortie adressé d'appliquer ses données sur le bus de données. Il est piloté par MPU\ DMA controller.

CHAPITRE III: PRESENTATION DU MC 68488:

CHAPITRE III: PRESENTATION DU MC 68488:

1)- DESCRIPTION ET BROCHAGE:

Le **MC 68488** ou **GPIA** (General Purpose Interface Adapter) est un circuit LSI à 40 broches qui fonctionne en logique positive. Il a été conçu par MOTOROLA pour servir d'interface entre les microprocesseurs de la famille 6800 et le bus IEEE-488.
Ce dernier, comme il a été déja cité, fournit tous les moyens nécéssaires pour le contrôle et le transfert des données dans des systèmes complexes à instruments multiples.

Le GPIA manipule automatiquement tout le protolcole du handshake nécéssaire au bon fonctionnement du bus IEEE-488. Il a été conçu pour travailler avec les drivers du bus IEEE-488 : **MC3447** ou **MC3448A** respectant ainsi toutes les caractéristiques et les spécifications électriques de la norme. Avec une logique additionnelle, il peut être utilisé avec d'autres microprocesseurs .

La figure suivante montre le brochage du MC68488 et le tableau III.1 donne les noms de ses différentes broches ainsi que leurs types.

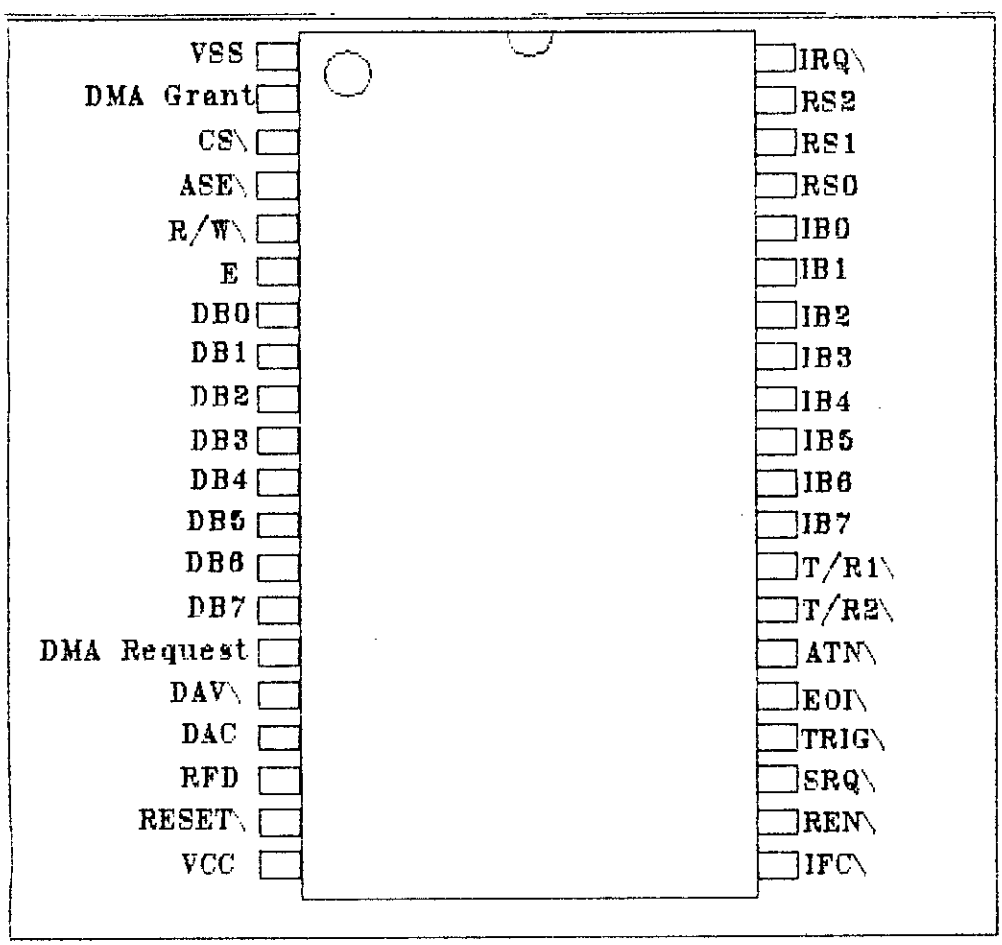


Fig.III.1 : **Brochage du MC 68488 (Vue de dessus)**

Noms des broches	Signification et type
DB0-DB7	Bidirectionnal Data lines
CS \	Chip Select input
R/W\	Read /Write input
RS0,RS1,RS2	Register Select inputs
IRQ\	Interrupt ReQuest output
RESET\	Chip RESET input
DMA Grant	DMA transfer in progress input
DMA Request	DMA transfer ready output
ASE\	Adress Switch Enable output
IB0\ ..IB7\	Bidirectionnal ASCII Bus
DAC	Bidirectionnal Data ACcepted line
RFD	Bidiretionnal Ready For Data line
DAV\	Bidirecionnal DAta Valid line
ATN\	ATteNtion input
IFC\	InterFace Clear input
SRQ\	Service ResQuet output
REN\	Remote ENable input
EOI\	Bidirecionnal End Or Identify line
TRIG\	Group execute TRIGger output
T/R1,T/R2	Transmit / Receive control outputs
E	Enabe clock input
VSS	Groud
VCC	+5 V Power supply

Tableau III.1: **Designation des différentes broches du MC 68488**

2)- FONCTIONS IMPLEMENTEES PAR LE MC 68488:

Plusieurs fonctions de la norme IEEE-488 sont gérées automatiquement par le MC-68488 et ne demandent aucune action additionnelle du MPU. D'autres fonctions demandent une réponse minimum dûe au grand nombre de registres internes contenant l'information sur l'état du MC 68488 et du bus IEEE-488.

Le MC68488 implémente directement les fonctions suivantes :

* Adressage primaire unique ou double;
* Adressage secondaire;
* Handshake complet émetteur-récepteur;
* Intérruptions programmmables;
* RFD Hold-off pour la prévention contre les dépassements de données;
* Possibilité de transfert avec le contrôleur DMA;
* Possibilité d'une recherche série et parallèle d'une demande de service;
* Modes "**Talk-only**" et "**Listen-only**";
* Sortie de synchronisation trigger;
* Programmation à distance ou locale;
* Initialisation ou remise à zéro (DC ou SDC);
* Emetteur et émeteur étendu;
* Récepteur et récepteur étendu.

Le MC 68488 assure seul (sans intervention du microprocesseur) le "Handshake" pendant la phase d'adressage. Pendant la transmission des données, il est capable de gérer aussi tout seul les lignes NDAC et NRFD en écouteur et DAV en parleur, mais la façon dont il opère peut être modifiée par logiciel.

Il n'interprète pas certaines commandes. Il doit donc les ignorer mais assurer correctement le handshake. Il est parfaitement capable d'adopter cette attitude. Mais, le MPU peut dans, une phase de mise en route par exemple, le configurer pour qu'il réagisse d'une autre façon à la réception d'une commande de classe AC ou UC qu'il ne comprend pas le MC 68488 interrompt le handshake et présente (dans le registre R6R) la commande "insolite" au MPU. Celui-ci en prend connaissance, décide de l'utilisation qu'il en fait, puis indique à son interface de poursuivre le handshake et d'acquérir le caractère suivant.

Par cette procédure le MC 68488 peut, directement avec l'intervention du MPU, réaliser toutes les fonctions qu'il n'implémente pas seul (sauf contrôleur).

De façon analogue, le MC 68488 peut abondonner le handshake automatique pour toutes ou une partie des données et attendre l'autorisation du microproceseur pour manoeuver les lignes NDAC et NRFD.

3)- LES SIGNAUX DU MC 68488:

Toutes les entées du MC 68488 sont à trois états et compatibles TTL.

Toutes les sorties sont aussi à trois états et compatibles TTL cependant IRQ\ et SRQ\ sont des sorties à collecteurs ouverts.

a)- Les signaux d'interface MPU-MC 68488: L'interface entre le MPU et le MC 68488 se fait par l'ensemble des signaux suivants:

1 - Le bus de données bidirectionnel (DB0-DB7): Les lignes de données bidirectionelles permettent le transfert des données entre le MPU et le MC 68488. Ces lignes sont amplifiées par des buffers trois états.

2 - La sélection du boitier (CS\): Ce signal d'entrée est utilisé pour sélectioner le MC 68488. Il est actif au niveau bas. La génération de CS\ est effectuée par une logique externe de décodage d'adresse.

3 - La ligne de lecture/écriture(R/W\): signal est généré par le MPU pour contrôler L'accés aux registres internes du MC 68488 et la direction du transfert des données sur le bus de données (DB0-DB7):

- Un niveau bas sur cette ligne permet la sélection d'un parmi les sept registres à écriture seulement quand elle est utilisée en conjonction avec les lignes de sélection des registres RS0, RS1 et RS2.
- Un niveau haut sur cette ligne permet la séléction d'un parmi les huit registres à lecture seulement quand elle est utilisée en conjonction avec les lignes de séléction des registres RS0,RS1 et RS2.

4 - La sélection des registres (RS0,RS1 et RS2): Ces trois entrées sont utilisées pour sélectionner les différents registres internes du MC 68488. Elles sont utilisées en conjonction avec la ligne R/W\ pour séléctionner un regitre particulier qui va être lu ou écrit.

Le tableau ci-dessus montre le codage servant à la séléction de ces registres:

RS2	RS1	RS0	R/W\	Titre du registre	Symbole
0	0	0	1	Interrupt status	R0R
0	0	0	0	Interrupt mask	R0W
0	0	1	1	Command status	R1R
0	0	1	0	Unused	R1W
0	1	0	1	Address status	R2R
0	1	0	0	Address mode	R2W
0	1	1	1	Auxillary command	R3R
0	1	1	0	Auxillary command	R3W
1	0	0	1	Address switch	R4R
1	0	0	0	Address	R4W
1	0	1	1	Serial poll	R5R
1	0	1	0	Serial poll	R5W
1	1	0	1	Command pass-through	R6R
1	1	0	0	Parallèle poll	R6W
1	1	1	1	Data-In	R7R
1	1	1	0	Data-Out	R7W

Tabeau III.2: **Codage des registres internes du MC 68488**

5 - La demande d'interruption (IRQ\): La sortie IRQ\ va au bus d'intérruptions commun du MPU. C'est une sortie à collecteur ouvert qui est câblée de façon à réaliser un "OU" avec les autres lignes d'intérruptions IRQ\. Cette ligne est positionnée à un niveau bas quand une interruption arrive et reste à l'état bas jusqu'à ce que le MPU lit le registre d'état d'interruption (R0R). La lecture de R0R remet IRQ\ à l'état haut.

6-L'initialisation (RESET\): L'entrée RESET\, active au niveau bas, permet d'initialiser le MC 68488 lors de la mise sous tension du système. A l'état bas, RESET\ cause :

* La rénitialisation de toutes les conditions d'état;

* L'initialisation du registre masque d'interruptions ;

* Place le MC 68488 dans l'état "UNTALK/ UNLISTEN";

* La réinitialisation des registres "Parallel poll", "Serial poll", "Data-In" et Data-Out".

* L'éffacement des registres "Address" et "Address Mode ";

* La rénitialisation de toutes les conditions dans le registre de commandes auxilliaires à l'exception du bit 7;

* Remet à l'état bas les signaux T/R1\ et T/R2\.

Remarques:

- Quand RESET\ revient à l'état haut (état inactif) , le MC 68488 reste dans l'état RESET\ jusqu'à ce que le MPU remet le bit 7 du registre de commandes auxilliaires à l'état bas.

- Avant le déclenchement du programme de bit reset, le seul registre auquel on peut accéder est le registre adresse.

- Les conditions affectées par RESET\ ne peuvent être changées pendant que cette broche est à l'état bas.

7- Les lignes de contrôle DMA (DMA Grant - DMA Request): La ligne**DMA Request** est utilisée pour signaler au contrôleur de DMA qu'une demande de transfert de données est en cours et qu'il faut attendre. Cette sortie est mise à l'état haut si le bit BI (ou BO) est mis à "1" dans le registre R0R et le bit correspondant dans R0W est mis à "1" aussi. Elle est remise à "0" quand le signal DMA Grant est à "1".

La ligne **DMA Grant** est utilisée pour signaler au MC 68488 que le bus du système est sous contrôle du contrôleur de DMA. Cette entrée ,mise à 1, sélectionne le registre 7 . Elle déconnecte les RS0, RS1 et RS2.

Durant ce temps, CS\ doit être à l'état haut. La fonction R/W\ est inversée en R\/W. Aussi , si le contrôleur de DMA demande une écriture dans la mémoire, cette même ligne éffectue une opération de lecture du MC 68488 (lecture du R7R) et vice versa.

Remarques:

- La ligne "DMA Grant" est mise à la masse lorsqu'on envisage de ne pas utiliser un transfert par DMA.

- Le signal R/W\est généré par le MPU ou par le contrôleur de DMA pour contrôler l'accés aux registres internes du MC 68488 et la direction du transfert des données sur le bus.

R/W\	DMA Grant	OPERATION
0	0	Ecriture
0	1	Lecture
1	0	Lecture
1	1	Ecriture

8 - La sortie Trigger (TRIG\): La broche TRIG\ est une sortie correspondant aux commandes GET et fget. Une initialisation hardware ou software met cette sortie à l'état bas. Elle peut être programmée pour être à l'état haut par l'une des deux méthodes suivantes:

* La mise à "1" du bit fget (bit 0 de R3W) par le MPU, ce qui cause la mise à "1" de la sortie Trigger. Cette dernière reste dans cet état jusqu'à ce que le bit fget soit programmé à l'état bas ou qu'un reset arrive.

* La sortie Trigger est mise à "1" dés la réception d'une commande GET du contrôleur .Elle est remise à l'état initial quand le MC 68488 quitte l'état TADS, c'est à dire quand les commandes GET, LADS ou ACDS arrivent.

9 - Adress Switch Enable (ASE\): La sortie ASE\ est utilisée pour activer les buffers trois états permettant ainsi le passage de l'adresse physique de l'instrument, fixée par des micro-interrupteurs, au bus de données du MPU.Elle est remise à "0" par la lecture du registre R4R du MC 68488.

10 - Enable Input (E): L'entrée E active les entrées adresses (CS\, RS0, RS1 et RS2) et le signal R/W\.Elle active aussi le transfert des données sur le bus de données. Elle est normalement dérivée à partir de l'horloge du MPU.

b)- Les signaux d'interface MC 68488-Bus IEEE-488:

Le MC 68488 fournit un système de 18 signaux d'interface avec le bus IEEE-488. Et comme il a été déja cité, la norme IEEE-488 définit ces signaux en logique négative tandis que les signaux du MPU et du MC68488 le sont en logique positive.

1 - Les Lignes de données (IB0-IB7): Ces lignes bidirectionnelles permettent l'échange de messages (messages d'interface et messages d'appareil) entre le MC 68488 et les transceivers MC3448A. Les données apparaissent sur ces lignes en bits parallèles, bytes séries. Elles sont amplifiées par les transceivers puis, appliquées au bus (DIO1-DIO8).

2 - Les Signaux de synchronisation du transfert: Ces lignes permettent le transfert de chaque byte de données sur le bus entre émetteur et récepteurs.

- La ligne RFD se positionne "vrai" indiquant que tous les récepteurs sont prêts à recevoir la donnée.
- L'émetteur indiquera la donnée valide en positionnant DAV\ au niveau bas.
- Dés la réception de la donnée valide par tous les récepteurs, DAC se positione "vrai" pour indiquer que la donnée a été acceptée par tous les récepteurs validés.

3 - Les lignes de contrôle du bus (ATN\, IFC\, SRQ\, REN\, EOI\): Ces lignes sont utilisées pour la gestion de l'interface.

*** ATN\:** est gérée par le contrôleur du bus IEEE- 488. Le MC 68488 répond au changement de cette ligne en moins de 200 ns, en activant les lignes de commande et les signaux de transmission /réception T/R1\ et T/R2\.

- Si les lignes ATN\ et EOI\ sont à l'état bas en même temps, le MC 68488 place le contenu de R6W sur le bus IEEE-488.
- Si ATN\ = 1, le MC 68488, le parleur et tous les écouteurs courants seront désactivés en libérant ainsi les lignes de données.

Durant le temps où ATN\ = 0 (état actif), seules des adresses ou des commandes transitent à travers les lignes IB0\..IB7\ et durant l'état inactif seules des données le font.

*** IFC\:** Cette ligne est issue du contrôleur du bus pour mettre le MC 68488 dans un état de repos prédéterminé. L'arrivée de IFC\ place le MC 68488 dans l'état de repos écouteur/parleur [LIDS : Listen IDle State / TIDS : Talker IDle State].

Si le MC68488 est dans l'état actif de l'écouteur avec un byte dans le registre Data-In (bit BI à 1),IFC\ place le MC 68488 dans l'état LIDS mais n'affecte pas le byte reçu ni l'état de l'indicateur BI.

Chaque fonction d'interface qui demande au périphérique d'être dans l'état actif de l'écouteur ou de parleur sera initialisée si IFC\ arrive.

Une commande générée par le MPU (exemple, lo, fget, hed) sera affectée seulement durant l'arrivée de IFC\ (IFC\ bas) et retourne à son état programmé initial quand IFC\ est haut; c'est à dire que IFC\ n'affecte pas les messages locaux.

*** SRQ\:** Ce signal est utilisé pour indiquer une demande de service ainsi qu'une demande d'interruption dans la séquence courante des évènements. Ceci indique au contrôleur du système qu'un appareil sur le bus a demandé un service.

* REN\: Ce signal est utilisé pour sélectionner l'une parmi les deux sources de commande (ou de programmation): locale ou à distance.

* EOI\: Le signal EOI\ est utilisé pour signaler la fin d'un transfert d'une séquence de bytes et, en conjonction avec ATN\ ,éxécute une reconnaissance parallèle.

ATN\	EOI\	Type de l'information sur le bus DIO
0	0	Octet de donnée
0	1	Dernier octet d'un bloc (fin de bloc)
1	0	Adresse
1	1	Demande d'identification aprés une interruption SRQ\

4 - Les signaux de contrôle de transmission /réception (T/R1\ ,T/R2\): Ces deux signaux sont utilisés pour le contrôle des transceivers véhiculant les signaux du bus IEEE-488. Chaque transceiver a une broche séparée de contrôle de transmission/ réception correspondant à une ligne particulière du bus. Ces broches peuvent supporter chacune une charge TTL.

- La ligne SRQ\ est câblée à l'état haut pour transmettre.
- Les lignes REN\, IFC\ et ATN\ sont câblées à l'état bas pour recevoir.
- La ligne EOI\ est contrôlée par T/R1\ à travers le transceiver lui permettant ainsi de transmettre ou de recevoir.

