EFCIS

Circuits Intégrés MOS THOMSON-EFCIS

mosmosmosmosmosmosmosmosmosmosmos

EF68000

Ancienne appellation : SFF9 · 68000

SPECIFICATIONS PROVISOIRES

HMOS

HAUTE DENSITE, CANAL N,
GRILLE Si
CHARGE A DEPLETION

MICROPROCESSEUR
16 BITS

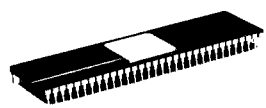
MICROPROCESSEUR 16 BITS

Les progrès en technologie semi-conducteur ont donné la possibilité d'intégrer sur une seule pastille de silicium un microprocesseur dont la puissance et la complexité du circuit ont gagné au moins un ordre de grandeur sur ce qui était disponible auparavant.
Le circuit EF 68000 est le premier d'une famille de microprocesseurs VLSI d'EFCIS. Il fait la synthèse entre le nec plus ultra de la technologie, les techniques avancées de conception de circuit et les sciences informatiques pour réaliser un microprocesseur 16 bits d'architecture avancée.

Les ressources accessibles du circuit EF 68000 sont les suivantes :

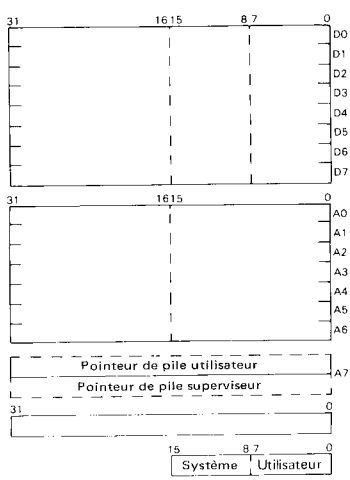
- Registres données et adresses de 32 bits
- Espace d'adressage direct de 16 Méga octets
- 56 types d'instructions puissantes
- Opérations sur cinq principaux types de données
- E/S en configuration mémoire
- 14 modes d'adressage.

Comme le montre le modèle de programmation, le circuit EF 68000 offre dix-sept registres de 32 bits qui s'ajoutent au registre d'état 16 bits et au compteur programme de 32 bits. Les huit premiers registres (D0 - D7) sont utilisés comme registres de données dans les opérations portant sur des octets (8 bits), des mots (16 bits) et des mots longs (32 bits). Les sept registres (A0-A6) et le pointeur de pile système peuvent être utilisés comme pointeurs de pile logiciels et comme registres base d'adressage. De plus, ces registres peuvent être utilisés dans les opérations d'adressage portant sur des mots, et des mots longs. Les 17 registres peuvent être utilisés comme registres index.

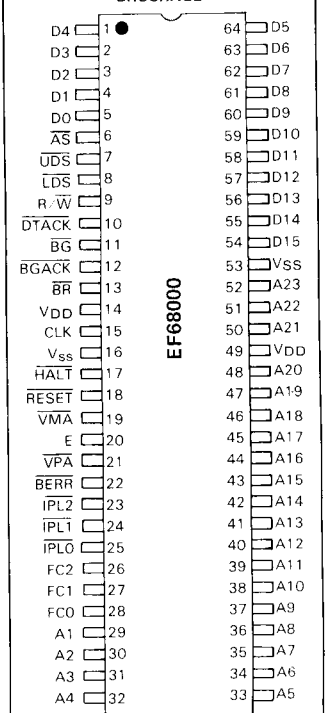


SUFFIXE C
BOITIER CERAMIQUE

MODELE DE PROGRAMMATION



BROCHAGE



ADI 814-F

EFCIS
FRANCE

45, av. de l'Europe
78140 VELIZY

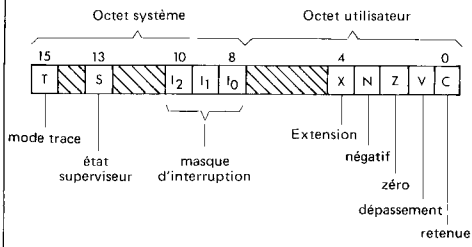
Tel.: (1) 946 97 19
Telex : 698866F

Un bus d'adresses de 23 bits fournit une étendue d'adressage mémoire supérieure à 16 mégaoctets. Ce vaste espace d'adressage, associé à une unité de gestion de mémoire, permet le développement d'importants programmes modulaires pouvant fonctionner sans avoir recours à des techniques de pagination et de comptabilité logicielle, pénalisantes en temps et gênantes.

Le registre d'état contient le masque d'interruptions (huit niveaux disponibles) et les codes condition ; extension (X), négatif (N), zéro (Z), dépassement (V), et retenue (C).

Des bits d'état supplémentaires indiquent que le processeur est en mode trace (T) et/ou en état superviseur (S).

REGISTRE D'ETAT



Il y a cinq type de données de base :

- bits
- digits BCD (4 bits)
- octets (8 bits)
- mots (16 bits)
- mots longs (32 bits)

De plus, des opérations sur d'autres types de données telles que des adresses mémoire, données de mot d'état, etc., font partie du jeu d'instructions.

Les 14 modes d'adressage (tableau 1) contiennent six types d'adressage de base :

- registre direct
- registre indirect
- absolu
- immédiat
- compteur programme relatif
- implicite

Les possibilités d'indexation, de déplacement de prédécrémentation, de post-incrémentation font partie des modes d'adressage registre indirect. Le mode relatif compteur programme peut être aussi modifié via l'indexation et le déplacement.

TABLEAU 1 – MODES D'ADRESSAGE

MODE	ADRESSE EFFECTIVE
Adressage direct registre	
Données registre direct	EA = Dn
Adresse registre direct	EA = An
Adressage données absolu	
Absolu court	EA = (Mot suivant)
Absolu long	EA = (2 mots suivants)
Adressage compteur programme relatif	
Relatif avec déplacement	EA = (PC) + d_{16}
Relatif avec index et déplacement	EA = (PC) + (Xn) + d_8
Adressage registre indirect	
Registre indirect	EA = (An)
Registre indirect post-incrémenté	EA = (An), An ← An + N
Registre indirect prédécrémenté	AN ← An - N, EA = (An)
Registre indirect avec déplacement	EA = (An) + d_{16}
Registre indirect indexé avec déplacement	EA = (An) + (Xn) + d_8
Adressage données immédiat	
Immédiat	Données = mots suivants
Immédiat rapide	Données implicites
Adressage implicite	
Registre implicite	EA = SR, USP, SP, PC

NOTES :

EA = adresse effective
An = registre adresse
Dn = registre données
Xn = registre adresse ou données utilisé comme registre index
SR = registre d'état
PC = compteur programme

( ) = contenu de
d_8 = déplacement huit bits
d_{16} = déplacement seize bits
N = 1 pour octet, 2 pour mots et 4 pour mots longs
← = transféré dans

Le jeu d'instruction du circuit EF 68000 est présenté dans le tableau 2. Quelques instructions complémentaires sont des variantes, ou sous-ensembles, du jeu de base (tableau 3). Un effort particulier a été fait pour que le jeu d'instructions puisse s'adapter à des langages structurés de haut niveau qui facilitent la programmation. Chaque instruction, à quelques exceptions près, porte sur des octets, des mots, des mots longs, et la plupart des instructions peuvent utiliser chacun des 14 modes d'adressage. En combinant les types d'instructions, les types de données et les modes d'adressage, plus de 1000 instructions utiles sont disponibles. Ces instructions comprennent la multiplication et la division signées ou non, les opérations d'arithmétique "rapides", l'arithmétique BCD et les opérations étendues (à travers les trappes).

TABLEAU 2 – JEU D'INSTRUCTIONS

Mnémo.	Description	Mnémo.	Description	Mnémo.	Description
ABCD	addition décimale étendue	EOR	OU exclusif logique	PEA	empilement de l'adresse effective
ADD	addition	EXG	échange de registres	RESET	mise à zéro des circuits externes
AND	ET logique	EXT	extension de signe	ROL	décalage circulaire gauche non étendu
ASL	décalage arithmétique gauche	JMP	saut inconditionnel	ROR	décalage circulaire droite non étendu
ASR	décalage arithmétique droite	JSR	saut à un sous-programme	ROXL	décalage circulaire gauche étendu
B_{CC}	branchement conditionnel	LEA	chargement d'adresse effective	ROXR	décalage circulaire droite étendu
BCHG	test de bit et inversion	LINK	chaînage de pile	RTE	retour d'exception
BCLR	test de bit et mise à zéro	LSL	décalage logique gauche	RTR	retour et restaure
BRA	branchement inconditionnel	LSR	décalage logique droite	RTS	retour de sous-programme
BSET	test de bit et mise à un	MOVE	transfert	SBCD	soustraction décimale étendue
BSR	branchement à un sous-programme	MOVEM	transfert de plusieurs registres	S_{CC}	mise à un conditionnelle
BTST	test de bit	MOVEP	transfert données périphériques	STOP	arrêt d'exécution du programme
CHK	test de registre, aux limites	MULS	multiplication signée	SUB	soustraction
CLR	mise à zéro de l'opérande	MULU	multiplication non signée	SWAP	échanger les moitiés de registre
CMP	comparaison	NBCD	complément à 10 avec extension	TAS	test opérande et établis. d'1 indicateur
DB_{CC}	test cond. décrémentation et branchement	NEG	complément à 2	TRAP	trappe
DIVS	division signée	NOP	pas d'opération	TRAPV	trappe sur dépassement
DIVU	division non signée	NOT	complément logique	TST	test d'une opérande
		OR	OU logique	UNLK	rupture du chaînage de pile.