T/R1\ fonctionne exactement comme T/R2\ sauf pour la reconnaissance parallèle durant laquelle EOI\ sera dirigée en entrée par T/R1\ pendant que les lignes DAV\ et IB0-IB7 seront des sorties.

4)- LES REGISTRES INTERNES DU MC 68488:

Le MC68488 possède 15 registres, tous accéssibles par le MPU via son bus de données. Ces 15 registres permettent de stocker toutes sortes d'informations : données ,états ou commandes. Sept d'entre eux sont utilisés uniquement en écriture et permettent la programmation du MC68488 par le MPU et les huit autres travaillent seulement en lecture et traduisent au MPU tout ce qui a trait au bus IEEE-488.

L'accés aux différents registres s'éffectue par les trois lignes d'adresse de poids faible (A0, A1 et A2) du bus d'adresse du MPU et la ligne R/W\.

Un des registres est placé à l'extérieur du MC 68488, mais le registre R4R est fourni pour la lecture de l'adresse positionnée par des micro-interrupteurs. Ainsi, on a la possibitité de sélectionner l'adresse soit par des micro interrupteurs ou par logiciel en écrivant dans le registre R4W.

Le tableau ci-dessous détaille le contenu de ces registres:

Nom du registre	Abrv	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Interrupt Mask	R0R	IRQ	BO	GET		APT	CMD	END	BI
Interrupt Status	R0W	INT	BO	GET		APT	CMD	END	BI
Command Stat	R1R	UACG	REM	LOK		RLC	SPAS	DCAS	UUCG
Unused	R1W								
Address Status	R2R	ma	to	lo	ATN	TACS	LADS	LPAS	TPAS
Address Mode	R2W	dsel	to	lo		hlde	hlda		apte
Auxil. command	R3R	chip	DAC	DAV	RFD	msa	rtl	ulpa	fget
Auxil. command	R3W	Reset	rfdr	feoi	darc	msa	rtl	dacd	fget
Address Switch	R4R	UD3	UD2	UD1	AD5	AD4	AD3	AD2	AD1
Address	R4W	lsbe	dal	dat	AD5	AD4	AD3	AD2	AD1
Serial Poll	R5R	S8	SRQS	S6	S5	S4	S3	S2	S1
Serial Poll	R5W	S8	rsv	S6	S5	S4	S3	S2	S1
Command Pass-Through	R6R	B7	B6	B5	B4	B3	B2	B1	B0
Parallel Poll	R6W	PPR8	PPR7	PPR6	PPR5	PPR4	PPR3	PPR2	PPR1
Data-In	R7R	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
Data-Out	R7W	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

Tableau III.3: **Contenu des registres internes du MC 68488**

1)- **Data-In Register (R7R)**: Registre d'entrée de données

DI_7	DI_6	DI_5	DI_4	DI_3	DI_2	DI_1	DI_0

C'est un registre de stockage à 8 bits, utilisé pour recevoir un octet à partir du bus IEEE-488.Lorsque le MC 68488 est écouteur actif, il place l'octet qu'il reçoit dans R7R où le MPU peut le lire. Le seul fait de lire R7R libère NDAC puis NRFD dés que possible (si hlde ou hlda sont à 1, le MC 68488 ne libère NRFD que sur ordre du MPU par le bit 6 de R3W). La ligne DAC reste à l'état bas jusqu'à ce que le MPU déplace l'octet du R7R. le MC 68488 finit le handshake en mettant DAC à l'état haut.

Dans le mode RFD hold-off, un nouveau handshake ne commence que si une commande du MPU est envoyée au MC 68488. Ceci retardera le MC 68488 jusqu'à ce qu'une information disponible soit traitée.

Remarque:

DI_0 -DI_7 correspondent à: DIO1- DIO8 sur le bus IEEE-488,

IB_0 -IB_7 dans le MC 68488.

2)- **Data-Out Register (R7W):** Registre de sortie de données

DO_7	DO_6	DO_5	DO_4	DO_3	DO_2	DO_1	DO_0

C'est un registre de stockage à 8 bits, utilisé pour transférer les données du MC 68488 au bus IEEE-488. Le MPU place dans ce registre les données qu'il veut transmettre sur le bus. Lire dans R7R n'affecte en rien les informations dans R7W et écrire dans R7W n'affecte en rien aussi les informations dans R7R.

Remarque:

DO_0-DO_7 correspondent à :DIO1-DIO8 sur le bus IEEE-488,

:IB_0-IB_7 dans le MC 68488.

3)- Interrupt Mask Register (R0W): Registre de masque d'interruption

IRQ	BO	GET	X	APT	CMD	END	BI

C'est un registre de stockage à 7 bits, utilisé pour sélectionner des évènements particuliers qui causeront une interruption qui sera envoyée au MPU. Les 7 bits de ce registre peuvent être positionnés indépendement l'un de l'autre.

*IRQ: C'est le bit de masque pour la sortie IRQ. Il permet à toute interruption de passer au MPU.

*BO: Fait une interruption sur un byte sortant.

*GET: Positionné pour permettre à une interruption de se produire à l'arrivée d'une commande GET.

* X: Bit non utilisé.

*APT: Positionné pour permettre à une interruption de se produire indiquant:

- La disponibilité d'une adresse secondaire qui doit être examinée par le MPU si apte = 1 (actif).

- La réception d'une adresse primaire (Talker ou Listener).

- la réception d'un groupe de commandes secondaires.

Une réponse typique pour une adresse secondaire valide serait de positionner les bits msa et dacr, libérant DAC.

*CMD: Interruption sur SPAS+ RLC+ dsel\ (UACG+ UUCG+ DCAS). Le registre R1R peut être utilisé pour déterminer quelle commande a causé cette interruption.

*END: Interruption sur EOI\ actif et ATN\ inactif.

*BI: Interruption sur un byte entrant.

4- Interrupt Status Register (R0R): Registre d'état d'interruption.

INT	BO	GET	X	APT	CMD	END	BI

C'est un registre de stockage à 7 bits qui correspondent aux bits de R0W avec un bit supplémentaire INT (bit 7). A l'exception de ce dernier, les autres prennent les mêmes états que ceux des bits correspondants de R0W quand les mêmes intérruptions arrivent.

*INT: C'est le OR logique des AND des 6 autres bits de ce registre avec les bits correspondants de R0W.

* BO: Mis à 1 lorsque le registre R7W est vide (l'octet est sorti). Le MPU peut alors y écrire l'octet qu'il destine au bus IEEE-488.

* GET: mis à 1 l'arrivée d'une commande GET.

* CMD: Indique que l'interface a reçu une commande qu'elle n'a pas comprise ou que SPAS + RLC + dsel\ (DCAS + UUCG + UACG) a eu lieu.

* END: Positionné à l'arrivée de EOI\ actif et ATN\ inactif.

* BI: Mis à 1 lorsque l'interface (Listener actif) a reçu un octet.

Remarque: IRQ est effacée à la lecture de ce registre.

5- Serial Poll Register (R5R/W):

C'est un registre de stockage à 8 bits que le MPU peut lire ou écrire. Il est utilisé pour établir le byte d'état que le MC 68488 envoie en sortie pendant un sondage série.
Le byte d'état peut être placé dans les bits 0 à 5 et le bit 7. Le bit 6 rsv (request for service) est utilisé pour commander la logique qui contrôle la ligne SRQ sur le bus indiquant au contrôleur qu'un service est demandé. Cette même logique génère le signal SRQS (Service ReQuest State) qui est substitué dans le bit 6 à rsv quand le byte d'état est lu par le MPU.

Pour initialiser un rsv, le MPU met le bit 6 à 1 (générant ainsi un signal rsv); ceci oblige le périphérique demandant un service à mettre bas la ligne SRQ.

S8	SRQS	S6	S5	S4	S3	S2	S1

R5R

S8	rsv	S6	S5	S4	S3	S2	S1

R5W

*S1-S8: bits d'état.

*SRQS: le bus est dans l'état SRQS

*rsv: génére un service request.

6)- Parallel Poll Register (R6W):

PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1

Ce registre est chargé par le MPU. Ses bits sont envoyés au bus au bus IB0-IB7 durant l'état PPAS. (**PPAS** Parallel Poll Active State).

A la mise sous tension, ce registre prend l'état PP0 (Parallel Poll No Capability).

Le bit RESET (bit 7 de R3W) éffface ce registre et le remet à l'état PP0.

- La fonction d'interface PP est éxécutée par le MC 68488 en utilisant PP2 subset (Omit Controller Configuration Capability).Le contrôleur ne peut pas configurer directement la sortie PP du MC 68488. Cette configuration doit être faite par le MPU. Le contrôleur peut par contre configurer PP mais indirectement, en utilisant une commande adressée définie dans le software.

7)- Address Mode Register (R2W):

dsel	to	lo	X	hlde	hlda	X	apte

C'est un registre de stockage à 6 bits de contrôle.

* **dsel :** permet la configuration pour l'accomplissement du handshake à l'arrivée d'une des commandes GET, UACG, UUCG, SDC, ou DCL.

* **to:** sélectionne et adresse l'appareil (le MC 68488) pour parler uniquement .

* **lo:** sélectionne et adresse l'appareil pour écouter uniquement.

* **X:** bits non utilisés.

* **hlde:** Hold-off RFD on End stoppe le protocole à la fin. Il est actif à la transmission du dernier byte. Il réalise un hold-off RFD si EOI\ est actif et ATN\ inactif; ceci permet au dernier byte d' un bloc d'être lu continuellement. L'écriture de rfdr "vrai" libérera le handshake.

* **hlda:** met RFD en All Data tant que rfdr n'est pas actif.

* **apte:** autorise le MC 68488 à transmettre au MPU, pour les interpréter, les commandes qu'il ne comprend pas. Il est utilisé aussi pour permettre le mode d'adressage étendu.

- Si apte = 0, l'appareil quitte l'état TPAS à l'état TADS.

- Si maintenant apte = 1 et l'adresse secondaire est valide, il positionne msa = 1.

8)- Address Status Register (R2R):

ma	to	lo	ATN	TACS	LACS	LPAS	TPAS

Ce n'est pas un registre de stockage. mais simplement un port à 8 bits utilisé pour coupler signaux internes au bus du MPU. Il contient des indicateurs d'état relatifs au MC 68488, les stockés intérieurement dans la logique de l'appareil et qui indiquent son état d'adressage.

* **ma**: mis à 1 si le MC 68488 est dans un des états:

TACS: Taker ACtive State

TADS: Talker ADdressed State

LACS: Listener ACtive State

LADS: Listener ADdressed State

SPAS: Serial Poll Active State

ATN : Bit 4 contient la condition sur la ligne d'attention.

* **to:** le mode "Talk Only" est activé.

* **lo**: le mode "Lislen Ony" est activé.

* **ATN:** la commande ATN est activée.

* **TACS**: le MC 68488 est dans l'état Talker ACtive State.

*__LACS__: le MC 68488 est dans l'état Listen ACtive Astate.

*__LPAS:__ le MC 68488 est dans l'état Listener Primary Addressed State.

*__TPAS__: le MC 68488 est dans l'état Talker Primary Adrressed State.

9)- Address Switch Register (R4R):

UD_3	UD_2	UD_1	AD_5	AD_4	AD_3	AD_2	AD_1

Ce registre est externe au MC 68488. Une ligne d'activation (ASE\) est utilisée pour activer les drivers 3 états connectés entre des micro-intérrupteurs et le MPU.

Quand le MPU adresse ce registre, la ligne d'activation ASE\ permet le transfert de l'information sur l'adresse déterminée par les micro-intérrupteurs et l'envoit au MPU.

Les cinq bits de poids faible sont utilisés pour spécifier l'adresse du MC 68488 sur le bus IEEE-488. Les 3 bits restants peuvent être utilisés au gré de l'utilisateur. L'utilisation la plus probable d'un ou de deux bits de ces 3 bits est la commande des fonctions **lo ou to**.

* $\underline{AD_1 - AD_5}$: adresse du MC 68488.

* $\underline{UD_1 - UD_2}$: bits à définir par l'utilisateur.

10)- Address Register (R4W):

Isbe	dal	dat	AD_5	AD_4	AD_3	AD_2	AD_1

C'est un registre de stockage à 8 bits. Son rôle est de porter l'adresse primaire du MC 68488. Cette dernière est placée dans les 5 bits de faible poids de ce registre.
Si des micro-intérrupteurs externes sont utilisés pour l'adressage du MC 68488 , le MPU lit cette adresse à partir de R4R et la place dans R4W.

* $\underline{AD_1\text{-}AD_5}$: bits utilisés pour adresser le MC 68488.

* **Isbe**: est positionné à 1 pour activer le mode dual de l'adressage primaire. Durant ce mode, le MC 68488 répondra à 2 adresses consécutives: l'une avec $AD_1 = 0$ et l'autre avec $AD_1 = 1$.

* **dal**: positionné pour inhiber la fonction Listener.

* **dat**: positionné pour inhiber la fonction Talker.

Ce registre est remis à 0 par la broche RESET\ uniquement (et non pas par le bit reset de R3R/W).

11)- Auxiliary Command Register (R3R\W):

reset	rfdr	feoi	dacr	msa	rtl	dacd	fget

R3W

* **reset:** - Initialise le MC 68488 aux états suivants: SIDS, AIDS, TIDS, LIDS, LOCS, PIS, PUCS, PP0.

- positionné par l'entrée RESET\ ou par écriture du MPU dans ce registre.

* **rfdr:** permet l'accomplissement du handshake intérrompu par les commandes hlda et hlde.

* **feoi:** indique au MC 68488 d'envoyer EOI\ bas avec le byte transmis suivant.

La ligne EOI\ retourne à l'état haut après que le byte suivant est transmis.

* **dacr:** positionné pour activer DAC et indique que le MPU a examiné une adresse secondaire ou une commande indéfinie (classe UC).

* **msa:** positionné quand le MC 68488 est dans l'état TPAS ou LPAS. Le MC 68488 sera donc adressé pour parler ou pour écouter. L'adresse primaire doit être reçue antérieurement.

* **rtl:** permet au MC 68488 de répondre au contrôle local.

* **dacd:** positionné par le MPU et empêche l'accomplissement d'un handshake concernant les adresses et les commandes. dacr est utilisé pour activer le handshake.

* **fget:** a le même effet que le commande GET du contrôleur.

reset	DAC	DAV	RFD	msa	rtl	ulpa	fget

R3R

* **reset**: initialise le MC 68488 aux états suivants:

- Toutes les intérruptions sont remises à '0'.

- Les états suivants du bus sont affectés: SIDS, AIDS, TIPS, LIDS, LOCS, PPIS, PUCS, PP0.

- Le bit est positionné par l'entrée RESET\.

* **DAC, DAV et RFD:** prennent les mêmes états que les signaux correspondants sur le MC 68488. Ils sont seulement lus par le MPU et ne sont pas synchronisés avec l'horloge du MPU.

* **msa:** si le MC 68488 est dans les états LPAS ou TDAS, positionner**msa** le forcera à se mettre dans les états LADS ou TADS.

* **rtl:** retourner au contrôle local s'il est désactivé.

* **ulpa:** indique l'état du LSB de l'adresse reçue sur les lignes DIO1 - DIO8 au moment où la dernière adresse primaire est reçue. Il peut être lu (mais non écrit) par le MPU.

* **fget:** la commande fget du contrôleur est eu lieu.

12)- Command Stalus Register (R1R):

Ce registre indique les commandes et les états comme ils se passent.Il contient 7 bits indicateurs d'états couplés sur le bus.

UACG	REM	LOK	X	RLC	APAS	DCAS	UUCG

* UACG: indique qu'une commande d'adresse indéfinie a été reçue et suivant la programmation, le MPU va décider de son éxécution ou non.

* REM: indique l'état remote/local du Talker/Listener. Il est positionné si la commande à distance est activée.

* LOK: état local verouillé du Talker/Listener.

* RLC: se met à '1' quand un changement sur l'état Remote/Local arrive et sera remis à ' 0' à la lecture de R1R.

* SPAS: indique qu'une commande SPE a été reçue activant la fonction "Serial Poll".

* DCAS: indique que les commandes DC ou SDC ont été reçues activant la fonction associée à l'initialisation. (DC).

* UUCG: indique qu'une commande universelle a été reçue.

* X: bit non utilisé.

13)- Command Pass-Through Register (R6R):

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0

C'est un registre à 8 bits utilisé comme port. Lorsqu'il est adressé par le MPU, il met en liaison le Data Bus Instrument (IB0\-IB7\) au Data Bus du MPU (D0 - D7).
Ce port est utilisé pour faire passer au MPU les commandes et les adresses secondaires qui ne sont pas interprétées directement par le MC 68488, pour inspection.

CHAPITRE IV: CONCEPTION ET REALISATION DE LA CARTE FILLE IEEE-488:

CHAPITRE IV: CONCEPTION ET REALISATION DE LA CARTE FILLE IEEE-488:

Le PC constitue, aujourd'hui, le standard le plus répandu dans le monde de l'informatique. L'un de ses seuls points faibles est son côté communicatif assez rudimentaire. Un PC standard possède au minimum deux sorties: Une sortie centronics pour l'imprimante et une prise RS-232 accaparée dans bien de cas par la souris.

Si l'on prévoit de faire de la communication avec le monde extérieur, il est quasiment indispensable de doter son ordinateur d'une carte additionnelle spécialisée dans la gestion des entrées/sorties.

Notre étude consiste, justement, à concevoir et à réaliser une carte fille à base du MC 68488 de MOTOROLA et qui a pour tâche de rendre compatible notre ordinateur avec le bus normalisé IEEE-488.

La carte en question doit pouvoir réaliser sur le bus les séquences et les processus prévus par la norme. Elle devra assurer plusieurs fonctions à savoir:

* la garantie de la sécurité de l'ordinateur;
* l'amplification des signaux issus du connecteur d'extension;
* la possession d'une logique de commande pour l'écriture/lecture du MC 68488;
* la possession d'une logique de décodage de l'adresse;
* la possession aussi d'une logique de sélection de l'adresse physique;
* l'adaptation du microprocesseur au MC 68488 et au bus IEEE- 488;
* la synchronisation des opérations de transfert de l'information entre le bus et le MC 68488 d'une part et entre ce dernier et le microprocesseur d'une autre part;
* l'utilisation de transceivers de lignes pour rendre le MC 68488 opérationnel et assurer ainsi des transferts bidirectionnels des informations entre lui et le bus.

1)- SCHEMA SYNOPTIQUE:

Les principes cités ci-dessous nous ont permis de dégager le schéma synoptique de la figure V.1. Ce schéma montre la disposition adoptée pour les sous-ensembles constitutifs de la carte et le tableau V.1 donne une discription sommaire du rôle de chacun d'eux:

N°	Sous-Ensemble	ROLE
1	Connecteur d'extension	-Alimente la carte et lui délivre les signaux d'interface. -Sert aussi de support pour enficher la carte.
2	Amplificateurs de bus	-Améliorent les sortances des différents signaux du bus. -Assurent l'isolation de la carte. -Garantissent la sécurité du PC .
3	Circuit de décodage d'adresse	-Sélectionne la carte par la génération du signal CS\ de sélection du MC 68488.
4	Diviseur de fréquence	-Assure l'adaptation entre la vitesse de l'horloge du MPU et celle du GPIA.
5	Circuit de sélection de l'adresse physique	-Permet la sélection de l'adresse physique de l'interface sur le bus IEEE-488 et sa lecture par le MPU.
6	Circuit à retard.	-Assure la synchronisation des différentes opérations de transfert de données entre le MPU et le MC68488.
7	Le MC 68488	- C'est le circuit de base de cette carte - C'est à lui qui revient l'implémentation, la réalisation et la gestion des différents fonctions, séquences et processus prévus par la norme IEEE-488.
8	Transceivers de lignes	- Adaptent le MC68488 au bus IEEE-488 et commandent le sens des transferts de données sur le bus.
9	Connecteur IEEE-488	- Assure la liaison entre le MPU et les différents dispositifs faisant partie d'un système de mesure automatique.

Tableau VI.1: **Rôle des différents blocs du schéma synoptique**

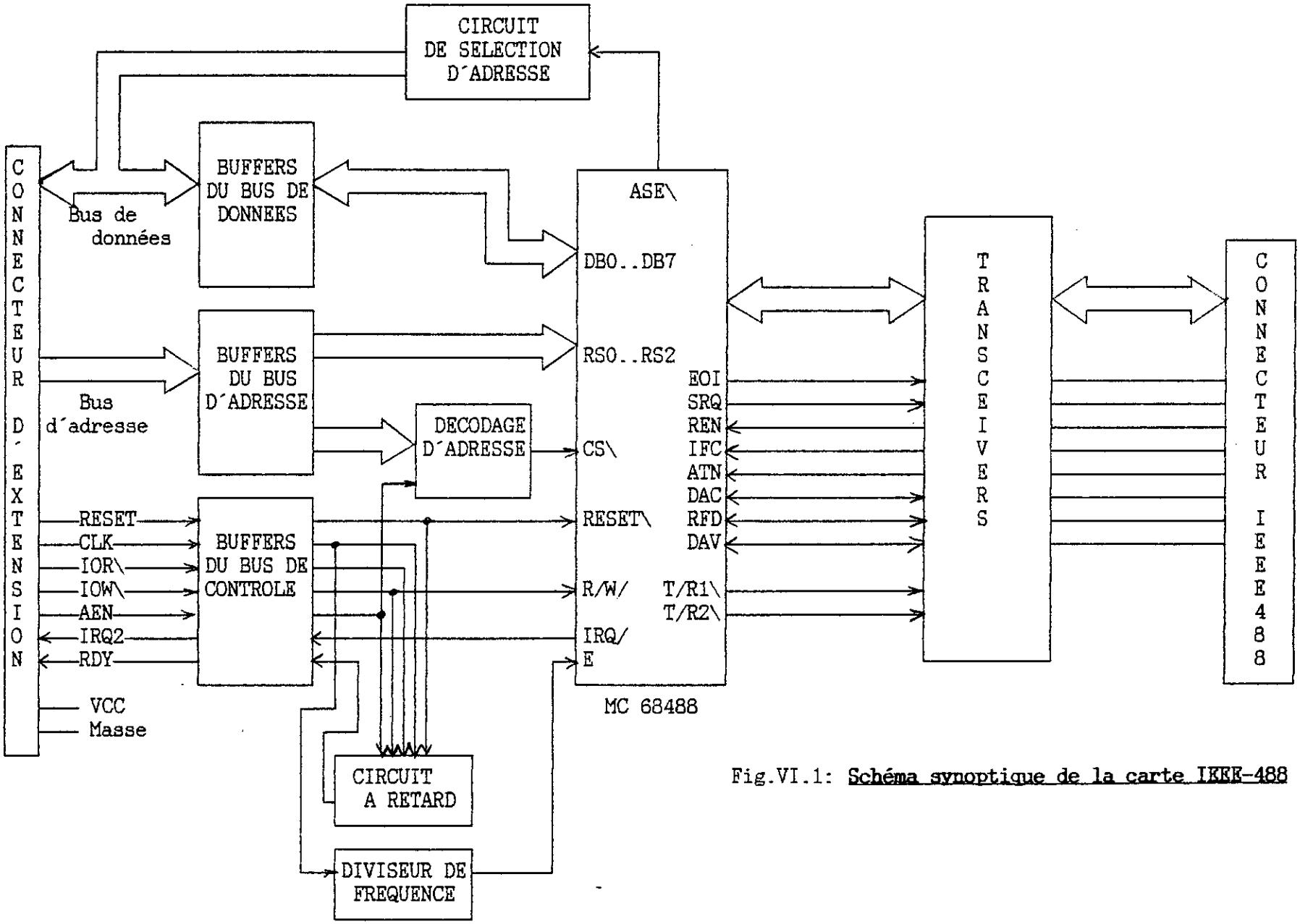


Fig.VI.1: Schéma synoptique de la carte IEEE-488

2)- ETUDE DETAILLEE (BLOC PAR BLOC):

2.1)- LE CONNECTEUR D'EXTENSION:

Ce connecteur a été déjà étudié en détail dans un précédent chapitre. Rappelons ici qu'il met en oeuvre les signaux suivants assurant la liaison du MPU avec la carte:

* La paire Vcc- GND pour l'alimentation de la carte.

* Les huit lignes du bus de données D00-07 du MPU qui seront reliées aux huit lignes de données du MC 68488.

* Les douze lignes du bus d'adresse du MPU A00-A11 dont trois d'entre elles (A0-A02) servent à la sélection des registres internes du MC 68488 et les lignes restantes participent au décodage de l'adresse de la carte.

* La ligne AEN qui participe aussi au décodage de l'adresse de la carte.

* Les deux lignes IOR\ et IOW\ qui servent dans la logique de commande de l'écriture/lecture du MC 68488.

* La ligne CLK qui sera utilisée pour assurer la synchronisation et l'adaptation en fréquence du MPU et du MC 68488.

* La ligne RESET pour l'initialisation de la carte.

* La ligne IRQ pour une demande éventuelle d'interruption au microprocesseur par le MC 68488.

* La ligne RDY qui sera utilisée pour l'insertion des cycles d'attente afin d'établir des transmissions correctes entre la carte et le MPU.

2-2)- LES AMPLIFICATEURS DE BUS:

Le microprocesseur possède au niveau de ses lignes de sortie (données, adresses et contrôle) une sortance généralement faible. Pour qu'elle ne soit pas dépassée, tous les signaux issus du connecteur d'extension sont amplifiés par des circuits amplificateurs ou buffers trois états.

Ces buffers sont des circuits logiques inverseurs et/ou non inverseurs dont l'impédance d'entrée et la sortance sont trés élevées. Ils ont également pour tâche d'isoler le microprocesseur lorsqu'ils sont en haute impédance garantissant ainsi sa sécurité.

Les buffers utilisés dans la liaison MPU-MC68488 sont de deux types:

- Les buffers unidirectionnels pour les lignes d'adresse et de commande.
- Les buffers bidirectionnels pour les lignes de données puisque ces dernières transitent dans les deux sens.

a) Les amplificateurs du bus de données:

comme nous l'avons déjà avancé, ces buffers sont bidirectionnels et non inverseurs. Nous avons choisi pour cela le circuit 74LS245 qui est un amplificateur de bus bidirectionnel de 8 bits à sortie trois états.

Son activation est réalisée par le signal de sélection du boitier CS\ quant à son sens de transfert des données, il est assuré par le signal IOR\:

* une lecture du MC68488 correspond à DIR = IOR\ = 0.

* une écriture dans le MC68488 correspond à DIR = IOR\ =1.

La mise en place de cette logique de décodage impose la validation du 74 LS 245 en entrée ou en sortie, uniquement lorsque la carte est selectionnée et lorsque l'on envoie les instructions IN ou OUT par logiciel évitant ainsi la circulation des données dans l'interface en dehors de la sélection de celle-ci.

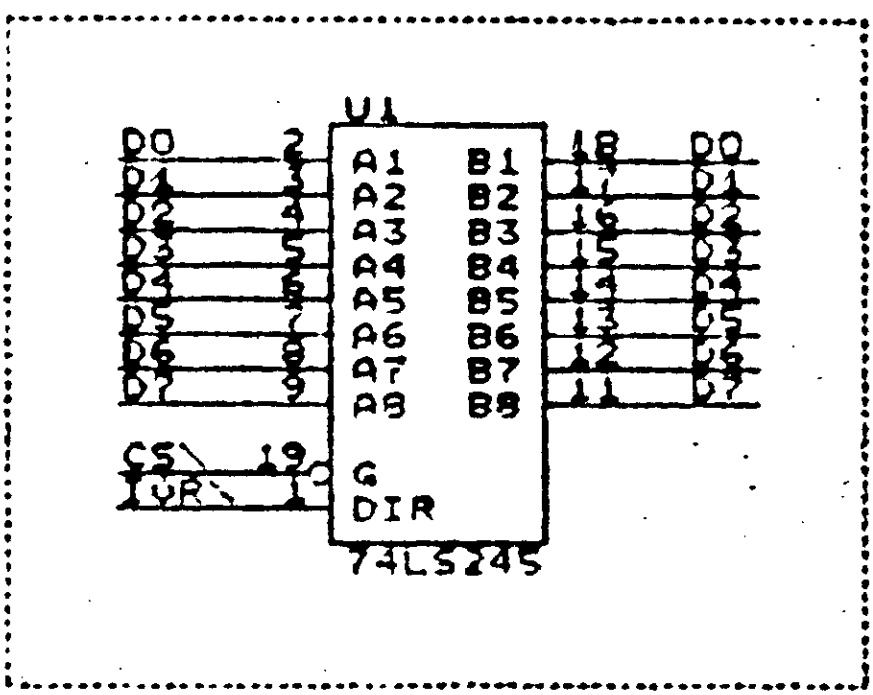


Fig.IV.2: **Amplificateurs du bus de données**

b)- Les amplificateurs du bus d'adresse:

L'adresse d'une carte d'entrée/sortie s'étend sur 12 bits (de A0 à A11). On a choisi pour adresser notre carte le domaine d'adresses 400_H-407_H .

On aura donc besoin de 12 buffers unidirectionnels, et on a choisi pour cela les deux circuits suivants:

- Le 74 LS 241 qui est un double amplificateur de bus unidirectionnel inverseur de 4 bits à sorties trois états.

- Le 74 LS 126 qui est un quadruple amplificateur de bus unidirectionnel non inverseur de 1 bit à sorties trois états.

Le 74 LS241 sera employé pour les lignes d'adresse (A3, A9 et A11) et sera activé par les signaux A10\ et AEN et le 74 LS 126 sera employé pour les lignes A0- A2, et A10 et sera validé en permanence par Vcc dans le sens de la réception (vers le MC 68488).

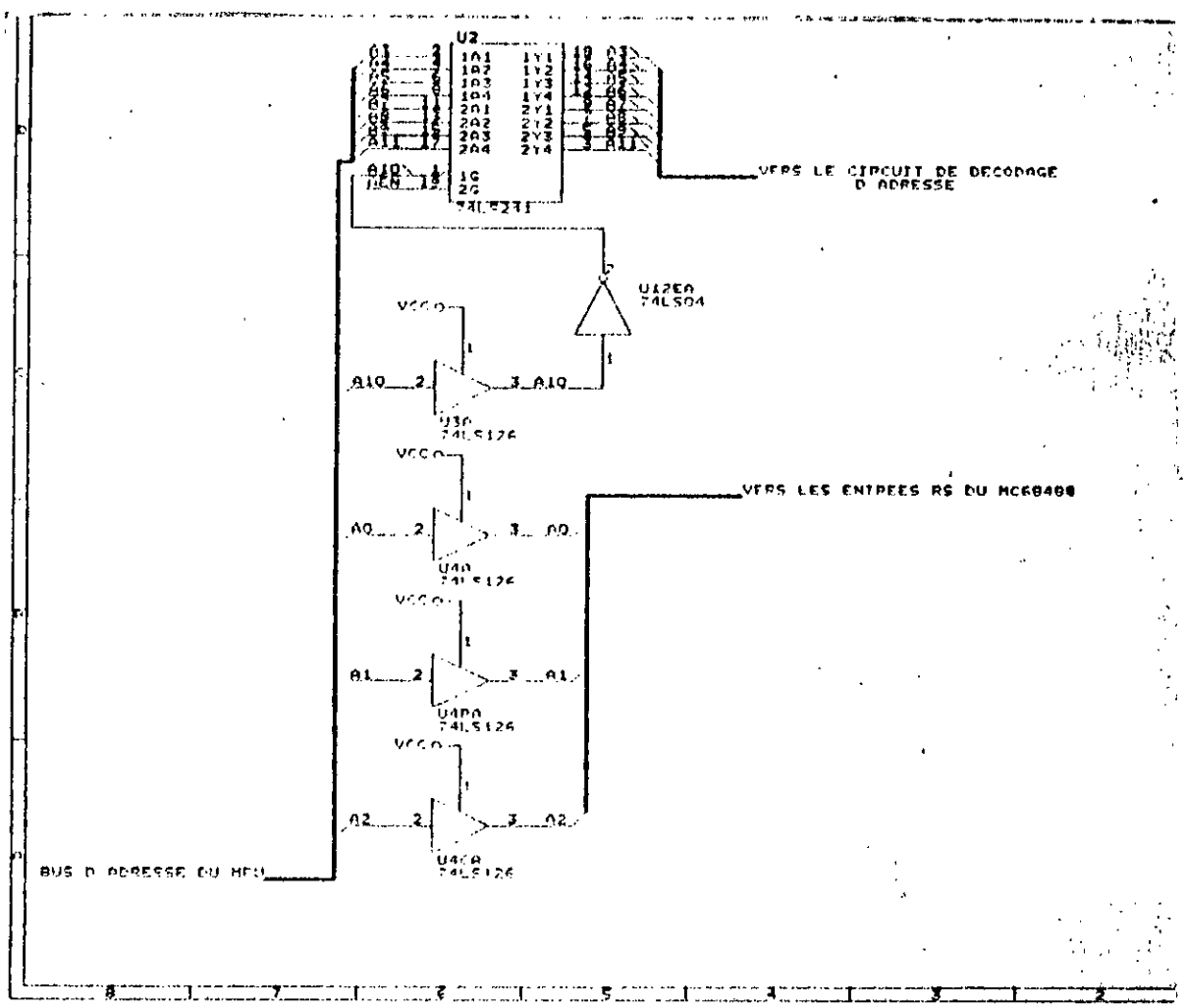


Fig.IV.3: **Les amplificateurs du bus d'adresse**

c)- **Les amplificateurs du bus de contrôle:**

Pour amplifier les signaux de contrôle (CLK, AEN, RESET, IOW\, IOR\, IRQ\ et RDY) on a choisi l'utilisation de 2 double amplificateurs de bus unidirectionnels non inverseurs de 4 bits à sorties 3 états(74LS126) et qui seront validés en permanence par Vcc sauf pour le buffer de la ligne RDY qui sera validé par la 1ere sortie du registre à décalage comme on le verra lors de la déscription du circuit à retard.

Note: Les signaux IRQ et RDY à l'opposé des autres signaux arrivent au connecteur d'extension.

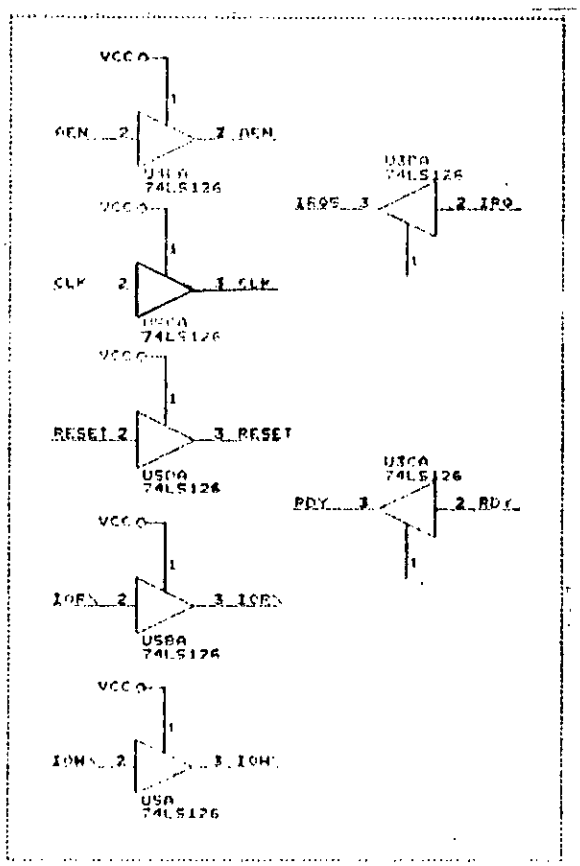


Fig.VI.4 : **Les amplificateurs du bus de contrôle**

2.3- LE CIRCUIT DE DECODAGE D'ADRESSE:

Les adresses de port des circuits auxiliaires et de certaines interfaces standard ont été fixées à l'origine par IBM et font donc partie des paramètres standard de la micro-informatique.

Le tableau suivant indique les adresses des ports attribués aux principaux composants matériels d'un AT:

CIRCUIT	ADRESSE
Contôleur DMA (8237A-5)	000-01F
Contrôleur d'intérruption (8259A)	020-03F
Temtorisateur	040-05F
clavier(8042)	060-06F
Horloge en temps réel(MC146818)	070- 07F
Registre de page DMA	080-09F
Contrôleur d'intérruption n°2(8259A)	0A0-0BF
Contrôleur de DMA n°2 (8237A-5)	0C0-0DF
Coprocesseur Arithmétique	0F0-0F1
Coprocesseur Arithmétique	0F8-0FF
Contrôleur de disque dur	1F0-1F8
Manette de jeux (Joysticks)	200-207
2nde Imprimante Parallèle	278-27F
2nde Interface Série	2F8-2FF
Carte de Prototype	300-31F
Carte Réseau	360-36F
1re Imprimante Parallèle	378-37F
Carte d'Ecran Monochrome et 1re Imprimante Parallèle	3B0-3BF
Carte vidéo couleur/graphique	3D0-3DF
Contrôleur de Disquette	3F0-3F7
1re Interface Série	3F8-3FF

Tableau IV.2: **Les adresses des ports des principaux composants matériels d'un AT.**

Pour affecter une adresse à notre carte on a eu le choix entre deux alternatives:

* La première consistait à lui affecter une adresse translatable pour qu'elle puisse être utilisée avec n'importe quel système déjà existant et lui éviter ainsi tout conflit avec les adresses des autres cartes.