TABLEAU 3 – VARIANTES DE TYPES D'INSTRUCTION

Type d'instruction	Variante	Description	Type d'instruction	Variante	Description
ADD	ADD	addition	**MOVE**	MOVE	transfert
	ADDA	addition adresse		MOVEA	transfert adresse
	ADDQ	addition rapide		MOVEQ	transfert rapide
	ADDI	addition immédiate		MOVE depuis SR	transfert du registre d'état
	ADDX	addition étendue		MOVE vers SR	transfert vers registre d'état
				MOVE vers CCR	transfert vers registre codes condition
AND	AND	ET logique		MOVE USP	transfert pointeur de pile utilisateur
	ANDI	ET immédiat	**NEG**	NEG	complémentation
				NEGX	complémentation étendue
CMP	CMP	comparaison	**OR**	OR	OU logique
	CMPA	comparaison d'adresse		ORI	OU immédiat
	CMPM	comparaison mémoire			
	CMPI	comparaison immédiate	**SUB**	SUB	soustraction
				SUBA	soustraction adresse
EOR	EOR	OU exclusif		SUBI	soustraction immédiate
	EORI	OU exclusif immédiat		SUBQ	soustraction rapide
				SUBX	soustraction étendue

VALEURS LIMITES

Paramètres	Symboles	Valeurs	Unités
Tension d'alimentation	V_{DD}	−0.3 à + 7.0	V
Gamme de température	V_{in}	−0.3 à + 7.0	V
Tension d'entrée	T_A	0 à 70	°C
Température de stockage	T_{stg}	−55 à 150	°C

CARACTERISTIQUES ELECTRIQUES (V_{DD} = 5.0 V ± 5% ; V_{SS} = 0 V ; T_A = 25°C)

Caractéristiques		Symboles	Min	Typ	Max	Unités
Tension d'entrée niveau haut		V_{IH}	—	2.0	V_{DD}	V
Tension d'entrée niveau bas		V_{IL}	V_{SS} - 0.3	0.8	—	V
Courant de fuite en entrée	$\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, $\overline{IPL0}$-$\overline{IPL2}$, VPA	I_{in}	—	1.0	—	µA
	$\overline{HALT}$, $\overline{RESET}$		—	2.0	—	
Courant d'entrée en trois états	$\overline{AS}$, A1-A23, D0-D15 FC0-FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	I_{TSI}	—	2.0	—	µA
Tension de sortie (I_{OH} = −400 µA) niveau haut	$\overline{AS}$, A1-A23, $\overline{BG}$, D0-D15, E, FC0-FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	V_{OH}	—	2.4	—	V
Tension de sortie niveau bas		V_{OL}				V
(I_{OL} = 1.6 mA)	$\overline{HALT}$		—	V_{SS} + 0.4	—	
(I_{OL} = 3.2 mA)	A1-A23, $\overline{BG}$, E, FC0-FC2		—	V_{SS} + 0.4	—	
(I_{OL} = 5.0 mA)	$\overline{RESET}$		—	V_{SS} + 0.4	—	
(I_{OL} = 5.3 mA)	$\overline{AS}$, D0-D15, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$		—	V_{SS} + 0.4	—	
Puissance dissipée (fréquence d'horloge = 8 MHz)		P_D	—	1.2	—	W
Capacité (fonction du type de boîtier) (V_{in} = 0 V ; T_A = 25°C ; fréquence = 1 MHz)		C_{in}	—	10.0		pF

FIGURE 1 – CHARGE TEST RESET

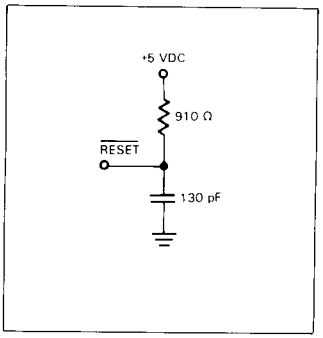


FIGURE 2 – CHARGE TEST HALT

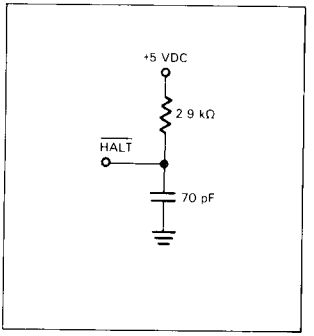
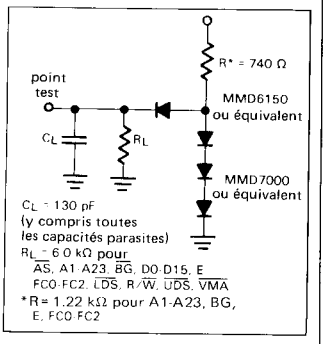


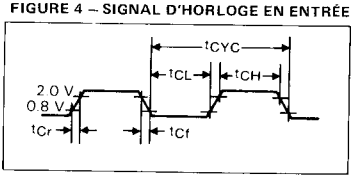
FIGURE 3 – CHARGES TEST



CARACTERISTIQUES DYNAMIQUES

Caractéristiques	Symboles	Min	Typ	Max	Unités
Fréquence nominale	F	—	—	8.0	MHz
Temps de cycle	tCYC	—	125	—	ns
Largeur d'impulsion	tCL	—	55	—	ns
	tCH	—	55	—	ns
Temps de montée et de descente	tCr	—	—	10	ns
	tCf	—	—	10	ns

FIGURE 4 – SIGNAL D'HORLOGE EN ENTRÉE



ORGANISATION DES DONNÉES ET POSSIBILITÉS D'ADRESSAGE

Les paragraphes suivants décrivent l'organisation des données et les possibilités d'adressage du circuit EF68000.

TAILLE DE L'OPERANDE

Les tailles des opérandes sont définies comme suit : 1 octet = 8 bits, 1 mot = 16 bits et 1 mot long = 32 bits. La taille de l'opérande pour chaque instruction est soit explicitement codée dans l'instruction, soit implicitement définie par le fonctionnement de l'instruction. Toutes les instructions explicites travaillent sur des opérandes dont la taille peut être l'octet, le mot ou le mot long.
Les instructions implicites travaillent avec des sous-ensembles dans les trois tailles.

ORGANISATION DES DONNEES DANS LES REGISTRES

Les huit registres de données acceptent des opérandes de 1, 8, 16 ou 32 bits. Les sept registres d'adresse et le pointeur de pile activé travaillent avec des opérandes de 32 bits.

REGISTRES DE DONNÉES. Chaque registre de données a une longueur de 32 bits. Les opérandes sous forme d'octets occupent les 8 bits de poids faible, les opérandes sous forme de mot, les 16 bits de poids faible, et les opérandes longs, la totalité des 32 bits. Le bit de poids le plus faible est adressé comme bit zéro ; le bit de poids le plus fort est adressé comme bit 31.

Lorsqu'un registre de données est utilisé soit comme opérande source, soit comme opérande destination, seule la partie appropriée de poids faible est changée ; la partie restante de poids fort n'est ni utilisée, ni modifiée.

REGISTRES D'ADRESSE. Chaque registre d'adresse et le pointeur de pile ont une longueur de 32 bits et contiennent une adresse sur 32 bits. Les registres d'adresse n'acceptent pas des opérandes dont la taille est l'octet. Par conséquent, lorsqu'un registre d'adresse est utilisé comme opérande source, soit le mot de poids faible, soit l'opérande long en sa totalité est utilisé, en fonction de la taille de l'opération. Lorsqu'un registre d'adresse est utilisé comme destination d'opérande, le registre entier est concerné indépendemment de la taille de l'opération. Si l'opération porte sur un mot, tous les autres opérandes subissent une extension de signe sur 32 bits, avant que l'opération ne soit exécutée.

ORGANISATION DES DONNÉES EN MÉMOIRE

Les octets sont adressables individuellement, l'octet de poids fort ayant une adresse paire identique à celle du mot, comme indiqué en figure 5. L'octet de poids faible a une adresse impaire supérieure de un à l'adresse du mot. Les instructions et les données sur plusieurs octets sont accédées seulement sur les limites de mot (octet pair). Si un élément d'information portant sur un mot long est situé à l'adresse n (n pair), alors le second mot de cet élément d'information est situé à l'adresse n + 2.

Le circuit EF68000 travaille sur les types de données suivants :
données binaires, nombres entiers sur 8, 16 ou 32 bits, adresse de 32 bits et données décimales codées en binaire. Chacun de ces types de données est mis en mémoire de la manière indiquée figure 6.

FIGURE 5 – ORGANISATION DES MOTS EN MEMOIRE

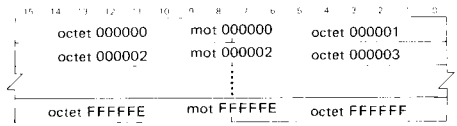
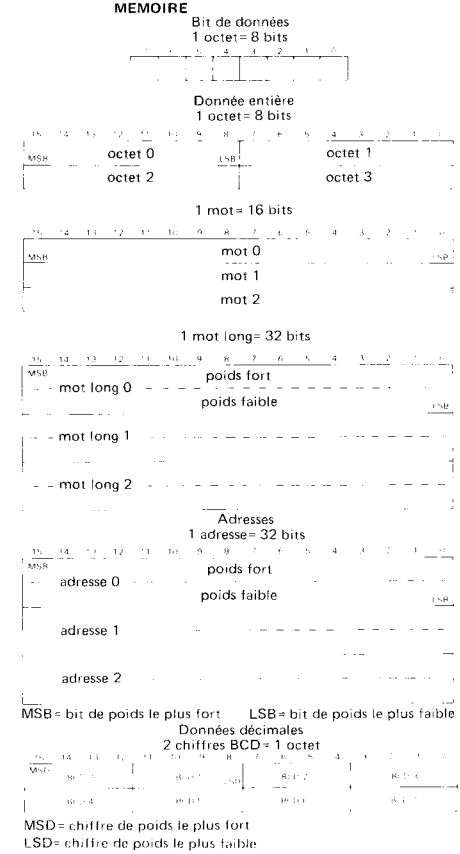


FIGURE 6 – ORGANISATION DES DONNEES EN MEMOIRE



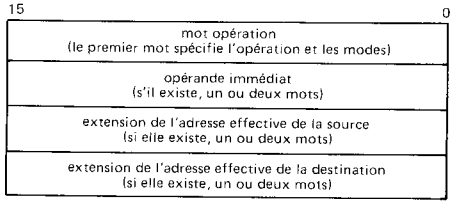
ADRESSAGE

Les instructions du circuit EF68000 contiennent deux types d'information : le type de fonction à réaliser et l'emplacement de l'opérande(s) sur lequel est réalisée la fonction. Les méthodes utilisées pour adresser le(s) opérande(s) sont expliquées dans les paragraphes suivants.
L'emplacement d'un opérande est spécifié par l'instruction de trois manières :
Spécification de registre - le numéro du registre est donné dans le champ registre de l'instruction.
Adresse effective - utilisation des différents modes d'adresse effective.
Référence implicite - la définition de certaines instructions implique l'utilisation de registres spécifiques.

FORMAT DE L'INSTRUCTION

Les instructions ont une longueur de un à cinq mots, comme indiqué figure 7. La longueur de l'instruction et l'opération à réaliser sont spécifiées par le premier mot de l'instruction qui est appelé mot opération. Les mots suivants spécifient les opérandes. Ces mots sont soit des opérandes immédiats, soit des extensions du mode d'adresse effective spécifié dans le mot opération.

FIGURE 7 – FORMAT DE L'INSTRUCTION



REFERENCES PROGRAMME / DONNÉES

Le circuit EF68000 distingue les références mémoire en deux classes : les références programme et les références données. Les références programme, comme le nom l'indique, font référence à la zone mémoire qui contient le programme à exécuter. Les références données concernent la zone mémoire qui contient les données. Habituellement, les lectures d'opérande s'effectuent dans l'espace données. Toutes les écritures d'opérande s'effectuent dans l'espace données.

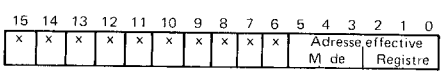
SPÉCIFICATIONS DE REGISTRE

Le champ registre dans une instruction spécifie le registre à utiliser. Les autres champs de l'instruction précisent si le registre sélectionné est un registre d'adresse ou un registre de données et comment le registre doit être utilisé.

ADRESSE EFFECTIVE

La plupart des instructions précisent l'emplacement d'un opérande en utilisant le champ d'adresse effective du mot opération. A titre d'exemple, le figure 8 montre le format général du mot opération d'une instruction d'adresse effective unique. L'adresse effective est composée de deux champs de 3 bits : le champ mode et le champ registre. La valeur du champ mode sélectionne les différents modes d'adressage. Le champ registre contient le numéro d'un registre.

FIGURE 8 – FORMAT GENERAL D'UN MOT DANS LE FONCTIONNEMENT D'UNE INSTRUCTION D'ADRESSE EFFECTIVE



Le champ adresse effective peut nécessiter des informations supplé

Le champ adresse effective peut nécessiter des informations supplémentaires pour spécifier totalement l'opérande. Cette information complémentaire, appelée extension d'adresse effective, est contenue dans le(s) mot(s) suivant(s) et est considérée comme faisant partie de l'instruction, comme indiqué figure 7. Les modes adresse effective sont groupés en trois catégories : registre direct, adressage mémoire, et spécial.

MODES D'ADRESSAGE REGISTRE DIRECT

Ces modes d'adressage effectif précisent que l'opérande est contenu dans l'un des 16 registres multifonction.
Registre données direct. L'opérande est dans le registre de de données spécifié par le champ registre de l'adresse effective.
Registre adresse direct. L'opérande est dans le registre d'adresse spécifié par le champ registre de l'adresse effective.

MODES D'ADRESSAGE MEMOIRE.

Ces modes d'adressage effectif précisent que l'opérande est en mémoire et fournissent l'adresse spécifique de l'opérande.
Adressage registre indirect. L'adresse de l'opérande se trouve dans le registre d'adresse spécifié par le champ registre. La référence est classée comme référence donnée à l'exception des instructions de branchement inconditionnel et de branchement à un sous-programme.
Adressage registre indirect avec postincrémentation. L'adresse de l'opérande se trouve dans le registre d'adresse spécifié par le champ registre. Une fois l'adresse de l'opérande utilisée, il est incrémenté de 1, 2, ou 4 en fonction de la taille de l'opérande, octet, mot ou mot long. Si le registre d'adresse est le pointeur de pile et si la taille de l'opérande est l'octet, l'adresse est augmentée de deux au lieu de un pour maintenir le pointeur de pile sur une limite de mot. La référence est classée comme référence donnée.
Adressage registre indirect avec prédécrémentation. L'adresse de l'opérande se trouve dans le registre d'adresse spécifié par le champ registre. Avant que l'adresse opérande ne soit utilisée, il est décrémenté de un, deux ou quatre en fonction de la taille de l'opérande (octet, mot ou mot long). Si le registre d'adresse est le pointeur de pile et si la taille de l'opérande est l'octet, l'adresse est décrémentée de deux et non de un pour conserver le pointeur de pile sur une limite de mot. La référence est classée comme une référence donnée.
Adressage registre indirect avec déplacement. Ce mode d'adressage requiert un mot d'extension. L'adresse de l'opérande est la somme de l'adresse contenue dans le registre d'adresse et du déplacement entier, ayant subit une extension de signe sur 16 bits, contenu dans le mot d'extension. La référence est classée comme une référence donnée à l'exception des instructions de branchement inconditionnel et de branchement à un sous-programme.
Adressage registre indirect avec index. Ce mode d'adressage requiert un mot d'extension. L'adresse de l'opérande est la somme de l'adresse contenue dans le registre d'adresse, du déplacement entier signé étendu contenu dans les huits bits de poids faible du mot d'extension, et du contenu du registre index. La référence est classée comme une référence donnée à l'exception des instructions de branchement inconditionnel et de branchement à un sous-programme.

MODES D'ADRESSAGE SPECIAUX

Les modes d'adressage spéciaux utilisent le champ registre d'adresse effective pour préciser le mode d'adressage spécial au lieu d'un numéro de registre.