* Mais le grand nombre de composants électroniques nécessaires à sa mise en oeuvre (micro-interrupteurs, comparateurs, portes logiques..) nous a amenés à lui préferer la deuxième: le choix d'une adresse figée qui tire profit de la standardisation des adresses des différents ports d'entrée/sortie et dont la mise en oeuvre ne nécessite qu'une seule porte logique: Le 74LS30 qui est une porte NAND à 8 entrées.

Rappelons ici que le décodage d'adresse ne fait intervenir que les ligne A0, A11 aprés leur passage à travers les amplificateurs de bus, que les 4 bits de poids sont figés par logiciel et que le domaine d'adresses choisi est 400H - 407H. Ce choix nous permet, en combinaison avec la ligne R/W\, d'adresser les 15 registres internes du MC 68488.

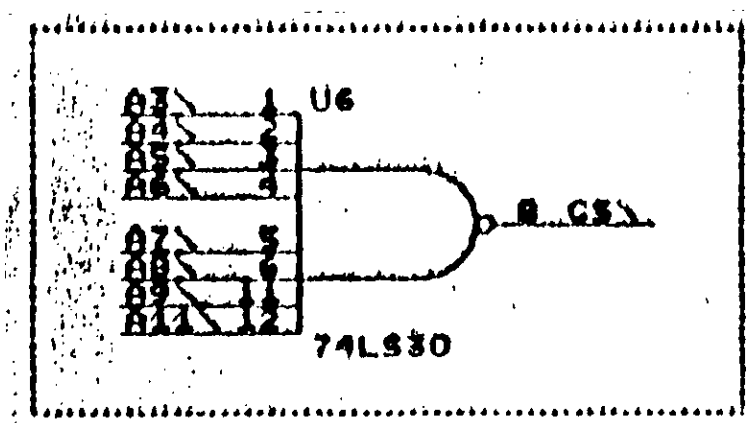


Fig.IV.5: **Le circuit de décodage d'adresse**

2.4)- LE DIVISEUR DE FREQUENCE:

Ce circuit utilise trois bascules JK (74LS107) pour adapter l'horloge système CLK de 8MHz à l'entrée horloge E du MC68488 dont la fréquence maximum est de 1 MHz.

Chaque bascule a ses entrées J et K au niveau 1; on sait que dans ce cas, la sortie de la bascule change d'état à chaque fois que l'impulsion d'horloge passe du niveau haut au niveau bas. Mais comme l'horloge système est active quand l'impulsion passe du niveau bas au niveau haut, l'emploi d'un inverseur s'avère indispensable.

Le train d'impulsions d'horloge est appliqué seulement à l'entrée CLK de la bascule B_0. La sortie de cette bascule est amenée à l'entrée CLK de la bascule B_1 dont la sortie elle-même amenée à l'entrée CLK de la bascule B2 . Enfin, la sortie inversée de B2 est reliée à l'entrée horloge E du MC 68488.

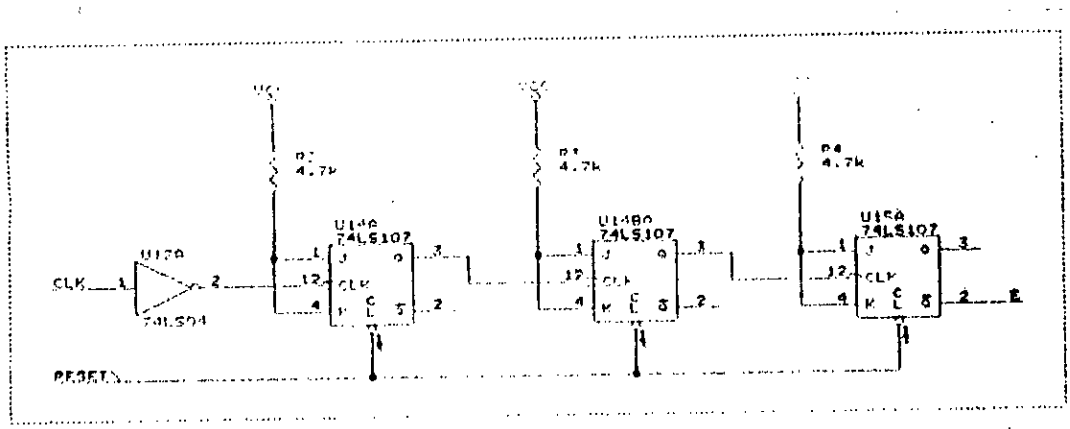


Fig.IV.6 : **Le diviseur de fréquence.**

2.5)- LE CIRCUIT DE SELECTION DE L'ADRESSE PHYSIQUE:

La sélection de l'adresse physique de la carte sur le bus est réalisée par le circuit ci-dessous:

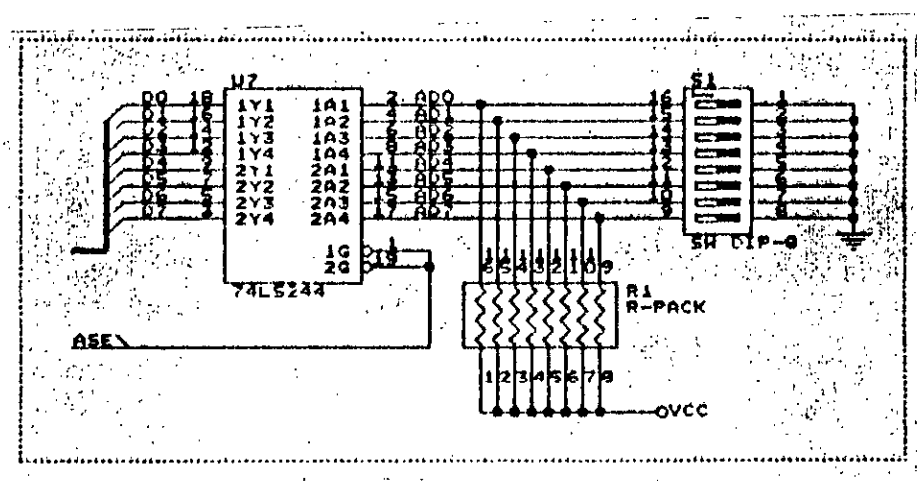


Fig. IV.7: **le circuit de sélection de l'adresse.**

En plus d'un boitier de 8 micro-interrupteurs servant à positionner manuellement cette adresse sur le bus, ce circuit comporte un amplificateur de bus unidirectionnel non inverseur de 8 bits à sorties 3 états (le 74 LS244) dont les sorties sont reliées au bus de données du MPU, permettant ainsi à ce dernier de lire l'information d'adresse et de la communiquer au MC 68488 chaque fois que cela est nécéssaire.
L'activation du 74LS244 se fait par le signal ASE\ issu du MC 68488 et le fonctionnement a été déjà expliqué. L'adresse assignée à la carte sur le bus est l'adresse 20 (10100).

2.6 LE CIRCUIT A RETARD:

L'interfaçage du MC 68488 au connecteur d'extension pose le problème de compatibilité en vitesse des deux systèmes: d'une part, on a un système rapide (le MPU) qui génère un signal d'horloge de 8 MHz et de l'autre, on a un circuit lent (le MC 68488) qui ne peut suivre des transitions d'une horloge de plus de 1 MHz.

L'utilisation du signal RDY fourni par le bus d'éxtension peut pallier à ce problème. En effet, ce signal, généré par la carte à partir des signaux IOR\, IOW\, CS\, CLK\, RESET\, sert à retarder le MPU d'un nombre déterminé de cycles d'horloge.

Le circuit à retard est conçu à base de deux registres à décalage 8 bits à entrées séries et sorties parallèles (74LS164) autour desquels sont disposés des circuits logiques.

La bascule 74LS74 qui est une bascule D, a son entrée CLK reliée à un circuit qui génère une transition bas-haut lorsque la carte est selectionnée (CS\=0) et un ordre de lecture ou d'écriture est donné (IOR\ ou IOW\ = 0). Et comme son entrée D est à l'état haut en permanence, cette bascule change d'état à chaque transition bas-haut. Sa sortie, quand à elle, est reliée aux entrées séries du registre à décalage. La sortie de ce dernier valide le buffer 74LS126 débutant ainsi la génération du signal de retard RDY.

Le décalage se fait au rythme de l'horloge système à 8MHz et le choix du nombre de cycle de ce retard s'étend de 1 à 15 cycles ce qui rend indispensable la mise en cascade d'un second registre à décalage. Une fois que ce nombre est choisi, la sortie parallèle correspondante est reliée à un inverseur qui agit sur les entrées de réinitialisation des différents circuits terminant ainsi la génération du signal RDY.

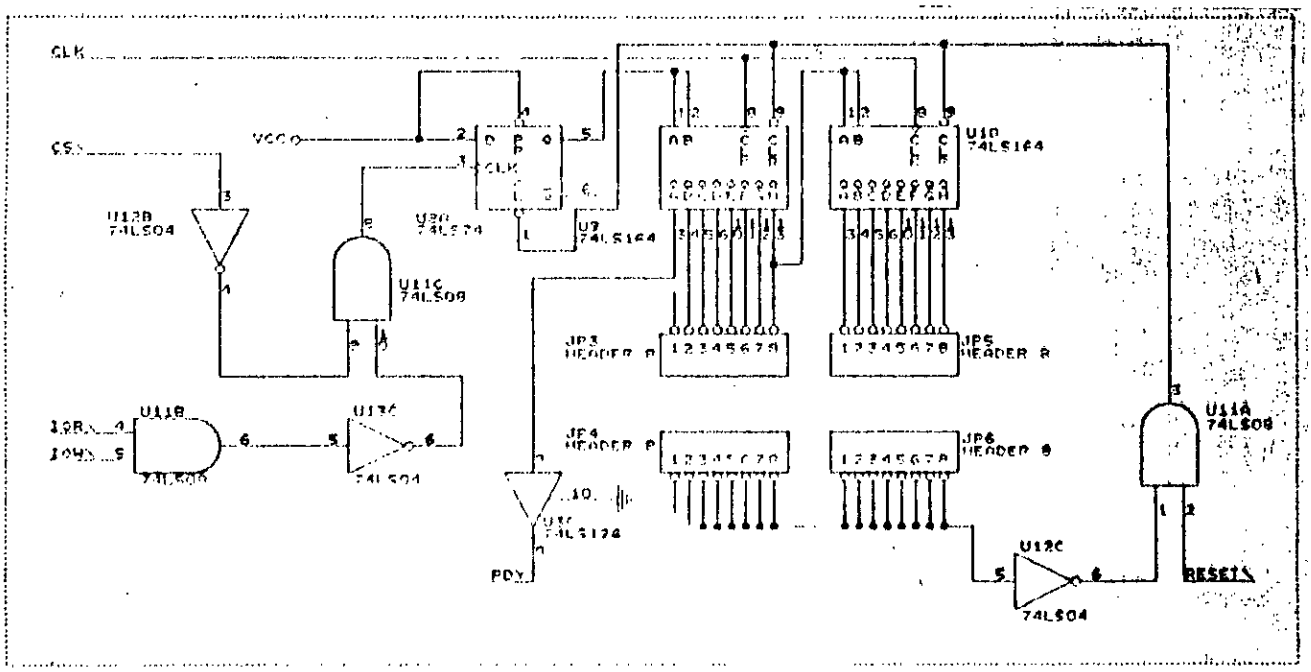


Fig. IV.8: **Circuit à retard**

2.7)- LE MC 68488:

C'est le circuit de base de notre carte. Il a été étudié en détail dans le précédent chapitre.

2.8)- LES TRANSCEIVERS DE LIGNES:

Pour assurer l'adaptation du MC 68488 aux lignes du bus IEEE et répondre aux conditions de fonctionnement imposées par la norme IEEE-488 (logique négative, possibilité de fournir 48mA à l'état bas lorsque la ligne fonctionne en driver et assurer la bidirection de presque la totalité des 16 lignes du bus IEEE), nous devons insérer sur les 16 lignes du bus des transceivers (émetteurs/récepteurs) bidirectionnels non inverseurs.

Il existe des circuits "drivers" de lignes totalement conformes à la norme: INTEL 8293; TEXAS SN 75160/ 161, MOTOROLA MC 3441-3446/47/48.

Nous avons utilisé quatre circuits MC 3448A pour plusieurs raisons: parce qu'ils sont les seuls disponibles d'une part, d'autre part, parce qu'ils optimisent l'espace sur la carte et par voie de conséquence le câblage du circuit imprimé. Enfin, parce qu'ils sont parfaitement adaptés à nos besoins à savoir: la possession des entrées de validation pouvant commander la direction ou le sens de transfert des données sur chaque ligne du bus IEEE.

Le MC 3448A est un circuit à 16 broches qui renferme 4 transceivers trois états ce qui justifie l'utilisation de quatre boitiers MC 3448. Pour les lignes à double sens, la direction de l'information est commandée par les lignes T/R\1 et T/R\2 issues du MC68488. Quand aux lignes à sens unique, la direction est commandée soit par un potentiel fixe (+ 5V) soit par la masse.

Deux circuits MC3448A pilotent les 8 lignes de données bidirectionnelles. Deux autres attaquent les lignes de gestion de protocole (RFD,DAC et DAV\ ) ainsi que les lignes de contrôle (EOI\, IFC\, REN\, ATN\ et SRQ\ ).(voir la figure 10 montrant la structure interne du MC3448A).

L'émetteur et le récepteur de chaque canal est valide par son entrée Send/Receive correspondante et il est invalidé lorsque celle-ci est mise en haute impédance. Le schéma de la **figure IV.11** montre la validation des différents signaux:

* **DAV** et les signaux IB0\-IB7\ sont validés par T/R\2

Nous avons éffectué une double inversion de la ligne T/R\2 avant de la relier aux lignes précédentes pour éviter le problème d'une sortance faible qui mènera à un mauvais fonctionnement sur le bus.

* **DAC et RFD** sont validés par la ligne T/R\2 inversée. Ils sont sortants quand le MC68488est écouteur et entrant quand il est parleur.

***ATN\, IFC\ et REN** sont validés par la masse. Ils sont toujours entrants au GPIA. Ils lui arrivent du bus.

* **EOI** est validé par T/R\1. Il circule dans le même sens que les données lorsqu'il signifie END (fin) et en sens inverse lorsqu'il signifie Identify (Identification). C'est pour cela qu'il lui faut une commande particulière.

* **SRQ** est validé en permanence par + 5V. Ce signal est toujours sortant afin que des demandes de service soient envoyées au contrôleur du bus.

Note: Une option supplémentaire permet aux sorties des émetteurs (drivers)d'être opérationnels en mode collecteur ouvert. Pour ce faire les broches EAB et ECD seront mises à la masse.

2.9)- LE CONNECTEUR IEEE:

Le connecteur utilisé est celui défini par la norme IEEE-488 (24 broches). Il a été étudié en détail lors de l'étude des caractéristiques mécaniques de la norme.

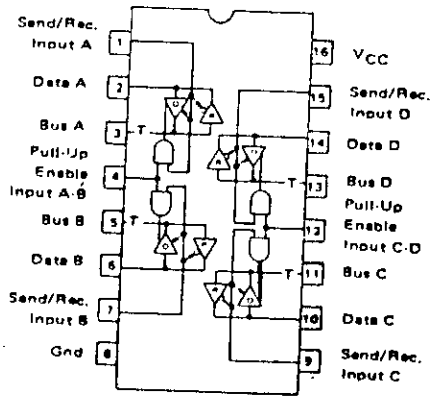


Fig.IV.9: MC 3448A

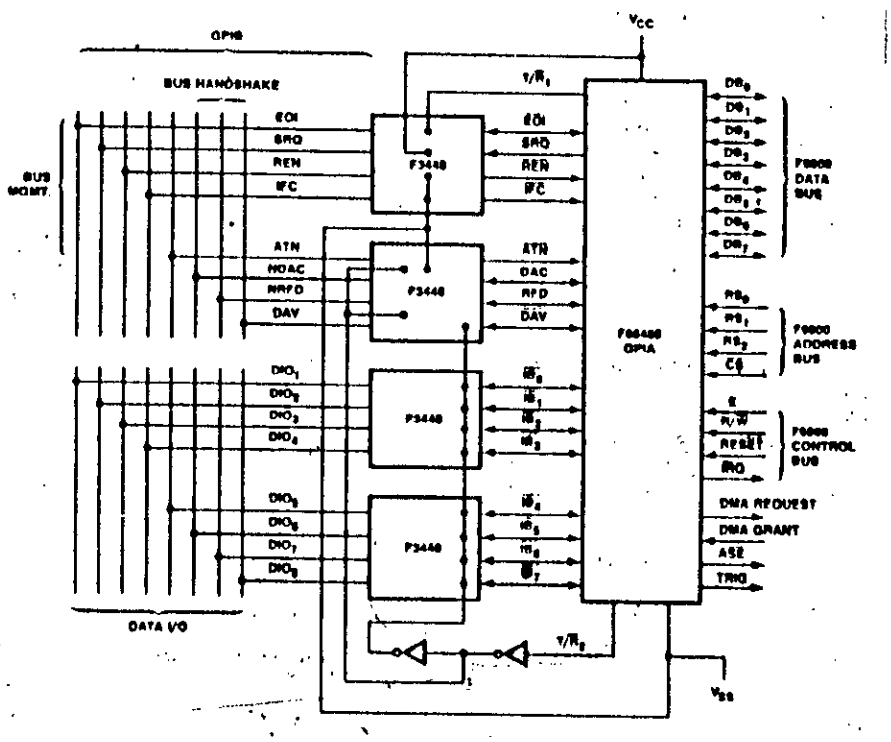


Fig.IV .10: La transceivers de ligne

2.10)- REMARQUES GENERALES:

1- On trouvera en annexe le schéma éléctrique global de la carte, le circuit imprimé et toutes les informations nécéssaires la concernant.

2- La carte, telle qu'elle a été conçu présente les défauts majeurs de la lenteur du traitement des données, de la non autonomie et de la non possession de la fonction C.
Ces trois défauts proviennent du MC 68488 lui-même:

- C'est lui dont le support maximum en fréquence est 1 MHz.

- C'est également lui qui fait appel au MPU pour réaliser les fonctions qu'il n'implémente pas seul.

- Et c'est lui, enfin qui ne possède pas du départ, la fonction contrôleur.

Pour pallier à ces défauts, on a envisagé des solutions mais des contraites techniques de réalisation (dimensions de la carte limitées; nombre très élevé des circuits intégrés nécessaires; encombrement sur la carte; pistes fines...) nous ont amenés à présenter la carte sous sa forme la plus optimale et la plus allégée que possible.

Ces solutions peuvent se résumer dans ce qui suit:

* L'implémentation d'une circuiterie permettant d'assigner la fonction contrôleur au MC 68488 et qui génère les signaux ATN\, REN\ et IFC\.
Pour le faire on aurait besoin d'un latch (permettant de générer ces signaux à des instants précis des échanges par logiciel) et d'une circuiterie annexe pour la validation des données en entrée et des données en sortie.

* L'implémentation aussi d'un microcontrôleur permettant au MPU de se décharger des tâches de gestion de l'interface et à l'interface d'être autonome du MPU.

* L'ajout d'une EPROM pour le stockage du programme de gestion. Sa réalisation nécéssiterait, en plus d'une EPROM, un boitier de micro-interrupteurs et un comparateur pour son adressage.

CHAPITRE V: ELABORATION DU DRIVER IEEEDRV

CHAPITRE V: ELABORATION DU DRIVER IEEEDRV

Il existe deux moyens pour programmer la carte IEEE-488 et la rendre opérationnelle:

* Le premier est direct et consiste en la programmation directe du MC 68488 et ses différents registres internes.

* Le second est indirect et consiste en l'intégration d'un driver, qu'on appelera IEEEDRV, qui isole le niveau logiciel supérieur des caractéristiques physiques de la carte en prenant en charge les commandes matérielles (voir Fig. V.1).

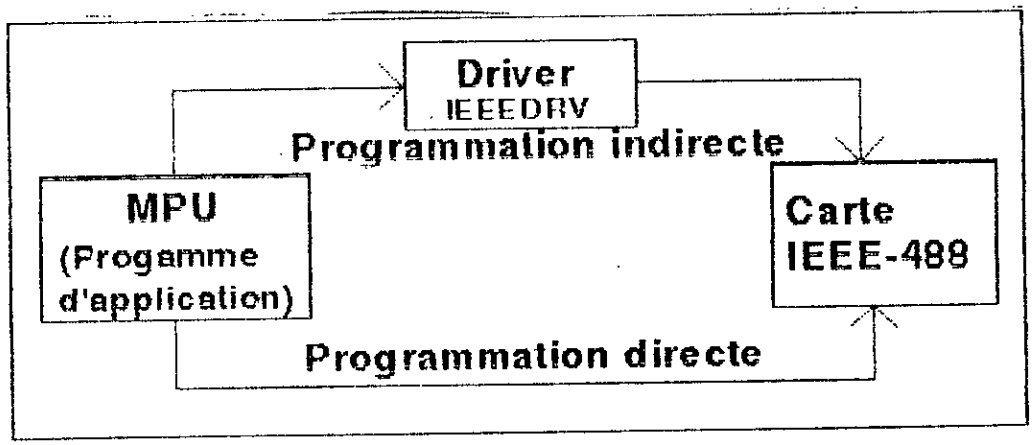


Fig. V.1: **Les deux types de programmation de la carte.**

Notre choix s'est porté sur la seconde méthode qui est, de loin, celle qui possède le plus d'avantages.

Les fonctions du DOS permettent d'accéder à la carte, elles couvrent toutes les tâches que nécéssite la communication avec elle. Il n'y a donc aucune raison de programmer directement l'électronique. Le programmeur évite ainsi beaucoup de peine car programmer un appel de fonction est en général infiniment plus simple que de prévoir un accés direct au matériel.

En effet, l'élaboration d'un driver offre plus de souplesse du moment que n'importe quel utilisateur peut l'exploiter en ecrivant des programmes en language éévolué de son choix en formulant seulement des instructions d'appels du driver.

L'utilisateur ne se soucie pas, par exemple, du lieu où il doit écrire une donnée ou de celui d'où il va récupérer une autre. Il n'a qu'à formuler des instructions de lecture ou d'écriture et le driver le fera à sa place.

1)- DEFINITION D'UN DRIVER:

Un driver est programme qui permet à DOS d'accéder à un périphérique (une carte) et le (la) commander. Il réalise une interface entre un niveau logiciel supérieur et un niveau matériel.

Si on considère un système d'exploitation comme une superposition de plusieurs couches, le driver constitue la couche la plus profonde qui permet à toutes les autres de fonctionner indépendamment du matériel.

Pour adapter un système d'exploitation à divers environnements matériels, le driver présente des avantages considérables car il est le seul concerné par les ajustements.

2)- STRUCTURE D'UN DRIVER:

Un driver se compose de quelques informations d'état qui donne à DOS des indications sur son type et ses capacités, et d'une série de routines appelées "Fonctions du driver". Ces dérnières prennent en charge les différents services dont DOS a besoin pour accéder au périphérique commandé. C'est ainsi qu'un driver d'une carte IEEE-488 devra disposer des fonctions d'initialisation, de lecture et d'écriture.

3)- DEVELOPPEMENT D'UN DRIVER:

La comminucation entre DOS et le driver repose sur un schéma simple d'appels de fonctions et de structures de données. C'est pourquoi le développement d'un driver se fait obligatoirement en assembleur.

Malgré la difficulté relative que présente la programmation en assembleur, elle offre, en contre partie, une vitesse d'éxécution maximale, ce qui est important pour un driver.

4)- TYPES DES DRIVERS:

DOS distingue deux types de drivers:

* Les drivers de caractères.
* Les drivers de blocs.

Les premiers communiquent, en effet, avec la matériel octet par octet. Les drivers de blocs, au contraire, peuvent transférer tout une série de caractères (bloc) en un seul appel de fonction.
Le driver IEEEDRV (comme ceux du clavier, de l'écran, de l'imprimante, du modem..) devra être un driver de caractères.

5)- INTEGRATION D'UN DRIVER:

Les drivers sont incorporés au système au moment du lancement du DOS. Ils ne peuvent être chargés au niveau de la ligne de commande de DOS comme les programmes. EXE et .COM ordinaires. Le "boot" commence par installer les drivers prédéfinis dans le noyau du DOS (NUL, CON, PRN, AUX, CLOCK, le driver du lecteur des disquettes et éventuellement, un driver pour le disque dur s'il est disponible). Ceux-ci sont rangés en mémoire directement à la suite les uns des autres et ils sont reliés par une sorte de chaine.

Pour installer le driver IEEEDRV, on doit l'indiquer à DOS dans le fichier CONFIG.SYS. Ce dernier est chargé et exploité au cours de l'opération de lancement du système, après l'intégration des drivers prédéfinis. Lorsque DOS y rencontre l'instruction DEVICE=IEEERDV, il saura qu'un nouveau driver doit être intégré et il le fait (voit fig. V.2)

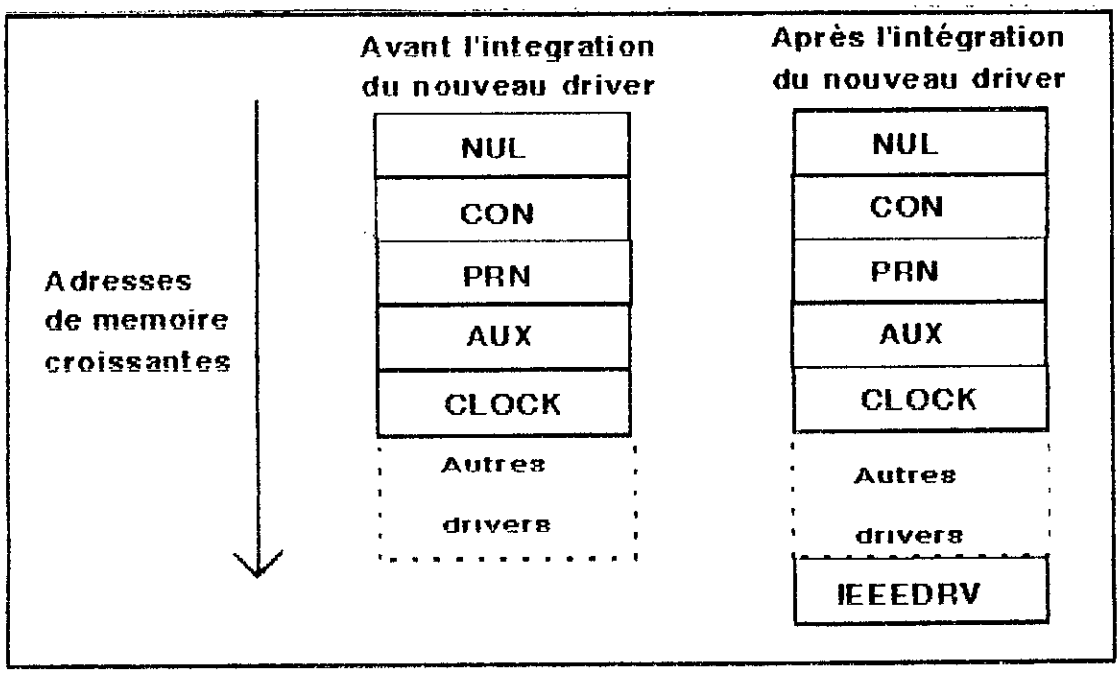


Fig. V.2: **Intégration de IEEEDRV**

6)- ACCES A UN DRIVER:

Il existe plusieurs possibilités pour accéder à un driver:

* Les fonctions habituelles gérant les FCB ou les handles peuvent parfaitement exploiter les drivers lorsqu'on y remplace un nom de fichier par un nom de driver.

L'accés évoqué n'est pas un accés direct car on se sert de différentes fonctions de DOS qui font appel au driver.

* Un autre moyen permet d'entrer en contact avec les drivers mais celui-ci est direct. Il s'agit de la fonction 44h de DOS appelée IOCTL (I/O Control) et qui joue un grand rôle dans ce domaine en offrant de nombreuses options.

7)- STRUCTURE DU DRIVER IEEEDRV:

Les drivers présentent tous une structure uniforme. Cette structure comporte:

* Un en-tête qui donne des informations générales.

* Deux routines qui gèrent toute la communication entre DOS et le driver. Les deux routines s'appellent routine de stratégie et routine d'interruption et doivent être du type FAR pour que DOS puisse aussi les appeler de l'éxtérieur du driver.

* Une série de routines, appelées "fonctions du driver" et qui prennent en charge les différents services dont DOS a besoin pour accéder au périphérique commandé.

Dans notre cas, cette série ne comporte que trois fonctions à savoir, l'initialisation, la lecture directe et l'écriture directe. (Voir figure V.3).

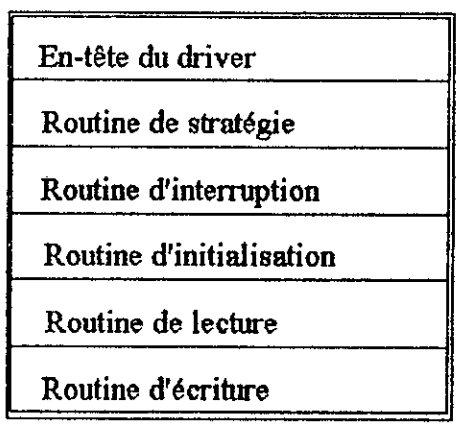


Fig. V.3: **Structure du driver IEEEDRV.**

a) L'en-tête de IEEEDRV: L'en-tête du driver contient certaines informations capitales pour permettre son exploitation par DOS. Par définition, cet en-tête est situé tout au début du drivers est présente la structure:

Adresse	Contenu	Type
+00h	Adresse du prochain driver	1 PTR
+04h	Attribut de périphérique	1 WORD
+06h	Offset de la routine de stratégie	1 WORD
+08h	Offset de la routine d'interruption	1 WORD
+0Ah	Nom du driver (complété par des espaces)	8 BYTES

Tableau V.1: **Structure de l'en-tête.**

- Le premier champ de cette structure établit la liaison avec le driver suivant dans la chaine des drivers. Il doit être initialisé avec la valeur (-1) pour que DOS reconnaise l'en-tête.

- Le second champ est un champ de bits qui sert à décrire le driver. Il indique entre autres à DOS son type. Cette indication figure dans le bit 15. Il est mis à 1, car IEEEDRV est un driver de caractères.

La signification des autres bits est décrite dans la figure suivante:

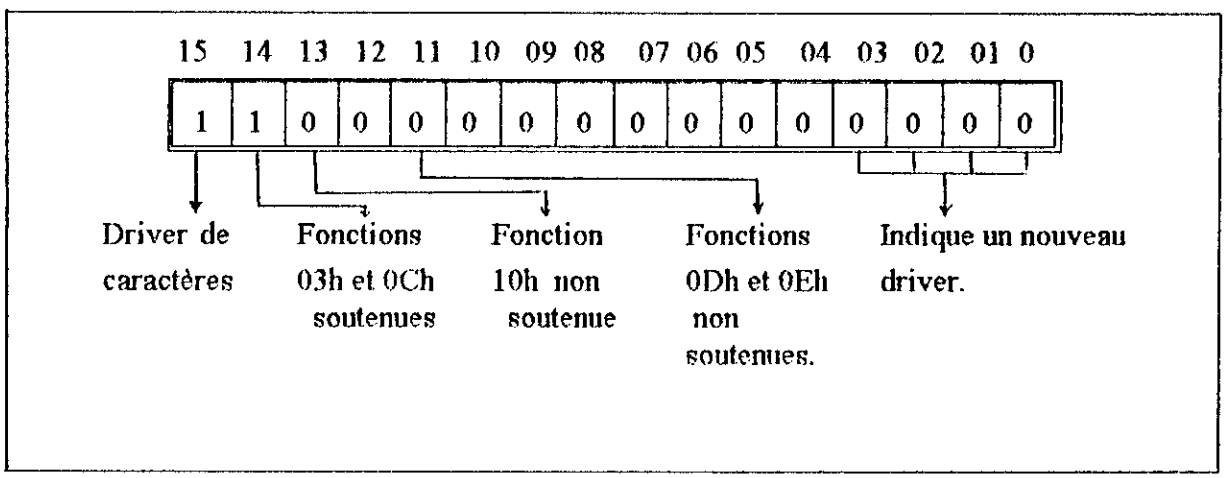


Fig. V.4: **Structure de l'attribut de périphérique dans l'en-tête du driver IEEEDRV**

* Les bits 0 à 3 servent à identifier le driver. On les a tous mis à zéro pour indiquer que IEEEDRV ne remplace aucun driver prédéfini mais constitue un driver additionnel ordinaire.

* Les bits 4 à 10 et 12 sont réservés et doivent être mis à 0.

* Le bit 11 est mis à 0 pour indiquer que les fonctions du driver 0Dh et 0Eh ne sont pas supportées.

* De même que le bit 13 qui indique le non support de la fonction 10h.

* Le bit 14 quand à lui est mis à 1 pour signaler que les fonctions de lecture directe (03h) et d'écriture directe (00h) sont supportées.

- Après l'attribut de périphérique, on trouve deux champs qui contiennent les offset des routines de stratégie et d'interruption. Ces routines désservent la communication entre DOS et le driver. .

- Dans le dernier champ on trouve enfin le nom du driver: IEEEDRV et comme ce nom comporte 7 caractères, on doit le compléter par un espace.

b) La routine de stratégie: La routine de stratégie est appelée par DOS dans deux circonstances:

* Lors de l'installation du driver

* Et avant tout appel de l'une des fonctions du driver.

Elle doit recevoir en ES: BX l'adresse d'une structure de données contenant les informations sur l'opération à éffectuer et les données correspondantes. Elle n'exécute pas elle-même ces opérations. Elle se contente seulement de stocker l'adresse du bloc de données transmis après quoi elle rend immédiatement le contrôle à DOS. (Voir FIg. V.6)
Le système appelle alors la routine d'interruption du driver qui se charge de l'éxécution de l'opération elle-même.

Derrière ce mécanisme se cache une idée très simple: DOS n'a pas besoin de connaitre les adresses de toutes les fonctions du driver, il lui suffit de se servir des routines de stratégie et d'interruption comme interface d'accés à ces fonctions. C'est pourquoi le bloc de données dont l'adresse est transmise à la routine de stratégie contient aussi le numéro de la fonction du driver à invoquer.

Ce bloc de données se compose en principe de 13 octets auxquels viennent s'en ajouter d'autes en fonctions des besoins de la fonction à appeler.

Le tableau de la page suivante montre la structure de ce bloc de données:

Adresse	Contenu	Type
+00h	Longueur du bloc de donnés en octets	1 BYTE
+02h	Numéro de la fonction à appeler	1 BYTE
+03h	Mot d'état	1 WORD
+05h	Réservé	8 BYTES
+0Eh	Adresse du buffer de transmission	1 PTR
+12h	Nombre d'octets	1 WORD

Tableau V.2: **Structure du bloc de données pour l'appel d'une fonction du driver.**

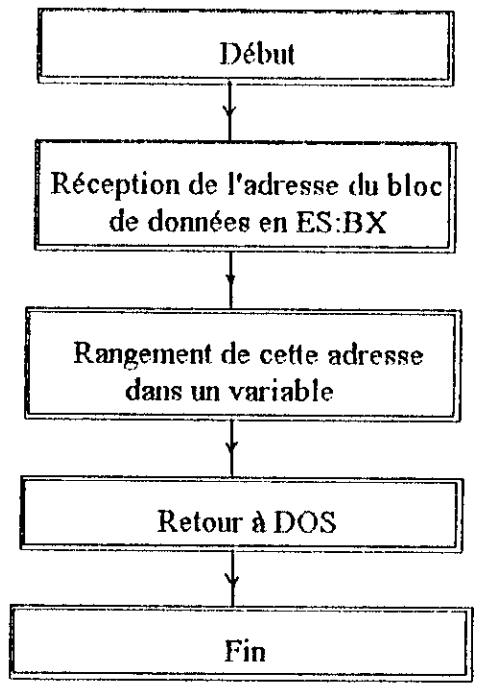


Fig. V.5: **Routine de stratégie**

c)- La routine d'interruption: La routine d'interruption est appelée immédiatement après que DOS ait déclenché la routine de stratégie.