Adresse courte absolue. Ce mode d'adressage requiert un mot d'extension. L'adresse de l'opérande se trouve dans le mot extension. L'adresse de 16 bits subit une extension de signe avant d'être utilisée. La référence est classée comme une référence donnée à l'exception des instructions de branchement inconditionnel et de branchement à un sous-programme.

Adresse longue absolue. Cette adresse requiert deux mots d'extension. L'adresse de l'opérande est réalisée par la concaténation des mots d'extension. La partie poids fort de l'adresse est le premier mot d'extension ; le second mot d'extension constitue la partie poids faible de l'adresse. La référence est classée comme une référence donnée à l'exception des instructions de branchement inconditionnel et de branchement à un sous-programme.

Compteur programme avec déplacement. Ce mode d'adressage requiert un mot d'extension. L'adresse de l'opérande est la somme de l'adresse contenue dans le compteur programme et du déplacement, entier signé étendu 16 bits contenu dans le mot d'extension. La valeur contenue dans le compteur programme est l'adresse du mot d'extension. La référence est classée comme une référence programme.

Compteur programme avec index. Ce mode d'adressage requiert un mot d'extension. L'adresse est la somme de l'adresse contenue dans le compteur programme, du déplacement, entier signé - étendu contenu dans les huit bits de poids faible du mot d'extension, et du contenu du registre index. La valeur dans le compteur programme est l'adresse du mot d'extension. Cette référence est classée comme une référence programme.

Donnée immédiate. Ce mode d'adressage requiert un ou deux mots d'extension en fonction de la taille de l'opération.

Opération sur octet — l'octet de poids faible du mot d'extension constitue l'opérande.
Opération sur mot — le mot d'extension constitue l'opérande.
Opération longue — l'opérande est contenu dans les deux mots d'extension, le premier mot d'extension constitue les 16 bits de poids fort, les 16 bits de poids faible étant dans le second mot d'extension.

Codes condition ou registre d'état : Certaines instructions peuvent faire référence au registre d'état grâce au champ d'adresse effective. Ces instructions sont :

ANDI avec CCR
ANDI avec SR
EORI avec CCR
EORI avec SR
ORI avec CCR
ORI avec SR

RÉSUMÉ DU CODAGE D'ADRESSE EFFECTIVE

Le tableau 4 est un résumé des modes d'adressage effectif discutés dans les paragraphes précédents.

RECHERCHE IMPLICITE

Quelques instructions font référence implicitement au compteur programme (PC), au pointeur de pile système (SP); au pointeur de pile superviseur (SSP), au pointeur de pile utilisateur (USP) ou au registre d'état (SR). Le tableau 5 fournit une liste de ces instructions et les registres concernés.

TABLEAU 4 — CODAGE D'ADRESSE EFFECTIVE RÉSUMÉ

Mode d'adressage	Mode	Registre
Registre donnée direct	000	numéro registre
Registre adresse direct	001	numéro registre
Registre adresse indirect	010	numéro registre
Registre adresse indirect avec postincrémentation	011	numéro registre
Registre adresse indirect avec prédécrémentation	100	numéro registre
Registre adresse indirect avec déplacement	101	numéro registre
Registre adresse indirect avec index	110	numéro registre
Absolu court	111	000
Absolu long	111	001
Compteur programme avec déplacement	111	010
Compteur programme avec index	111	011
Immédiat ou registre d'état	111	100

TABLEAU 5 — INSTRUCTIONS IMPLICITES

Instruction	Registres implicites
Branchement conditionnel (BCC), branchement inconditionnel (BRA)	PC
Branchement à un sous-programme (BSR)	PC, SP
Test Registre aux limites (CHK)	SSP, SR
Test condition, décrémentation et branchement (DBCC)	PC
Division signée (DIVS)	SSP, SR
Division non signée (DIVU)	SSP, SR
Branchement inconditionnel (JMP)	PC
Branchement à un sous-programme (JSR)	PC, SP
Lien et affectation (LINK)	SP
Transfert des codes condition (MOVE CCR)	SR
Transfert du registre d'état (MOVE SR)	SR
Transfert de pointeur de pile utilisateur (MOVE USP)	USP
Empiler l'adresse effective (PEA)	SP
Retour d'exception (RTE)	PC, SP, SR
Retour et restauration des codes condition (RTR)	PC, SP, SR
Retour de sous-programme (RTS)	PC, SP
Trappe (TRAP)	SSP, SR
Trappe sur dépassement (TRAPV)	SSP, SR
Non lien (UNLK)	SP

PILE SYSTEME

La pile système est utilisée implicitement par de nombreuses instructions ; les piles et files d'attente utilisateur peuvent être réalisées et gérées par les modes d'adressage. Le registre d'adresse sept (A7) constitue le pointeur de pile système (SP). Le pointeur de pile système est soit le pointeur de pile superviseur (SSP), soit le pointeur de pile utilisateur (USP), en fonction de l'état du bit S du registre d'état. Si le bit S indique un état superviseur, le SSP devient le pointeur de pile système, et USP ne peut pas être référencé comme un registre d'adresse. Si le bit S indique l'état utilisateur, USP devient le pointeur de pile système, et SSP ne peut pas être référencé. Chaque pile système se remplit de haut en bas de la mémoire.

RESUME DU JEU D'INSTRUCTIONS

Les paragraphes suivants contiennent une vue d'ensemble de la forme et de la structure du jeu d'instruction. Les instructions forment un jeu d'outils qui incluent toutes les fonctions machine pour réaliser les opérations suivantes :

Mouvement de données
Arithmétique sur les entiers
Décalages et décalages circulaires
Manipulation de bits
Binaire codé décimal
Contrôle du programme
Contrôle du système.

Les possibilités de la gamme complète d'instructions combinées à la souplesse des modes d'adressage décrits précédemment, fournissent une base très souple pour le développement de programmes.

OPERATIONS DE MOUVEMENT DE DONNÉES

La méthode de base d'acquisition de données (transfert et rangement) est réalisée par l'instruction MOVE. L'instruction MOVE et les modes d'adressage effectif permettent à la fois la manipulation d'adresse et de données. Les instructions de mouvement de données permettent de transférer des opérandes dont la taille est l'octet, le mot ou le long mot de mémoire à mémoire, de mémoire à registre, de registre à mémoire et de registre à registre. Les instructions de mouvement d'adresse permettent le transfert d'opérandes sous forme de mots et de mots longs et assurent que seules les manipulations d'adresse autorisées sont exécutées. Quelques instructions spéciales de mouvement de données viennent s'ajouter à l'instruction générale de mouvement : transfert de registres multiples (MOVEM), transfert de données avec les périphériques (MOVEP), échange entre registres (EXG), chargement d'adresse effective (LEA), empilement d'adresse effective (PEA), liaison pile (LINK), rupture liaison pile (UNLK), et transfert rapide (MOVEQ).

Le tableau 6 donne un résumé des opérations de mouvement de données.

TABLEAU 6 – OPERATIONS DE MOUVEMENT DE DONNÉES

Instruction	Taille opérande	Fonctionnement
EXG	32	Rx ◄► Ry
LEA	32	EA ►An
LINK	—	An ►SP@- SP ►An SP + d ►SP
MOVE	8, 16, 32	(EA)s ►EAd
MOVEM	16, 32	(EA) ►An, Dn An, Dn ►EA
MOVEP	16, 32	(EA) ►Dn Dn ►EA
MOVEQ	8	#xxx ►Dn
PEA	32	EA ►SP@-
SWAP	32	Dn [31:16] ◄► Dn [15:0]
UNLK	- -	An ►Sp SP @+ ►An

Notes
s = source
d = destination
[ ] = numéro de bit
@- = indirect avec prédécrémentation
@+ = indirect avec postincrémentation

OPERATIONS ARITHMETIQUES SUR LES ENTIERS

Les opérations arithmétiques comprennent les quatre opérations de base : addition (ADD), soustraction (SUB), multiplication (MUL), division (DIV), ainsi que les opérations de comparaison arithmétique (CMP), remise à zéro (CLR), et complémentation à 2 (NEG). Les instructions d'addition et de soustraction sont disponibles à la fois pour les opérations sur les adresses ou sur les données ; les opérations sur données acceptant toutes tailles d'opérandes. Les opérations sur adresse sont limitées aux opérandes de tailles d'adresse autorisées (16 ou 32 bits). Les opérations de comparaison de données, d'adresse et de mémoire sont aussi disponibles. Les instructions de remise à zéro et de complémentation à 2 peuvent utiliser toutes tailles d'opérandes.

Les opérations de multiplication et de division sont accessibles aux opérandes signés ou non signés, la multiplication portant sur un mot donnant un résultat tenant sur un mot long, un dividende sur mot long avec diviseur sur un mot donnant un quotient sur mot et un reste sur mot. On peut réaliser des opérations arithmétiques en multiprécision et avec des opérandes de tailles différentes en utilisant les instructions suivantes : addition étendue (ADDX), soustraction étendue (SUBX), extension de signe (EXT), et complémentation à 2 étendue (NEGX).

TABLEAU 7 – OPERATIONS ARITHMÉTIQUES SUR LES ENTIERS

Instruction	Taille opérande	Fonctionnement
ADD	8, 16, 32 16, 32	Dn + (EA) ►Dn (EA) + Dn ►EA (EA) + #xxx ►EA An + (EA) ►An
ADDX	8, 16, 32 16, 32	Dx + Dy + X ►Dx Ax@- Ay@- + X ►Ax@
CLR	8, 16, 32	0 ►EA
CMP	8, 16, 32 16, 32	Dn - (EA) (EA) - #xxx Ax@+ - Ay@+ An - (EA)
DIVS	32 ÷ 16	Dn/(EA) ►Dn
DIVU	32 ÷ 16	Dn/(EA) ►Dn
EXT	8 ►16 16 ►32	(Dn)8 ►Dn16 (Dn)16 ►Dn32
MULS	16 * 16 ►32	Dn * (EA) ►Dn
MULU	16 * 16 ►32	Dn * (EA) ►Dn
NEG	8, 16, 32	0 - (EA) ►EA
NEGX	8, 16, 32	0 - (EA) - X - EA
SUB	8, 16, 32 16, 32	Dn - (EA) ►Dn (EA) - Dn ►EA (EA) - #xxx ►EA An - (EA) ►An
SUBX	8, 16, 32	Dx - Dy - X ►Dx Ax@- - Ay@- - X ►Ax@
TAS	8	(EA) - 0, 1 ►EA[7]
TST	8, 16, 32	(EA) - 0

Note.
[ ] : numéro de bit

Il existe une instruction de test d'opérande (TST) qui positionne les codes condition à la suite d'une comparaison de l'opérande avec zéro. L'instruction de test opérande et établissement d'un indicateur (TAS) est une instruction de synchronisation très utile dans les systèmes multiprocesseurs. Le tableau 7 est un résumé des opérations arithmétiques sur les entiers.

OPERATIONS LOGIQUES

Les instructions sur les opérations logiques ET, OU, OU exclusif et NON sont disponibles pour toutes les tailles d'opérandes entiers. Un jeu similaire d'instructions immédiates (ANDI, ORI, et EORI) permet d'utiliser ces opérations logiques avec des données immédiates de toutes les tailles. Le tableau 8 est un résumé des opérations logiques.

TABLEAU 8 – INSTRUCTIONS LOGIQUES

Instruction	Taille opérande	Fonctionnement
AND	8, 16, 32	Dn ∧ (EA) ⟵ Dn (EA) ∧ Dn ⟵ EA (EA) ∧ #xxx ⟵ EA
OR	8, 16, 32	Dn ∨ (EA) ⟵ Dn (EA) ∨ Dn ⟵ EA (EA) ∨ #xxx ⟵ EA
EOR	8, 16, 32	(EA) ⊕ Dy ⟵ EA (EA) ⊕ #xxx ⟵ EA
NOT	8, 16, 32	~ (EA) ⟵ EA

Remarque : ~ = complément à 1.

OPERATIONS DE DECALAGE LOGIQUE, ARITHMETIQUE ET CIRCULAIRE

Les opérations de décalage dans les deux directions sont réalisées par les instructions arithmétiques ASR et ASL et les instructions de décalage logique LSR et LSL. Les instructions de décalage circulaire (étendu ou non) disponibles sont ROXR, ROXL, ROR, et ROL. Toutes les opérations de décalage et décalage circulaire peuvent être réalisées soit dans les registres, soit en mémoire. Les décalages sur registre acceptent toutes les tailles d'opérande et permettent un compte du décalage de un à huit bits spécifié dans l'instruction, ou de 0 à 63 spécifié dans un registre de données.

Décalages et décalages circulaires mémoire s'appliquent seulement à des opérandes d'un mot de longueur, et permettent des décalages de 1 bit seulement. Le tableau 9 résume les opérations de décalage.

OPERATIONS DE MANIPULATION DE BITS

Les opérations de manipulation de bits sont réalisées en utilisant les fonctions suivantes : test de bit (BTST), test de bit et mise à un d'indicateur (BSET), test de bit et mise à zéro (BCLR), et test de bit et inversion (BCHG). Le tableau 10 est un résumé des opérations de manipulation de bits.

OPERATIONS EN DECIMAL CODE BINAIRE

Des opérations arithmétiques multiprécision sur des nombres BCD sont réalisées en utilisant les instructions suivantes : addition décimale étendue (ABCD), soustraction décimale étendue (SBCD), et le complément à 10 étendu (NBCD). Le tableau 11 résume les opérations en décimal codé binaire.

TABLEAU 9 – INSTRUCTIONS DE DECALAGE ET ROTATION

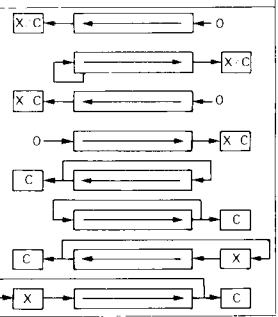
Instruction	Taille opérande	Fonctionnement
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

TABLEAU 10 – INSTRUCTIONS DE MANIPULATION DE BIT

Instruction	Taille opérande	Fonctionnement
BTST	8, 32	~ bit de (EA) ⟵ Z
BSET	8, 32	~ bit de(EA) ⟵ Z 1 ⟵ bit de EA
BCLR	8, 32	~ bit de (EA) ⟵ Z 0 ⟵ bit de EA
BCHG	8, 32	~ bit de (EA) ⟵ Z ~ bit de (EA) ⟵ bit de EA

TABLEAU 11 – INSTRUCTIONS EN BCD

Instruction	Taille opérande	Fonctionnement
ABCD	8	$Dx_{10} + Dy_{10} + X \longleftarrow Dx$ $Ax@_{-10} + Ay@_{-10} + X \longleftarrow Ax@$
SBCD	8	$Dx_{10} - Dy_{10} - X \longleftarrow Dx$ $Ax@_{-10} - Ay@_{-10} - X \longleftarrow Ax@$
NBCD	8	$0 - (EA)_{10} - X \longleftarrow EA$

OPERATIONS DE CONTROLE DU PROGRAMME

Les opérations de contrôle du programme sont réalisées en utilisant une série d'instruction de branchement conditionnel et inconditionnel et d'instructions de retour. Ces instructions sont résumées dans le tableau 12.

Les instructions conditionnelles réalisent le positionnement des indicateurs et le branchement pour les conditions suivantes :

CC – pas de retenue
CS – retenue
EQ – égal
F – jamais vrai
GE – supérieur ou égal
GT – supérieur à
HI – haut
LE – inférieur ou égal
LS – inférieur ou égal
LT – inférieur
MI – moins
NE – non égal
PL – plus
T – toujours vrai
VC – pas de dépassement
VS – dépassement.

TABLEAU 12 – INSTRUCTIONS DE CONTROLE DE PROGRAMME

Instructions	Fonctionnement
Conditionnelles	
B_{CC}	Branchement conditionnel (14 conditions) déplacement 8 et 16 bits
DB_{CC}	Test condition, décrémentation et branchement, déplacement 16 bits
S_{CC}	Etablir octet conditionnellement (16 conditions)
Inconditionnelles	
BRA	Branchement inconditionnel déplacement 8 et 16 bits
BSR	Branchement à un sous-programme déplacement 8 et 16 bits
JMP	Saut inconditionnel
JSR	Saut à un sous-programme
de retour	
RTR	Retour et restauration des codes condition
RTS	Retour de sous-programme

TABLEAU 13 – INSTRUCTIONS DE CONTROLE DU SYSTEME

Instructions	Fonctionnement
Privilégiées	
RESET	Mise à zéro des circuits extérieurs
RTE	Retour d'exception
STOP	Arrêt d'exécution du programme
ORI to SR	OU logique avec registre d'état
MOVE USP	Transfert pointeur de pile utilisateur
ANDI to SR	ET logique avec registre d'état
EORI to SR	OU exclusif logique avec registre d'état
MOVE EA to SR	Chargement nouveau registre d'état
Génératrices de trappes	
TRAP	Trappe
TRAPV	Trappe sur dépassement
CHK	Test registre aux limites
Registre d'état	
ANDI to CCR	ET logique avec codes condition
EORI to CCR	OU exclusif avec codes condition
MOVE EA to CCR	Chargement nouveaux codes condition
ORI to CCR	OU logique avec codes condition
MOVE SR to EA	Rangement du registre d'état.