La première tâche de cette routine consiste à sauvegarder (sur la pile) les registres du processeur qui risquent d'être modifiés par la suite. Elle lit ensuite dans le champ 3 du bloc de données transmis à la routine de stratégie, le numéro de la fonction à éxécuter, qu'elle appelle

alors sur le champ. Après éxécution de la dite fonction, elle inscrit dans le champ d'état du bloc de données et restaure ensuite les registres du processeur qui avaient été souvegardés au début. Elle rend enfin le contrôle à la fonction de DOS qui l'a appelée. (Voir fig. V.6)

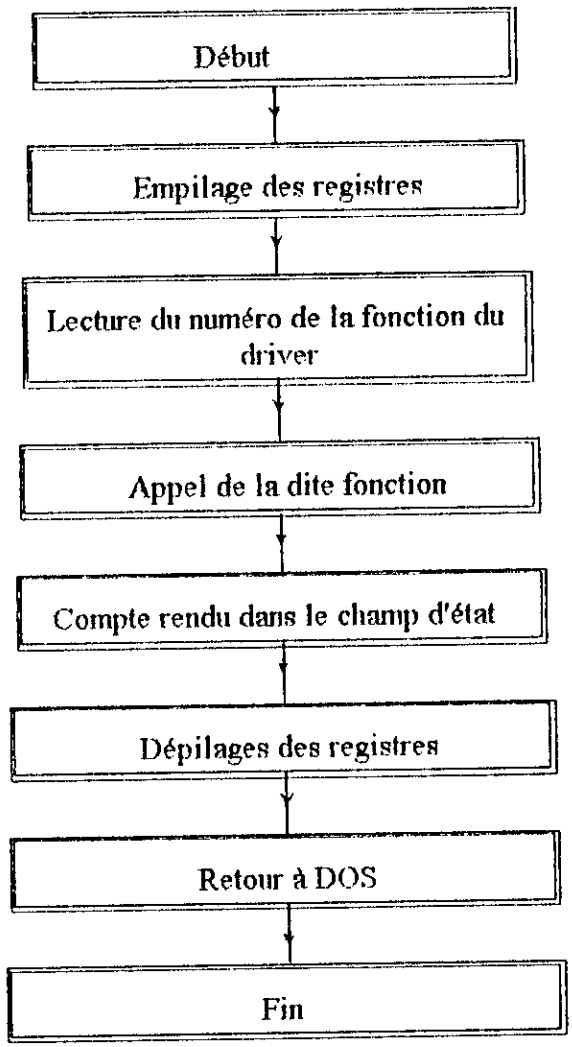


Fig. V.6: **Routine d'interruption**

Une fois la routine d'interruption arrivée à son terme, la valeur du champ d'état est très importante pour la fonction DOS appelante car elle indique si l'éxécution s'est bien passée. C'est pourquoi toute fonction du driver doit absolument mettre à 1 le bit TERMINE (bit 8) du champ d'état même si elle est fictive (dummy).

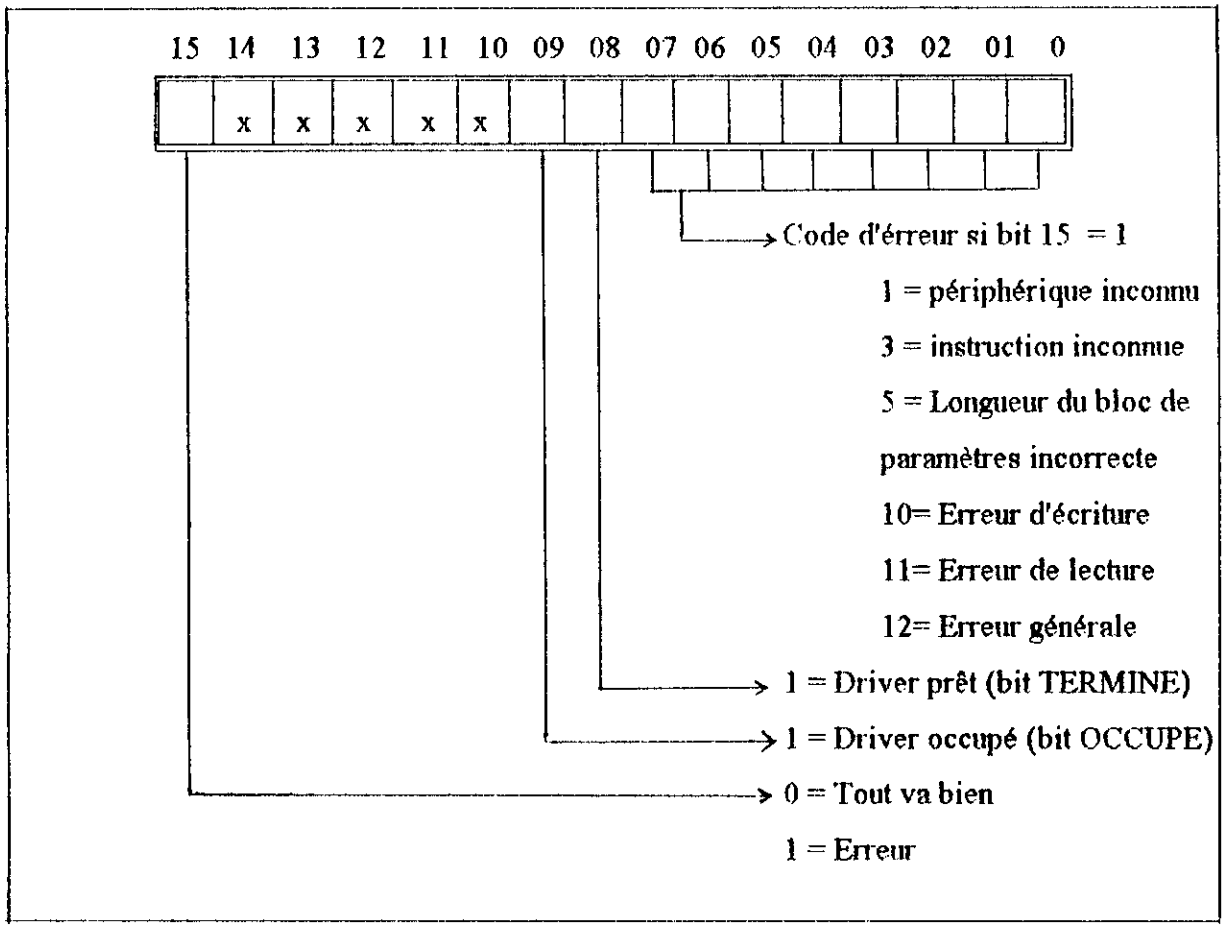


Fig. V.7: **Structure du mot d'état après appel d'une fonction du driver**

d)- Routine d'initialisation : Fonction 00h

Avant de décrire cette fonction signalons ces remarques à propos des différentes fonctions du driver:

* Tout driver à installer doit supporter 13 fonctions, numérotées de 0 à 12.

* Plusieurs autres fonctions s'y rajoutent mais leur support est facultatif.

* Toutes les fonctions mêmes celles qui ne servent à rien doivent au moins mettre à 1 le bit TERMINE au champ d'état.

* Chacune des fonctions tire ses arguments du bloc de données dont l'adresse est transmise à la routine de stratégie. C'est également dans ce bloc qu'elle placera ses "résultats".

Revenons maintenant à la fonction d'initialisation. Cette dérnière est appelée par DOS lors du chargement du système. Elle sert à initialiser le driver. L'initialisation concerne aussi bien le matériel que les variables internes du driver.

Et comme l'ensemble du système n'est pas encore entièrement initialisé à cet instant, la routine d'initialisation ne peut appeler que les fonctions 01h à 0Ch (entrées-sorties de caractères) et 30h (version du DOS) de l'interruption 21h de DOS. Ces fonctions lui permettent déjà de tester le numéro de la version de DOS et d'afficher à l'écran un message signalant que le driver est activé. Le nouveau driver n'est entièrement fonctionnel que lorsque la routine d'initialisation est achevée. La séquence des différents opérations est montrée dans la figure ci-dessous.

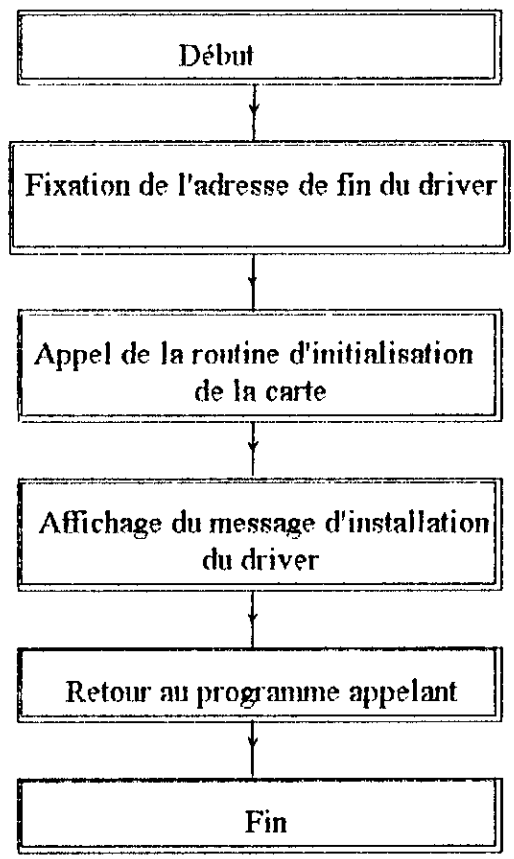


Fig. V.8: **La routine d'initialisation**

e)- La routine de lecture: Fonction 03h (lecture directe)

Cette fonction permet une communication directe entre un programme d'application et le driver. C'est cette fonction qui sera appelée par le moyen de la fonction 44h de l'interruption 21h de DOS car le bit 14 de l'attribut de périphérique est mis à 1.

- Elle attend qu'on lui transmette le nombre de caractères qui seront envoyés au programme d'application ainsi qu'un pointeur FAR sur le buffer où seront stockés les caractères. Le nombre de caractères lus doit correspondre au nombre de caractères transmis, sauf si une erreur survient qui empêche le traitement de l'intégralité des caractères.

La séquence des différentes opérations assurant la lecture de tous les caractères exigés est montrée dans la figure ci-dessous:

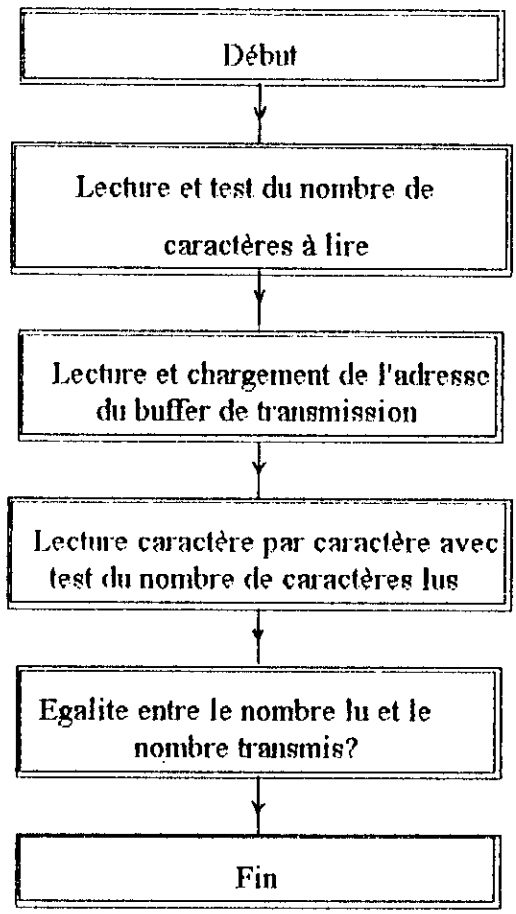


Fig. V.9: **La routine de lecture**

f) La routine d'écriture: Fonction 0Ch (Ecriture directe)

Cette fonction est la contrepartie exacte de la fonction 03h. Alors que cette dernière permettait aux programmes d'application de lire des caractères sur le driver, la fonction 0Ch éffectue un transfert en sens inverse. Un programme peut ainsi écrire directement sur la carte.

La structure de communication est à l'initaitive du driver car DOS n'impose aucune norme dans ce domaine. La figure suivante montre le séquencement des opérations qu'exécute cette routine:

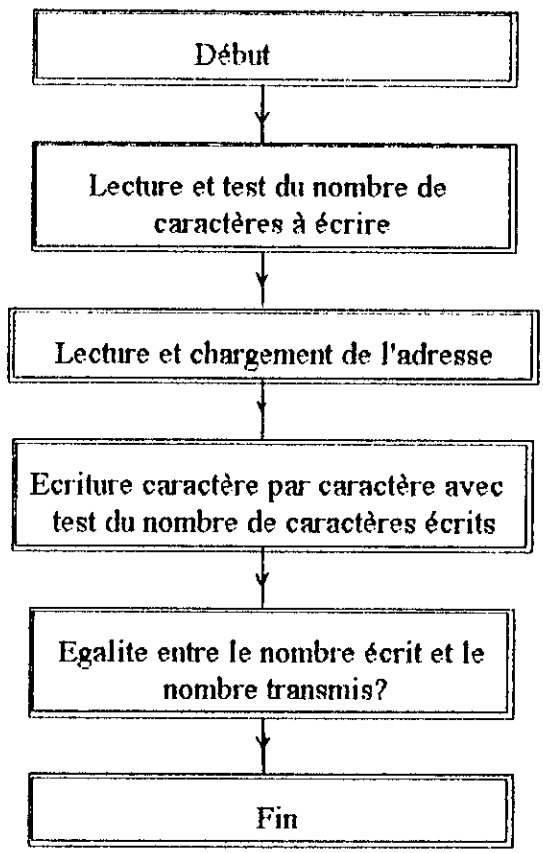


Fig. V. 10: **La routine d'écriture**

Remarque:

L'annexe D contient des tableaux exposant les paramètres d'entrées et de sorties des trois fonctions utilisées dans le driver (00h, 03h, 0Ch).

8)- FONCTIONNEMENT DU DRIVER: (Mode opératoire)

On s'intéresse ici à la séquence des opérations exécutées avant et après appel d'une fonction du driver.

* Le premier évènement de la chaine est un appel à une fonction de l'interruption 21h de DOS qui touche de près ou de loin aux entrées-sorties de caractères.

* Le fait d'invoquer une telle fonction peut avoir en conséquence toute une série d'autres appels qui se traduisent par un grand nombre d'accés en lecture/écriture.

* Après cela, DOS mettra en place, à l'intérieur d'une zone de mémoire réservée à cet effet, un bloc de données qui recevra les informations nécessitées par la fonction du driver.

* C'est alors que commence l'appel de la fonction du driver.

* La routine de stratégie du driver est déclenchée en premier et reçoit l'adresse du bloc de données qui vient d'être mise en place.

* C'est ensuite la routine d'interruption qui s'exécute.

Elle sauvegarde tous les registres et isole dans le bloc de données le numéro de la fonction à appeler, dont elle déclenche aussitôt l'éxécution.

* La fonction appelée doit simplement envoyer au matériel les caractères à transférer ou lui réclamer les caractères à lire.

* Une fois que la fonction de lecture/écriture a été éxécutée, la fonction du driver devra inscrire dans le champ d'état le résultat de son travail pour le communiquer à la fonction appelante.

* Le contenu de l'ensemble des registres peut alors être restauré et le contrôle est rendu à la fonction appelante.

* L'interruption 21h qui, suivant le résultat de la fonction du driver met à 1 ou à 0 l'indicateur de retenue et charge un code d'erreur dans le registre AX.

* Enfin, l'interruption 21 rend ensuite le contrôle à la fonction qui l'avait appelée

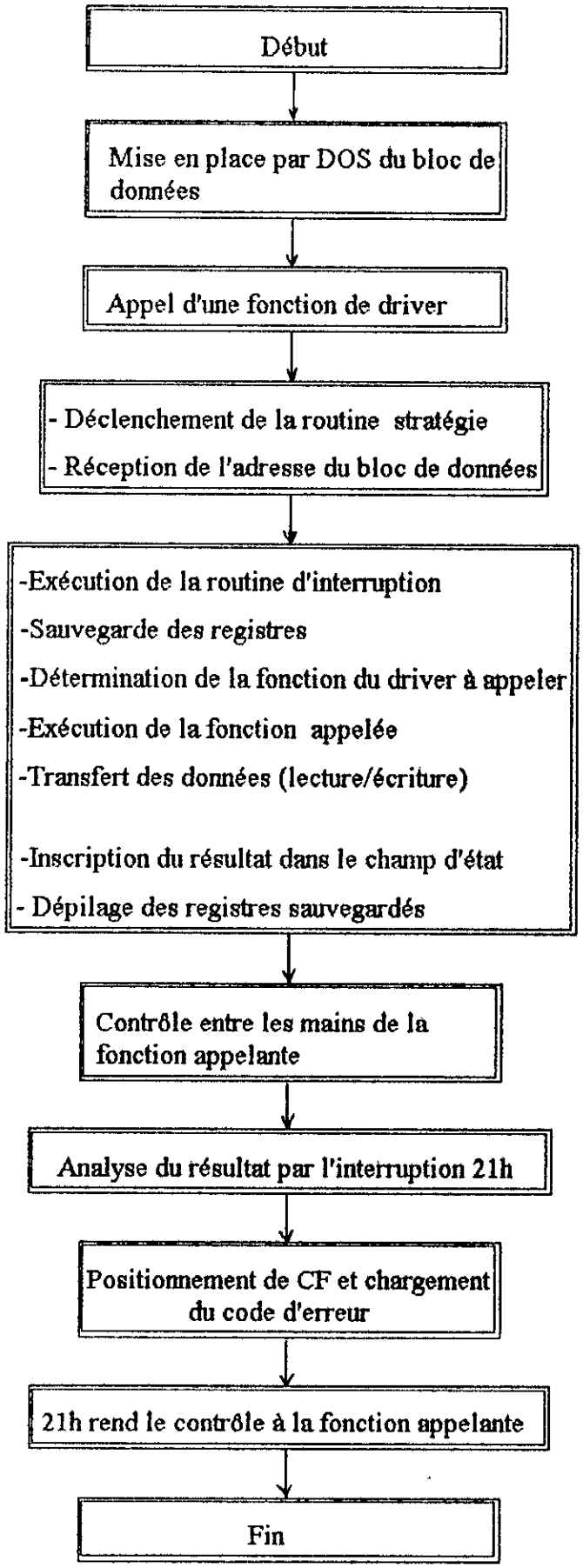


Fig. V.11: **Mode opératoire**

9)- DESCRIPTION DE IEEEDRV:

IEEEDRV, comme tout autre driver, respecte les règles des programmes.COM, n'occupe qu'un segment qui regroupe à la fois le code et les données et commence à l'offset 00h

- L'en-tête signale clairement qu'il s'agit d'un driver de caractères qui gère la carte fille IEEE-488. Ce driver, une fois intégré dans le système, sera mis en service à chaque appel aiguillé sur la carte IEEE-488.

- Le driver dispose naturellement d'une routine de stratégie et d'une routine d'interruption.

- La première sauvegarde simplemet l'adresse du bloc de données transmis dans la variable DB_PTR.