OPÉRATIONS DE CONTROLE DU SYSTEME

Les opérations de contrôle du système sont accomplies en utilisant les instructions privilégiées, les instructions générant des trappes et les instructions qui utilisent ou modifient le registre d'état. Ces instructions sont résumées dans le tableau 13.

DESCRIPTION DES SIGNAUX ET FONCTIONNEMENT DU BUS

Les paragraphes suivants contiennent une brève description des signaux d'entrée et de sortie. Une discussion est aussi donnée sur le fonctionnement du bus pendant les divers cycles machine et opérations.

DESCRIPTION DES SIGNAUX

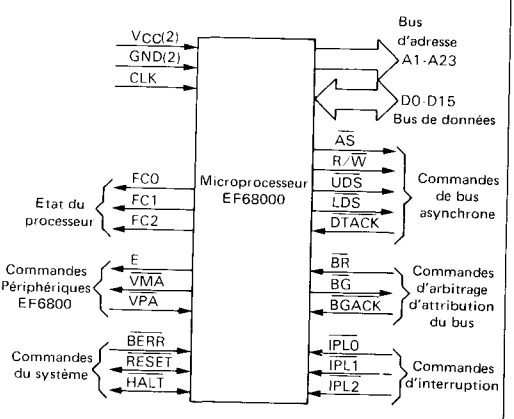
Les signaux d'entrée et sortie peuvent-être organisés fonctionnellement en groupes montrés figure 9. Les paragraphes suivants donnent une brève description des signaux ainsi qu'une référence (si possible) aux autres chapitres qui contiennent plus de détails sur la fonction réalisée.

BUS D'ADRESSE (A1 à A23). Ce bus de 23 bits unidirectionnel, à trois états, est capable d'adresser 8 méga mots de données. Il fournit l'adresse pour le fonctionnement du bus durant tous les cycles à l'exception des cycles d'interruption. Lors des cycles d'interruption, les lignes d'adresse A1, A2 et A3 donnent des informations sur le niveau d'interruption qui est servi tandis que les lignes d'adresse A4 à A23 sont toutes placées au niveau logique haut.

BUS DE DONNÉES (D0 à D15). Ce bus de 16 bits, bidirectionnel, à trois états, est le chemin des données universel. Il peut transférer et accepter des données soit sous forme de mot, soit sous forme d'octet. Au cours d'un cycle de reconnaissance d'interruption, les circuits externes fournissent un numéro de vecteur sur les lignes de données D0-D7.

COMMANDES DE BUS ASYNCHRONE. Les transferts de données asynchrones sont réalisés en utilisant les signaux de commande suivants : échantillonnage d'adresse, lecture/écriture, échantillonnages données haut et bas et reconnaissance de transfert de données. Ces signaux sont expliqués dans les paragraphes suivants.

FIGURE 9 – SIGNAUX ENTRÉES/SORTIES



Echantillonnage d'adresse ($\overline{AS}$). Ce signal indique qu'il y a une adresse valide sur le bus d'adresse.

Lecture/Ecriture (R/$\overline{W}$). Ce signal définit le transfert bus de données comme un cycle lecture ou un cycle écriture. Le signal R/$\overline{W}$ fonctionne aussi en relation avec les échantillonnages données haut et bas comme expliqué dans le paragraphe suivant.

Echantillonnage données haut et bas ($\overline{UDS}$, $\overline{LDS}$). Ces signaux contrôlent les données sur le bus de données (tableau 14). Lorsque la ligne R/$\overline{W}$ est à l'état bas, le processeur écrit sur le bus de données.

TABLEAU 14 – CONTROLE ECHANTILLONNAGE DONNÉES DE BUS

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8-D15	D0-D7
haut	Haut	–	données non valides	données non valides
bas	bas	haut	bits données valides 8-15	bits données valides 0-7
haut	bas	haut	données non valides 8-15	bits données valides 0-7
bas	haut	haut	bits données valides 8-15	données non valides 0-7
bas	bas	bas	bits données valides 8-15	bits données valides 0-7
haut	bas	bas	bits données valides 0-7*	bits données valides 0-7
bas	haut	bas	bits données valides 8-15	bits données valides 8-15*

* Ces conditions résultent de l'implémentation en cours et peuvent ne pas apparaître sur les circuits futurs.

Reconnaissance de transfert de données ($\overline{DTACK}$). Cette entrée indique que le transfert de données est réalisé. Lorsque le processeur reconnaît $\overline{DTACK}$ pendant un cycle lecture, les données sont mémorisées/verrouillées et le cycle bus est terminé.
Lorsque $\overline{DTACK}$ est reconnu pendant un cycle d'écriture, le cycle bus est terminé.

COMMANDES D'ARBITRAGE D'ATTRIBUTION DU BUS. Ces trois signaux réalisent un circuit d'arbitrage d'attribution du bus pour déterminer le circuit qui sera maître du bus.

Requête de bus ($\overline{BR}$). Cette entrée est câblée en OU avec tous les autres circuits qui peuvent être maître du bus. Cette entrée indique au processeur qu'un autre circuit désire être maître du bus.

Bus accordé ($\overline{BG}$). Cette sortie indique à tous les autres circuits pouvant prendre le contrôle du bus que le processeur va libérer le contrôle du bus à la fin du cycle bus en cours.

Reconnaissance d'allocation de bus ($\overline{BGACK}$). Cette entrée informe qu'un autre circuit a pris le contrôle du bus. Ce signal ne peut pas être activé tant que les quatre conditions suivantes ne sont pas réalisées :

1. Un signal bus accordé a été reçu
2. L'échantillonnage d'adresse est inactif, ce qui indique que le micro-processeur n'utilise pas le bus.
3. Le signal reconnaissance de transfert de données est inactif, indiquant que mémoire et périphériques n'utilisent pas le bus.
4. Le signal de reconnaissance d'allocation de bus est inactif ce qui indique qu'aucun autre circuit ne réclame le contrôle du bus.

COMMANDES D'INTERRUPTION ($\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$) Ces broches d'entrée indiquent le niveau de priorité du circuit demandant une interruption. Le niveau sept a la plus haute priorité tandis que le niveau zéro indique qu'aucune interruption n'est demandée. $\overline{IPL0}$ est le bit de poids le plus faible et $\overline{IPL2}$ est le bit de poids le plus fort.

COMMANDES SYSTEME. Les entrées de commande système sont utilisées soit pour initialiser, soit pour arrêter le processeur et lui indiquer que des erreurs bus sont survenues. Les trois entrées de commande du système sont expliquées dans les paragraphes suivants :

Erreur bus ($\overline{BERR}$). Cette entrée informe le processeur qu'il y a un problème avec le cycle en cours d'exécution. Ces problèmes peuvent résulter de :

1. Circuits ne répondant pas.
2. Défaillance dans l'acquisition du numéro de vecteur d'interruption.
3. Requête d'accès interdit comme déterminé par une unité de gestion mémoire.
4. Autres erreurs fonction de l'application.

Le signal d'erreur bus interagit avec le signal d'arrêt pour déterminer si le traitement d'exception doit être réalisé ou si le cycle bus en cours doit être renouvelé.

Se reporter au paragraphe **ERREUR BUS ET ARRET** pour avoir des informations complémentaires sur l'interaction des signaux erreur bus et arrêt.

Initialisation ($\overline{RESET}$). Cette ligne bidirectionnelle initialise (démarre une séquence d'initialisation du système) le processeur en réponse à un signal de remise à zéro externe. Une initialisation générée de façon interne (résultat d'une instruction RESET) entraîne la mise à zéro de tous les circuits externes et l'état interne du processeur n'est pas affecté. Une mise à zéro globale du système (processeur et circuits extérieurs) a lieu lorsque les signaux externes d'initialisation et d'arrêt sont appliqués en même temps. Se reporter au paragraphe **OPERATION D'INITIALISATION** pour des informations complémentaires concernant l'initialisation.

Arrêt ($\overline{HALT}$). Lorsque cette ligne bidirectionnelle est pilotée par un circuit extérieur, l'arrêt du processeur est réalisé à la fin du cycle bus en cours. Lorsque le processeur a été arrêté en utilisant cette entrée, tous les signaux de commande sont inactifs et toutes les lignes trois états passent dans l'état haute impédance. Se reporter au chapitre **ERREUR BUS ET ARRET** pour des informations complémentaires sur l'interaction des signaux erreur bus et arrêt.
Lorsque le processeur a cessé l'exécution d'instructions, comme après une double erreur sur le bus, la ligne arrêt est pilotée par le processeur pour indiquer aux circuits externes que le processeur est à l'arrêt.

COMMANDES PERIPHERIQUES EF6800. Ces signaux de commande sont utilisés pour permettre l'interfaçage de circuits périphériques EF6800 synchrones, avec le circuit asynchrone EF68000. Ces signaux sont expliqués dans les paragraphes suivants.

Validation (E). Ce signal est le signal standard de la validation commun à tous les circuits périphériques EF6800. La période pour cette sortie est de dix périodes d'horloge EF68000 (six signaux d'horloge bas ; quatre signaux d'horloge haut).

Validation adresse périphérique ($\overline{VPA}$). Cette entrée indique que le circuit ou la région adressé est un circuit de la famille EF6800 et que le transfert des données doit coïncider avec le signal de validation (E). Cette entrée indique aussi que le processeur doit utiliser le système de vectorisation automatique pour les interruptions. Se reporter au chapitre **INTERFACE AVEC LES PERIPHERIQUES EF6800.**

Validation adresse mémoire ($\overline{VMA}$). Cette sortie est utilisée pour indiquer aux circuits périphériques du EF6800 que l'adresse présente sur le bus d'adresse est valide et que le processeur est synchronisé avec le signal de validation (E). Ce signal ne répond qu'à une entrée validation adresse périphérique ($\overline{VPA}$) qui indique que le périphérique est un circuit de la famille EF68)).

ETAT DU PROCESSEUR (FC0, FC1, FC2). Ces sorties code fonction, indiquent le mode (mode utilisateur ou superviseur) et le type de cycle en cours d'exécution (Tableau 15).
Les informations indiquées par les sorties code fonction sont utilisables chaque fois que le signal d'échantillonnage d'adresse ($\overline{AS}$) est actif.

TABLEAU 15 – SORTIES CODE FONCTION

FC2	FC1	FC0	Type de cycle
bas	bas	bas	(Indéfini, réservé)
bas	bas	haut	Données utilisateur
bas	haut	bas	Programme utilisateur
bas	haut	haut	(Indéfini, réservé)
haut	bas	bas	(Indéfini, réservé)
haut	bas	haut	Données superviseur
haut	haut	bas	Programme superviseur
haut	haut	haut	Reconnaissance d'interruption

HORLOGE (CLK). L'entrée horloge est un signal compatible TTL traité de façon interne pour générer des signaux d'horloge nécessaires au processeur. L'entrée horloge doit avoir une fréquence constante.

RESUME DES SIGNAUX. Le tableau 16 est un résumé de tous les signaux présentés dans les paragraphes précédents.

TABLEAU 16 – RESUME DES SIGNAUX

Nom du signal	Mnémonique	Entrée/Sortie	Niveau actif	Trois états
Bus d'adresse	A1-A23	sortie	haut	oui
Bus de données	D0-D15	entrée/sortie	haut	oui
Echantillonnage adresse	$\overline{AS}$	sortie	bas	oui
Lecture/écriture	R/$\overline{W}$	sortie	lecture/haut écriture/bas	oui
Echantillonnage données haut et bas	$\overline{UDS}$, $\overline{LDS}$	sortie	bas	oui
Reconnaissance de transfert de données	$\overline{DTACK}$	entrée	bas	non
Requête du bus	$\overline{BR}$	entrée	bas	non
Bus accordé	$\overline{BG}$	sortie	bas	non
Reconnaissance d'allocation de bus	$\overline{BGACK}$	entrée	bas	non
Niveau de priorité d'interruption	$\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$	entrée	bas	non
Erreur bus	$\overline{BERR}$	entrée	bas	non
Initialisation	$\overline{RESET}$	entrée/sortie	bas	non *
Arrêt	$\overline{HALT}$	entrée/sortie	bas	non *
Validation (horloge)	E	sortie	haut	non
Validation adresse mémoire	$\overline{VMA}$	sortie	bas	oui
Validation adresse périphérique	$\overline{VPA}$	entrée	bas	non
Sortie code fonction	FC0, FC1, FC2	sortie	haut	oui
Horloge	CLK	entrée	haut	non
Entrée puissance	V_{CC}	entrée	–	–
Masse	GND	entrée	–	–

* drain ouvert.

FONCTIONNEMENT DU BUS

Les paragraphes suivants traitent des signaux de commande et du fonctionnement du bus pendant les opérations de transfert de données, d'arbitrage d'allocation de bus, d'erreur bus et des conditions d'arrêt, et d'initialisation.

OPERATIONS DE TRANSFERT DE DONNÉES. Le transfert de données entre circuits utilise les lignes suivantes :

- Bus d'adresses A1 à A23
- Bus de données D0 à D15
- Signaux de contrôle

Les bus de données et d'adresses sont des bus parallèles séparés, utilisés pour transférer des données dans une structure de bus asynchrone. Dans tous les cycles, le maître du bus assume la responsabilité de resynchronisation de tous les signaux qu'il délivre au début et à la fin du cycle. De plus, le circuit maître du bus est responsable de la resynchronisation des signaux de reconnaissance et de données venant du circuit esclave.

Les paragraphes suivants expliquent les cycles de lecture, écriture, et lecture-modification-écriture. Le cycle indivisible de lecture/modification/écriture est la méthode utilisée par le circuit EF68000 pour interverrouiller des communications multiprocesseurs.

NOTE

Les termes **affirmation et infirmation** seront généralement utilisés. Ceci doit éviter la confusion à traiter d'un mélange de signaux "actif à l'état bas" et "actif à l'état haut". Le terme affirmer ou affirmation est utilisé pour indiquer qu'un signal est actif ou vrai indépendamment du fait que le niveau de tension soit bas ou haut. Le terme infirmer ou infirmation est utilisé pour indiquer qu'un signal est inactif ou faux.

Cycle lecture. Pendant un cycle de lecture, le processeur reçoit des données de la mémoire ou d'un circuit périphérique. Le processeur lit des octets de données dans tous les cas. Si l'instruction précise que l'opération porte sur un mot (ou un double mot), le processeur lit les deux octets. Lorsque l'instruction précise une opération sur octet, le processeur utilise un bit interne A0 pour déterminer quel octet lire, puis délivre le signal d'échantillonnage de données requis pour cet octet. Pour des opérations portant sur un octet, lorsque le bit A0 égal zéro, l'échantillonnage de données d'octet supérieur est délivré. Lorsque le bit A0 égal 1, l'échantillonnage de données d'octet inférieur est délivré. Lorsque les données sont reçues, le processeur les place correctement à l'intérieur.

Un organigramme du cycle de lecture d'un mot est donné en figure 10. Un organigramme du cycle de lecture d'un octet est donné en figure 11. Un chronogramme du cycle de lecture est donné en figure 12, et la figure 13 détaille le fonctionnement du cycle lecture d'un octet et d'un mot.