- La seconde quand à elle, lorsqu'elle est appelée commence par placer (sur la pile) les registres qu'elle va modifier, avant de lire dans le bloc de données transmis le numéro de la routine à appeler.

- Elle examine ensuite si cette fonction est supportée par IEEEDRV. Si ce n'est pas le cas, on saute directement à la fin de la routine d'interruption ou le code d'erreur approprié est enregistré dans le champ d'état du bloc de donnés transmis. Les registres qui avaient été sauvegardés sur la pile sont alors restaurés et le contrôle est rendu à la fonction de DOS qui avait appelé la routine.

- Si par contre la fonction appelée est bien supportée, l'offset de la routine demandée à l'intérieur du driver est lu dans la table TAB_FCT pour permettre son appel et son exécution.

- Si on examine la dite table, on constate que les noms des routines DUMMY et NO_SUP y apparaissent plusieurs fois.

* DUMMY est appelée pour toutes les fonctions qui ne sont pas définis dans ce driver. C'est pourquoi la seule fonction de la routine DUMMY est de mettre à 0 le registre AX et du même coup le bit OCCUPE dans le champ d'état. Cette disposition est imposée par les fonctions de niveau supérieur de DOS.

* La routine NO_SUP est, quand à elle, appelée à la place de toutes les routines qui ne doivent pas être déclenchées par les fonctions supérieures de DOS parceque IEEEDRV a indiqué dans son attribut de périphérique qu'il ne supporte pas ces fonctions.

* La routine suivante est la routine d'initialisation. Elle est la première appelée par DOS pour éxécuter les tâches suivantes:

- L'affichage du message d'initialisation du driver.

- L'initialisation de la carte IEEE-488.

Et puisqu'elle ne servira plus après ce premier appel, elle transmet sa propre adresse du début pour l'enregistrer dans le bloc de données transmis.

Les deux dernières routines sont celles de la lecture directe et de l'écriture directe que nous avons deja décrits.

Remarque importante: Pour accéder au driver IEEEDRV et l'exploiter, on a le choix entre deux méthodes comme on l'a déjà avancé:

- La première méthode consiste en l'utilisation des fonctions handles. En Turbo Pascal, ces fonctions peuvent être disposées en une unité utilisateur. Ceci facilite beaucoup les opérations sur le périphérique (ou la carte) car elle est traité comme un fichier.

- La seconde méthode consiste en l'utilisation de la fonction 44h de l'interruption 21h du DOS avec ses différentes options. Cette méthode se ramène aussi à l'élaboration d'une unité utilisateur qui contient les procédures de lecture directe et d'écriture directe.

CONLUSION :

L'étude que nous avons réalisée avait pour objectif initial d'aboutir à la réalisation d'une carte *IEEE*-488 à base du MC 68488 et à l'élaboration d'un driver permettant de piloter la dite carte.

Mais de nombreuses contraintes d'ordre technique (dimensions reduites de la carte, nombre élevé de circuits intègrés, qualité du circuit imprimé...) ont fait que nous n'avons pas réalisé des tests concluants concernant le fonctionnement du bus IEEE-488.

Pour faire un travail parfait, il faut, partant de notre conception des différents blocs de la carte et suggestions d'améliorations que nous avons avancées, utiliser des moyens et des techniques de réalisation plus développés et convenables à ce genre de projets.

Notons que l'utilisation du MC 68488 de Motorola n'est la meilleure solution pour réaliser une carte IEEE-488.

En effet, ce circuit présente par rapport à ceux des autres firmes des défauts majeurs à savoir la lenteur, la dépendance du MPU dans la réalisation de certaines fonctions, la non-possession de la fonction contrôleur et la nécessité d'un nombre beaucoup plus grand de composants électroniques.

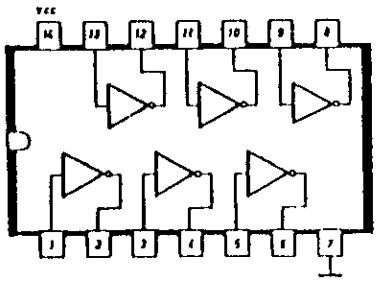
Ceci veut dire que notre carte aurait été plus performante si on envisage de la dôter d'une circuiterie supplémentaire pour pallier aux défauts précedemment décrits.

*L'*ajout, par exemple, d'un microcontrôleur rendra la carte autonome et indépendante du MPU et implémentera la fonction contrôleur ce qui va élargir le champ d'applications de la carte.

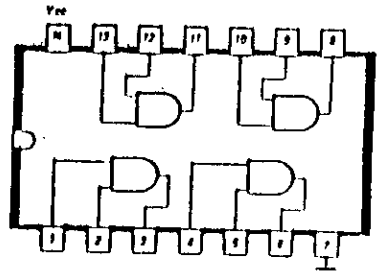
Notons enfin que même le driver est sujet à extension et amélioration. En effet, l'implémentation de la fonction contrôleur à la carte aura pour conséquence l'ajout d'un certain nombre de routines réalisant des fonctions telles que: Adressage, Configuration du bus, Initialisation des appareils .... et beaucoup d'autres fonctions de la norme IEEE-488.

ANNEXES

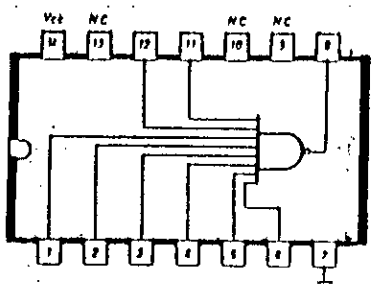
ANNEXE A: Circuits intégrés 74LS utilisés



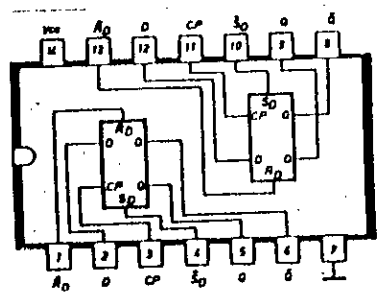
74LS 04



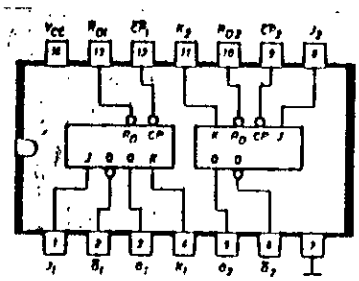
74 LS 08



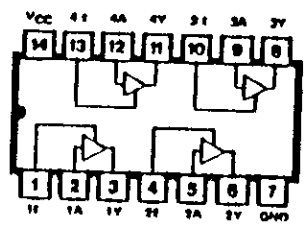
74 LS 30



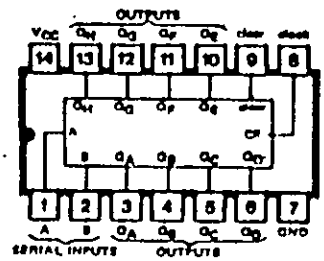
74 LS 74



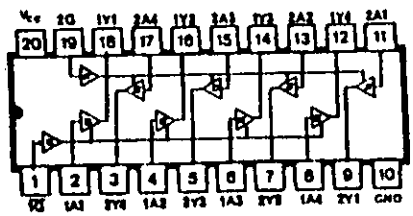
74 LS 107



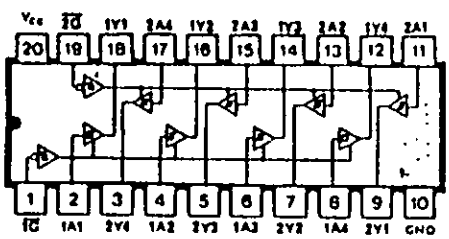
74 LS 126



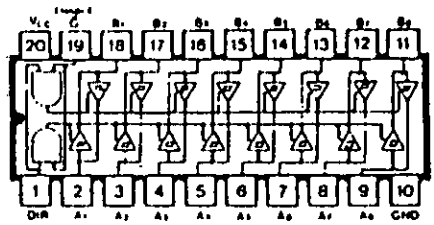
74 LS 164



74 LS 241



74 LS 74 244



74 LS 245

ANNEXE B: Les schémas

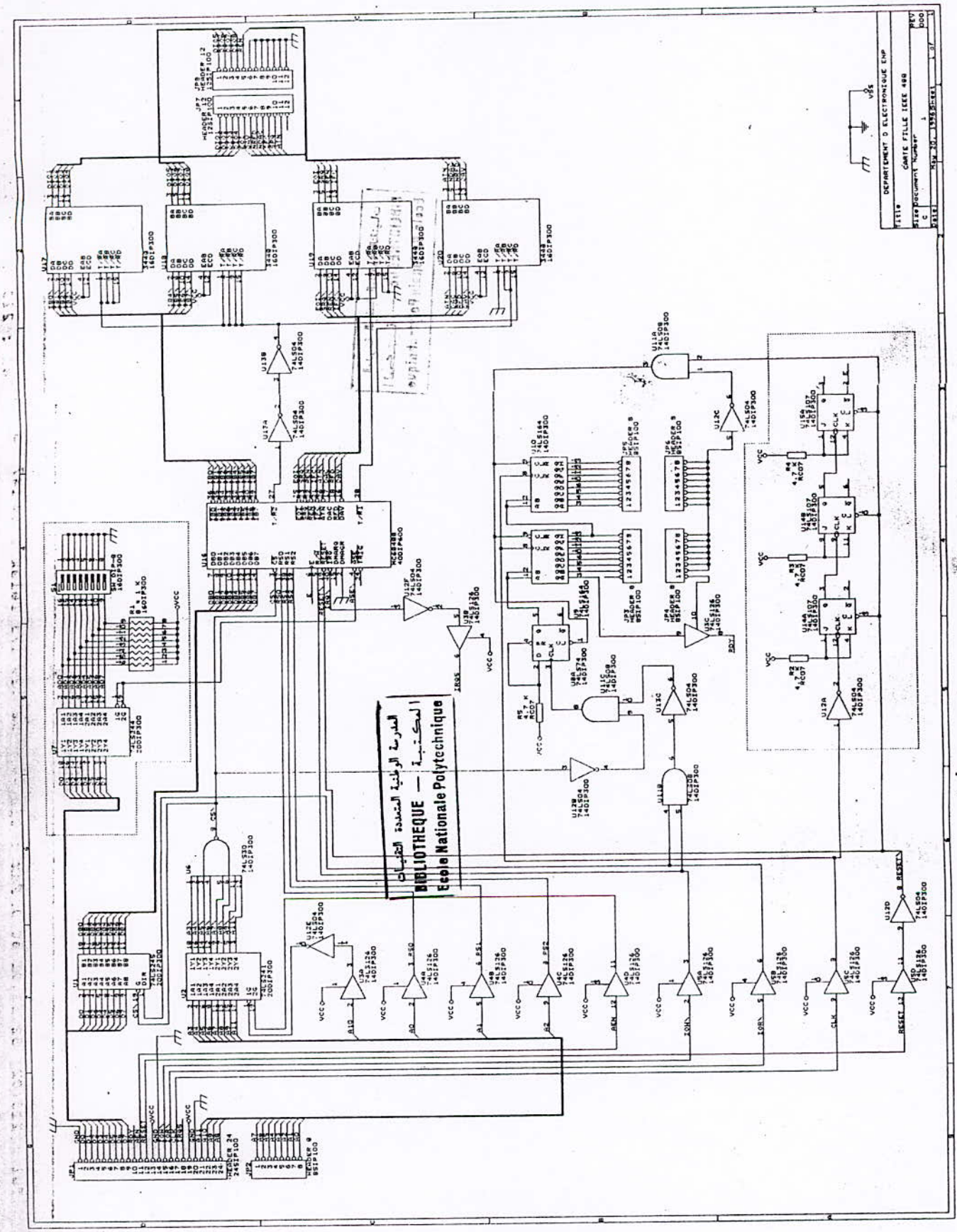


Fig. B.1: SCHEMA ELECTRIQUE DE LA CARTE IEEE-488

Fig. B.2: CIRCUIT IMPRIME (Face 1)

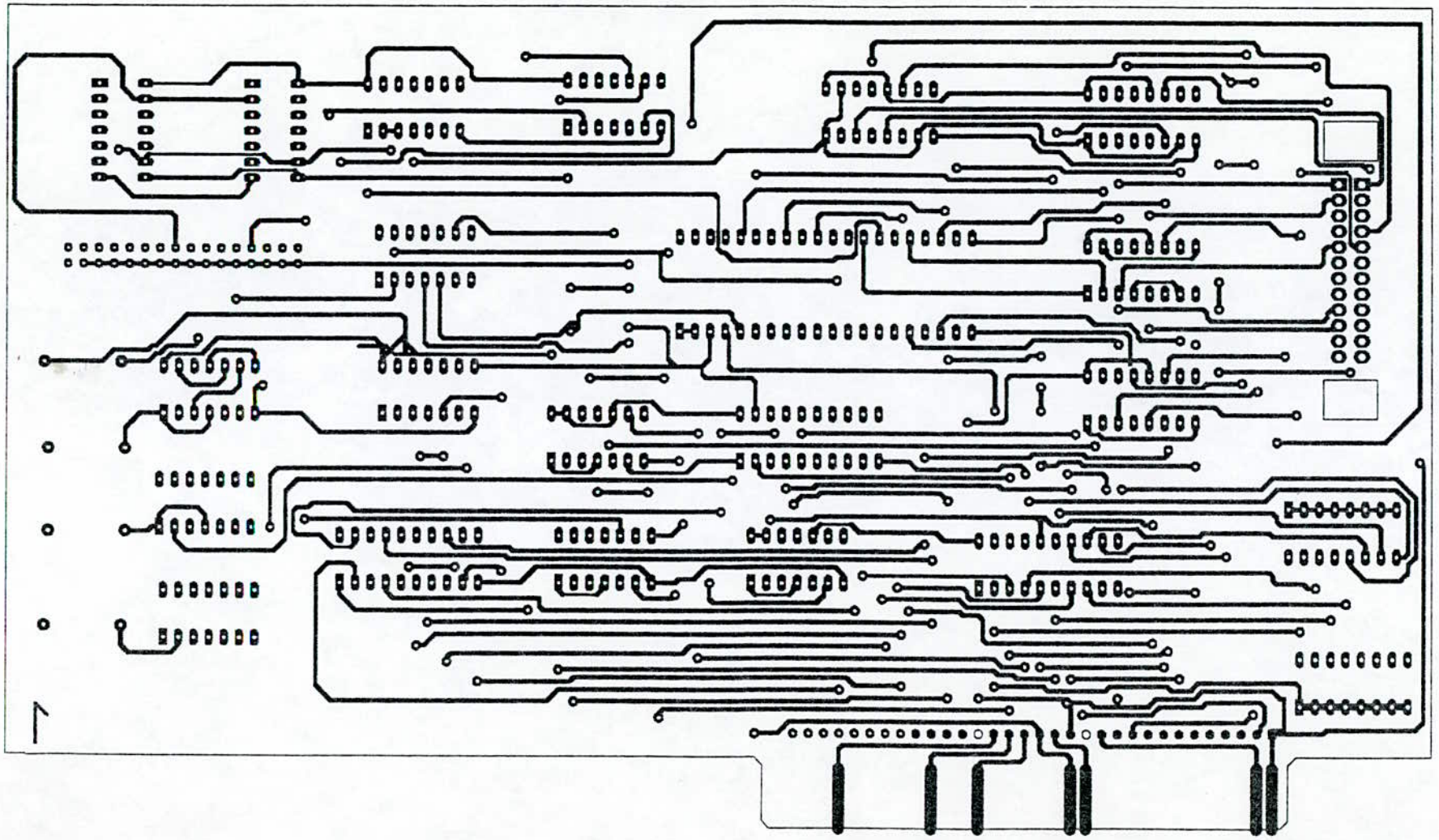


Fig. B.3: CIRCUIT IMPRIME (Face 2)

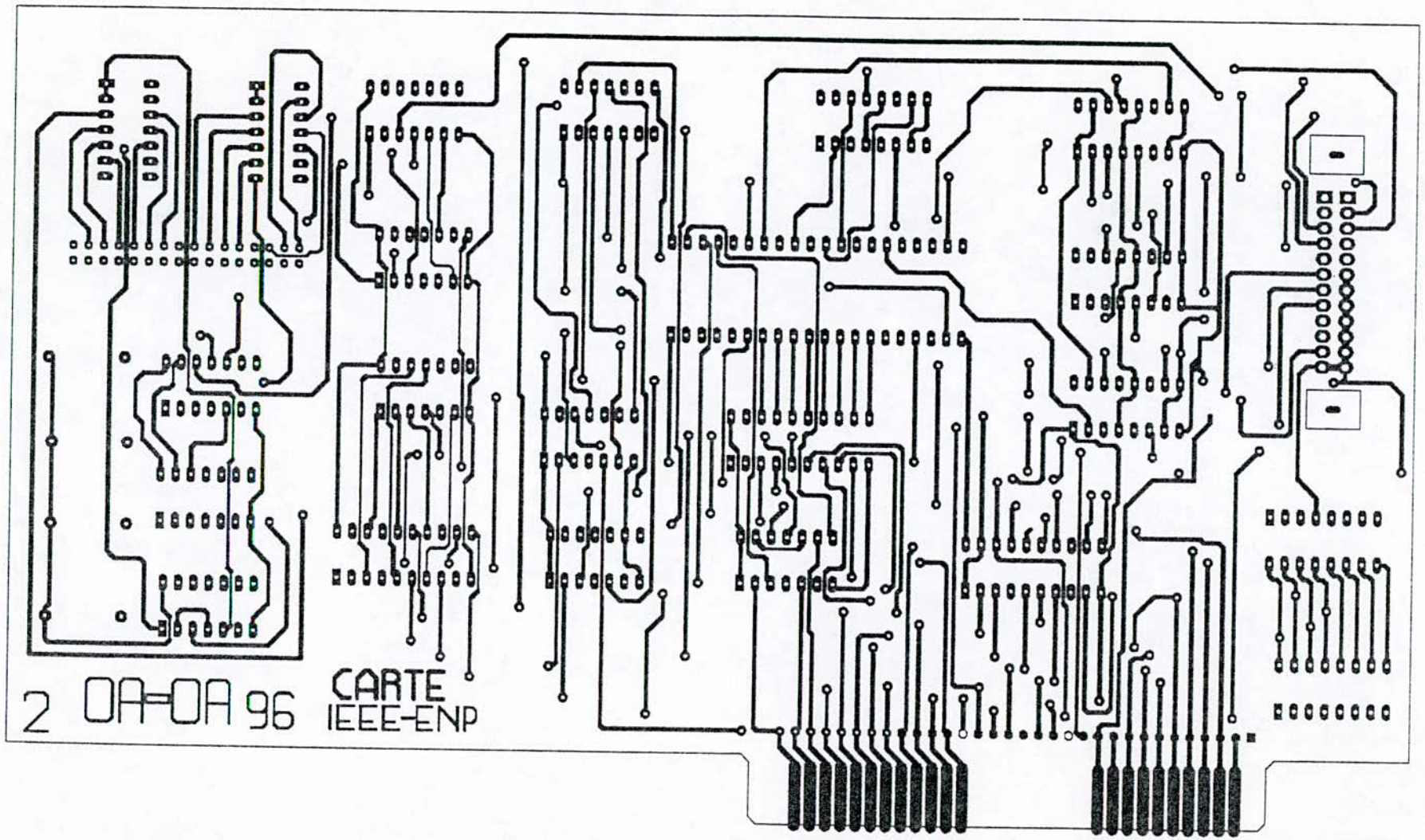
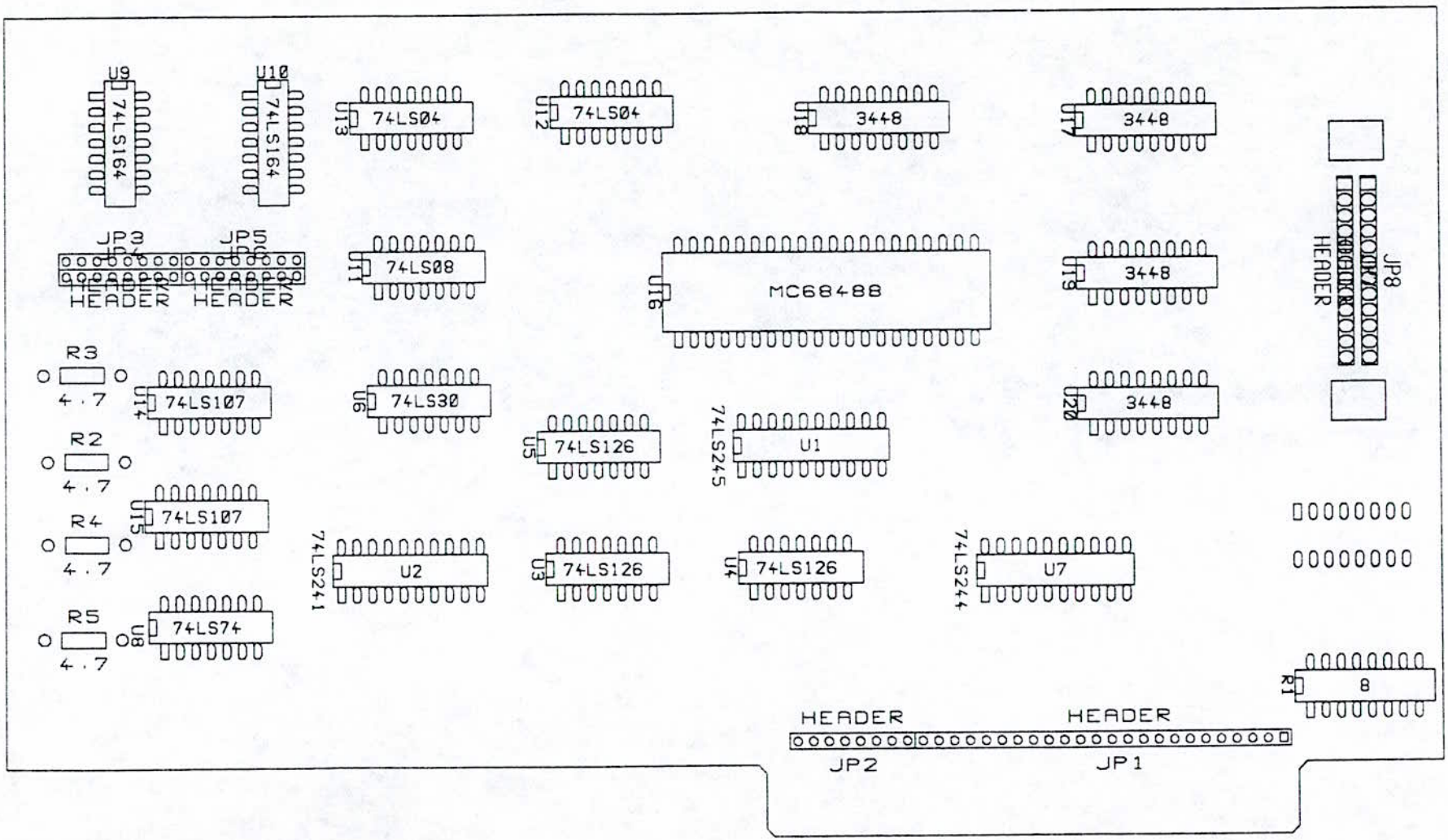


Fig. B.4: DISPOSITION DES COMPOSANTS



Item	Quantity	Reference	Part
1	1	JP1	HEADER 24
2	5	JP2	HEADER 8
		JP3	HEADER 8
		JP4	HEADER 8
		JP5	HEADER 8
		JP6	HEADER 8
3	2	JP7	HEADER 12
		JP8	HEADER 12
4	1	R1	8 * 1 K
5	4	R2	4.7 K
		R3	4.7 K
		R4	4.7 K
		R5	4.7 K
6	1	S1	SW DIP-8
7	1	U1	74LS245
8	1	U2	74LS241
9	3	U3	74LS126
		U4	74LS126
		U5	74LS126
10	1	U6	74LS30
11	1	U7	74LS244
12	1	U8	74LS74
13	2	U9	74LS164
		U10	74LS164
14	1	U11	74LS08
15	2	U12	74LS04
		U13	74LS04
16	2	U14	74LS107
		U15	74LS107
17	1	U16	MC68488
18	4	U17	3448
		U18	3448
		U19	3448
		U20	3448

ANNEXE C: Listing du driver IEEEDRV

```
;*****************************************************************
;*                          IEEEDRV                              *
;*                                                               *
;*  Fonction: Ceci constitue un driver pour la carte d'entrée et sortie *
;*            IEEE-488 conçue à base du MC 68488 de MOTOROLA.    *
;*                                                               *
;*                                                               *
;*  Auteurs:      OURAHMANE  Amor                                *
;*                OUKEMOUM   Ali                                 *
;*                                                               *
;*                                                               *
;*  Développé le:       18.06.1996                               *
;*  Dernière MAJ le:    30.07.1996                               *
;*                                                               *
;*                                                               *
;*  Assemblage avec:                                             *
;*                MASM IEEEDRV                                   *
;*                LINK  IEEEDRV                                  *
;*                EXE2BIN  IEEEDRV  IEEEDRV.SYS                  *
;*                                                               *
;*                                                               *
;*  Appel: Copier le programme IEEEDRV.SYS dans le répertoire racine *
;*         ajouter l'instruction DEVICE= IEEEDRV.SYS dans le fichier *
;*         CONFIG.SYS et relancer le système.                    *
;*****************************************************************

; ____________________________
; Positionnement des micro-interrupteurs
; ____________________________

; SWI-1: Off = 0
; SWI-2: Off = 0
; SWI-3: On  = 1
; SWI-4: Off = 0
; SWI-5: On  = 1
; SWI-6: Off = 0
; SWI-7: Off = 0
; SWI-8: Off = 0
```

```
;_____
; Adresses des registres internes du MC 68488
;_____

        ; R0: 0400H
        ; R1: 0401H
        ; R2: 0402H
        ; R3: 0403H
        ; R4: 0404H
        ; R5: 0405H
        ; R6: 0406H
        ; R7: 0407H


Code    segment
        assume      cs: code, ds: code,es:code; ss: code.
        org 0                                   ; Programme sans PSP donc
                                                ; débute à l'offset 0

;_____ Constantes _____

inst    equ2            ; Offset champ d'instructions dans le bloc de données
status  equ3            ; Offset champ d'état dans bloc de données
adr_fin equ14           , Offset adresse de fin du driver dans bloc de données
nombre  equ18           ; Offset nombre dans le bloc de données
adr_b   equ14           ; Offset adresse du buffer dans bloc de données
nmb_ins equ16           ; sont supportées les fonctions 0 à 16

;_____ Données _____

; --- En-tête du driver de périphérique ----------

dw      -1, -1              ; Lien avec le driver suivant
dw      1100000000000000b   ; Attribut du driver
dw      offset strat        ; Pointeur sur la routine de stratégie
dw      offset intr         ; Pointeur sur la routine d'interruption
db      "IEEEDRV "          ; Driver de la carte fille IEEE-488
```

```
        ; --- Table de branchement pour les différents fonctions ----------

tab_fct dw offset init          ; Fonction 0: Initialisation
        dw offset dummy         ; Fonction 1: Test de support
        dw offset dummy         ; Fonction 2: Création d'un bloc BPB
        dw offset lire          ; Fonction 3: Lecture directe
        dw offset dummy         ; Fonction 4: Lecture
        dw offset dummy         ; Fonction 5: Lecture sans retrait du buffer
        dw offset dummy         ; Fonction 6: Test de l'état d'entrée
        dw offset dummy         ; Fonction 7: Vidage du buffer d'entrée
        dw offset dummy         ; Fonction 8: Ecriture
        dw offset dummy         ; Fonction 9: Ecriture avec vérification
        dw offset dummy         ; Fonction10: Test de l'état de sortie
        dw offset dummy         ; Fonction11: Vidage du buffer de sortie
        dw offset ecrire        ; Fonction12: Ecriture directe
        dw offset no_sup        ; Fonction13: Ouverture
        dw offset no_sup        ; Fonction14: Fermeture
        dw offset dummy         ; Fonction15: Support amovible
        dw offset no_sup        ; Fonction16: Sortie jusqu'à saturation

db_ptr  dw (?), (?)             ; Adresse du bloc de données transmis.

        ;--- Routines et fonctions du driver ----------------------------------------------

Strat           proc far        ; Routine de stratégie
        mov     cs: db_ptr, bx  ; Range l'adresse du bloc de données
        mov     cs: db_ptr+2, es ; Dans la variable DB_PTR
        ret                     ; Retourne au programme appelant
Strat   endp

        ; ------------------------------------------------------------------------------------

Intr    proc  far               ; Routine d'interruption
        push ax                 ; Sauve les registres sur la pile
        push bx
        push cx
        push dx
        push di
        push si
        push bp
        push ds
        push es
        push f                  ; Range également le registre des indicateurs
        push cs                 ; Fixe le registre de segment de données
        push ds                 ; Le code coincide ici avec les données
```

```
        les di, dword ptr db_ptr        ; Adresse du bloc de données dans ES: DI
        mov bl, es: [di + inst]         ; Va chercher le code d'instruction
        cmp bl, nmb_ins                 ; Le code d'instruction est-il autorisé?
        jle bc_ok                       ; Oui ---> bc_ok
        mov ax, 8003h                   ; Code "instruction inconnue"
        jmp short intr_end              ; Retourne au programme appelant

        ; --- Code d'instruction correct  ---> exécute l'instruction ----------

bc_ok
        shl bl, 1                       ; Calcule le pointeur sur la table de branchement
        xor bh, bh                      ; Annule BH
        call [tab_fct + bx]             ; Appelle la fonction
        les di, dword ptr db_ptr        ; Adresse du bloc de données dans ES:DI
        ; --- Exécution de la fonction terminée ----------
intr_end label near
        or ax, 0100h                    ; Met à 1 le bit terminé
        mov es: [di + status], ax       ; Sauvegarde le tout dans le champ d'état

        pop f                           ; Restaure le registre des indicateurs
        pop es                          ; Restaure les autres registres
        pop ds
        pop bp
        pop si
        pop di
        pop dx
        pop cx
        pop bx
        pop ax
        ret                             ; Retourne au programme appelant

intr    endp

; ----------

dummy proc near                         ; Cette routine ne fait rien
        xor ax, ax                      ; Annule le bit occupé
        ret                             ; Retourne au programme
dummy endp

; ----------

no_sup proc near                ; Cette routine est appelée par toutes les fonctions
        mov ax, 8003h           ; qui sont en fait interdites
        ret                     ; Erreur: instruction inconnue
no_sup endp

; ----------
```

```
init_c  proc near                ; Initialisation du MC 68488
        mov  ah, 80h             ; Positionement du bit reset
        out  403h, ah            ; du MC 68488
        In   ax, 404h            ; Lecture de l'adresse du MC 68488
        out 404h, ax             ; Ecriture de l'adresse du MC 68488
        out 403h, 0              ; Suppression du reset
        out  400h, c1h           ; Autorisation des interruptions BI et B0
        ret
Init_c endp

; ------------------------------------------------------------------------------------------

init    proc  near                                  ; Routine d'initialisation
        mov word ptr es: [di+adr_fin], offset init  ; Fixe l'adresse de fin du driver
        mov es: [di+adr_fin+2], cs
        call  init_c                              ; Appelle la routine d'initialisation du GPIA
        mov  ah, 9                                  ; Affichage du message d'installation
        mov   dx, offset initm                      ; Adresse du texte
        int   21h                                   ; Appelle l'interruption du DOS
        xor   ax, ax                                ; Tout est en ordre
        jmp  intr_end                               ; Retour au programme appelant

initm   db 13, 10       "**** IEEEDRV installé, par"
        db              " OURAHMANE - OUKEMOUM"
        db              "Septembre 1996"

init    endp

; ------------------------------------------------------------------------------------------

lire    proc near
        mov cx, es: [di+nombre]                 ; Lit le nombre de caractères
        jcxz  lire_e                            ; Teste si égal à 0
        les di, es: [di+adr_b]                ; Adresse du buffer de transmission en ES: DI
        cld                                     ; Si STOSB, comptage incrémental
        xor bx, bx                            ; Mis à 0 du compteur du nombre d'octets lus

tesbi:
        in  ax,  404h                           ; Lit le registre d'état d'interruption
        and ax,  1                              ; Test du bit BI
        cmp ax,  1                              ; BI = 1 ?
        jne tesbi                               ; Non ----> TESBI
        inc  bx                                 ; Incrémentation du nombre d'octets lus
        in  al,  407h                           ; Lit la donnée présente dans R7R
        stosb                                 ; Transfert dans le buffer de la foction d'appel
        cmp  bx, cx                             ; Fin d'octets à lire atteinte ?
        jne  tesbi                              ; Non ---> TESTBI
                                              ; Lecture jusqu'à ce que tous les octets soient lus
```

```
lire_e:
        xor  ax, ax                          ; Tout est en ordre
        ret                                  ; Retour au programme appelant
lire    endp

; ------------------------------------------------------------------------------------------

ecrire  proc near
        mov  cx, es: [di+nombre]             ; Lit le nombre d'octets à écrire
        jcxz ecrire_e                        ; Test si egal à 0
        lds  si, es: [di+adr_b]              ; Adresse du buffer de transmission en  DS:SI
        xor  bx, bx                          ; Mise à 0 du compteur du nombre
                                             ;d'octets écrits
tesbo:
        in  ax, 400h                         ; Lit le registre d'état d'interruption
        and ax, 40h                          ; Test du bit BO
        cmp  ax, 40h                         ; BO = 1 ?
        jne  tesbo                           ; Non ---> TESBO
        inc  bx                              ;  Incrémente le compteur
        mov ax, ds:si                        ; Lit le caractère à écrire
        out  407h, ax                        ; Transfert à R7W
        inc  ds: si
        cmp bx,  cx                          ; Fin d'octets à écrire atteinte ?
        jne  tesbo                           ; Ecrire jusqu'au dernier caractère si non.

ecrire_e:
        xor  ax, ax                          ; Tout est en ordre
        ret                                  ; Retour au programme appelant

ecrire endp

; ------------------------------------------------------------------------------------------

code    ends
        end
```

ANNEXE D: Paramètres d'entrée et de sortie des fonctions du driver.

Paramètres d'entrée

Adresse	Contenu	Type
+02 h	Numéro de la fonction (ici 00h)	1 TYPE
+0Eh	Adresse max. atteinte par la fin du driver	1 PTR
+12 h	Adresse du caractère qui suit le signe = dans l'instruction DEVICE du fichier CONFIG.SYS.	1 PTR

Paramètres de sortie

Adresse	Contenu	Type
+03 h	Mot d'état	1 WORD
+0E h	Adresse de la prmière mémoire libre à la suite du driver	1 PTR
+17 h	Indicateur d'erreur, doit prendre une valeur différente de 0 lorsque le driver n'a pas pu être initialisé	1 PTR

PARAMETRES D'ENTREE ET DE SORTIE DE LA FONCTION 00 H

Paramètres d'entrée

Adresse	Contenu	Type
+02 h	Numéro de la fonction (ici 03h)	1 TYPE
+0E h	Adresse du buffer de transmission	PTR
+12 h	Nombre d'octets à lire	PTR

Paramètres de sortie

Adresse	Contenu	Type
+03 h	Mot d'état	1 WORD
+12 h	Nombre d'octets lus	1 WORD

PARAMETRES D'ENTREE ET DE SORTIE DE LA FONCTION 03H

Paramètres d'entrée

Adresse	Contenu	Type
+02 h	Numéro de la fonction	1 TYPE
+0E h	Adresse du buffer de transmission	1 PTR
+12 h	Nombre d'octets à écrire	WORD

Paramètres de sortie

Adresse	Contenu	Type
+03 h	Mot d'état	1 WORD
+12 h	Nombre de caractères écrits	1 WORD

PARAMETRES D'ENTREE ET DE SORTIE DE LA FONCTION 0CH

*** BIBLIOGRAPHIE ***

1) "Apprentissage et utilisation du bus IEEE-488/CEI 625", Jean-Jacques Vey, les éditions EYROLLES, 1985.

2) "Bus IEEE, Appareils programmables et micro-ordinateurs", Roland Grégoire, ETSF, 1984.

3) "Manuel des interfaces", Steve LEIBSON, Mc GRAW-HILL, 1984.

4) "Data sheet MOTOROLA", MOTOROLA.

5) "Data book TTL".

6) "Les microprocesseurs 16 bits à la loupe", Roland DUBOIS, les éditions EYROLLES, 1985

7) "Circuits numériques: Théorie et applications", Roland J. TOCCI, BORDAS, 1988.

8) "SCIEMENS PCD-Family Hardware".

9) " ORCAD: CAO Electronique", Alain RIVAT, 1990.

10) "La bible du PC", Michael TISCHER, Micro Application, 1993.

11) "Microprocesseurs 16 bits", Michel AUMIAUX, Masson, 1985.

12) " Programmation système" G. de GREBISSON, PSI, 1990.

13) "Clefs pour IBM PS/2", D. MARTIN et F. PIETTE, PSI, 1988.

14) " Electronique applications", n° 31, 1983.

15) " Electronique applications", n° 51, 1987.

16) "Elektor", Mai 1988.