FIGURE 10 – ORGANIGRAMME DU CYCLE DE LECTURE D'UN MOT

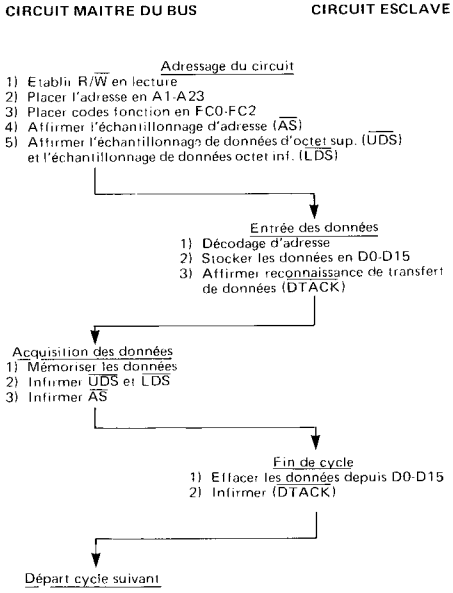


FIGURE 11 – ORGANIGRAMME DU CYCLE DE LECTURE D'UN OCTET

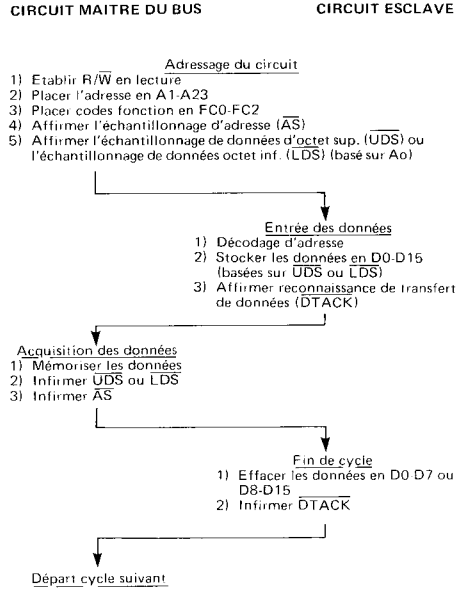


FIGURE 12 – DIAGRAMME DES TEMPS DES CYCLES DE LECTURE ET D'ECRITURE

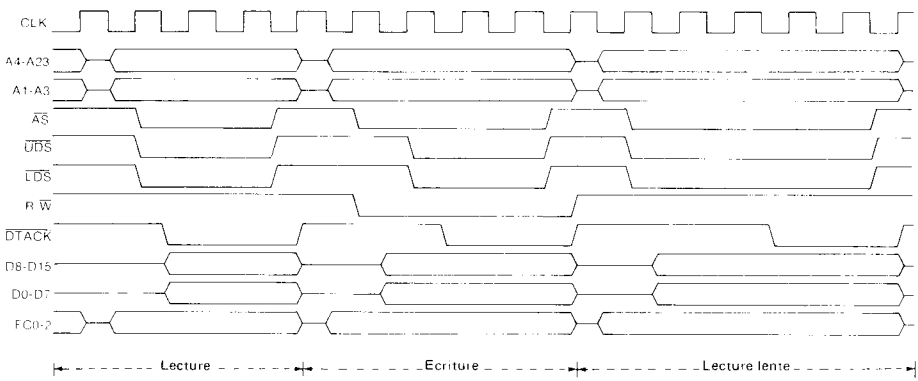
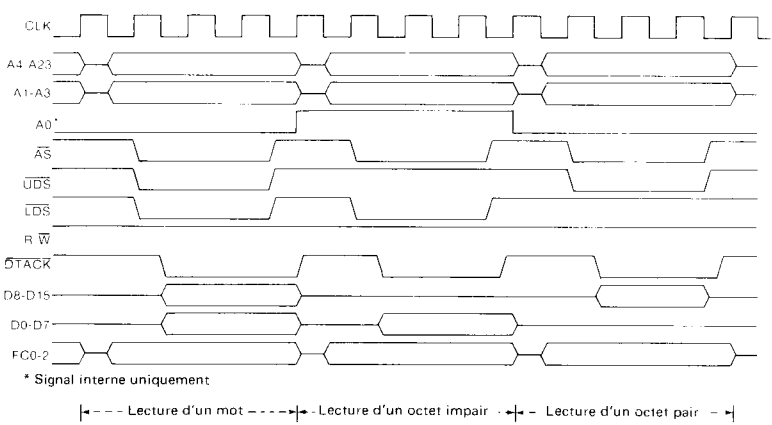


FIGURE 13 – DIAGRAMME DES TEMPS DES CYCLES DE LECTURE D'UN MOT ET D'UN OCTET



Cycle écriture. Lors d'un cycle écriture ; le processeur envoie les données en mémoire ou vers un circuit périphérique. Le processeur écrit des octets de données dans tous les cas. Si l'instruction spécifie une opération sur mot, le processeur écrit les deux octets. Lorsque l'instruction spécifie une opération sur octet, le processeur utilise un bit interne A0 pour déterminer quel octet écrire, puis délivre l'échantillonnage de données requis pour cet octet. Pour les opérations sur octet, lorsque le bit A0 est égal à zéro, l'échantillonnage de données d'octet supérieur est délivré. Lorsque le bit A0 est égal à un, l'échantillonnage de données d'octet inférieur est délivré. Un organigramme de cycle d'écriture de mot est donné en figure 14. Un organigramme du cycle d'écriture d'octet est donné en figure 15. Le diagramme des temps d'un cycle d'écriture est donné en figure 12 et la figure 16 détaille le cycle d'écriture d'un mot et d'un octet.

FIGURE 14 – ORGANIGRAMME DU CYCLE D'ECRITURE D'UN MOT

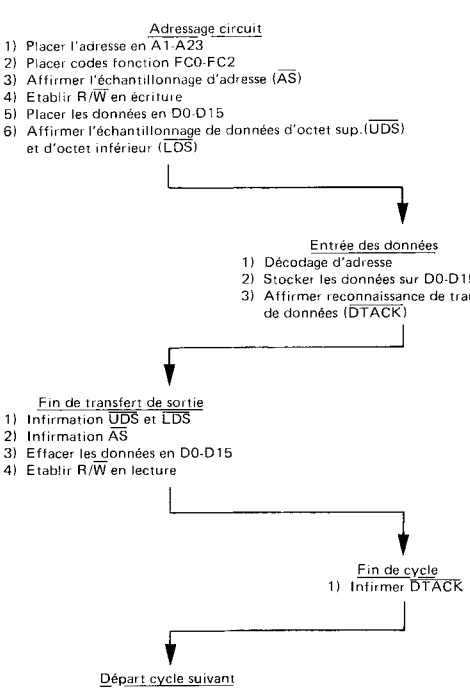


FIGURE 15 – ORGANIGRAMME DU DYCLE D'ECRITURE D'UN OCTET

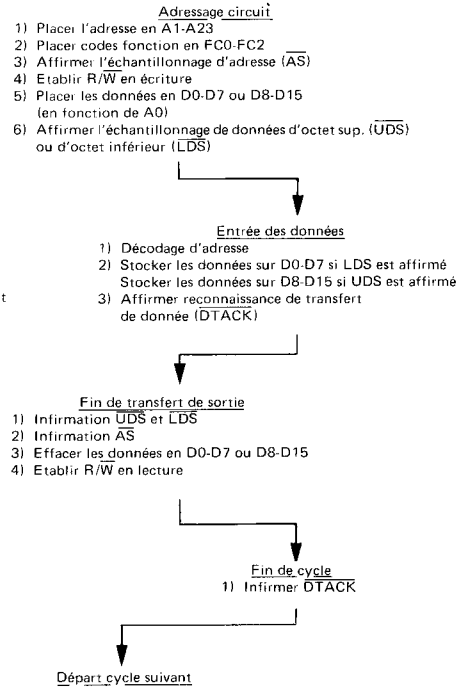
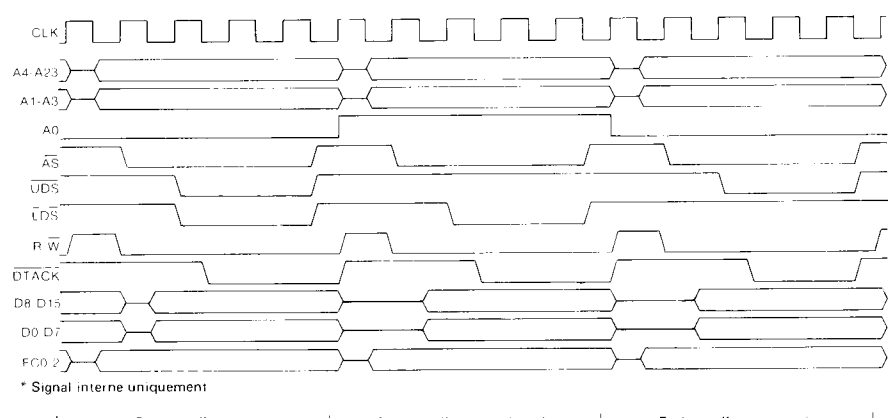


FIGURE 16 – DIAGRAMME DES TEMPS D'UN CYCLE D'ECRITURE D'UN MOT ET D'UN OCTET



Cycle lecture - modification - écriture. Le cycle lecture-modification-écriture réalise une lecture, modifie les données dans l'unité arithmétique et logique, et écrit les données à la même adresse. Dans le circuit EF68000 ce cycle est indivisible car l'échantillonnage adresse est affirmé tout au long du cycle entier. L'instruction test opérande et établissement d'un indicateur (TAS) utilise ce cycle pour fournir une communication significative entre processeurs dans un environnement multiprocesseurs. Cette instruction est la seule instruction qui utilise le cycle lecture - modification - écriture et compte tenu du fait que l'instruction TAS ne porte que sur des octets, tous les cycles lecture - modification - écriture sont des opérations sur octet. La figure 17 donne l'organigramme d'un cycle lecture - modification - écriture et la figure 18 en donne le diagramme des temps.

FIGURE 17 – ORGANIGRAMME DU CYCLE DE LECTURE-MODIFICATION-ECRITURE

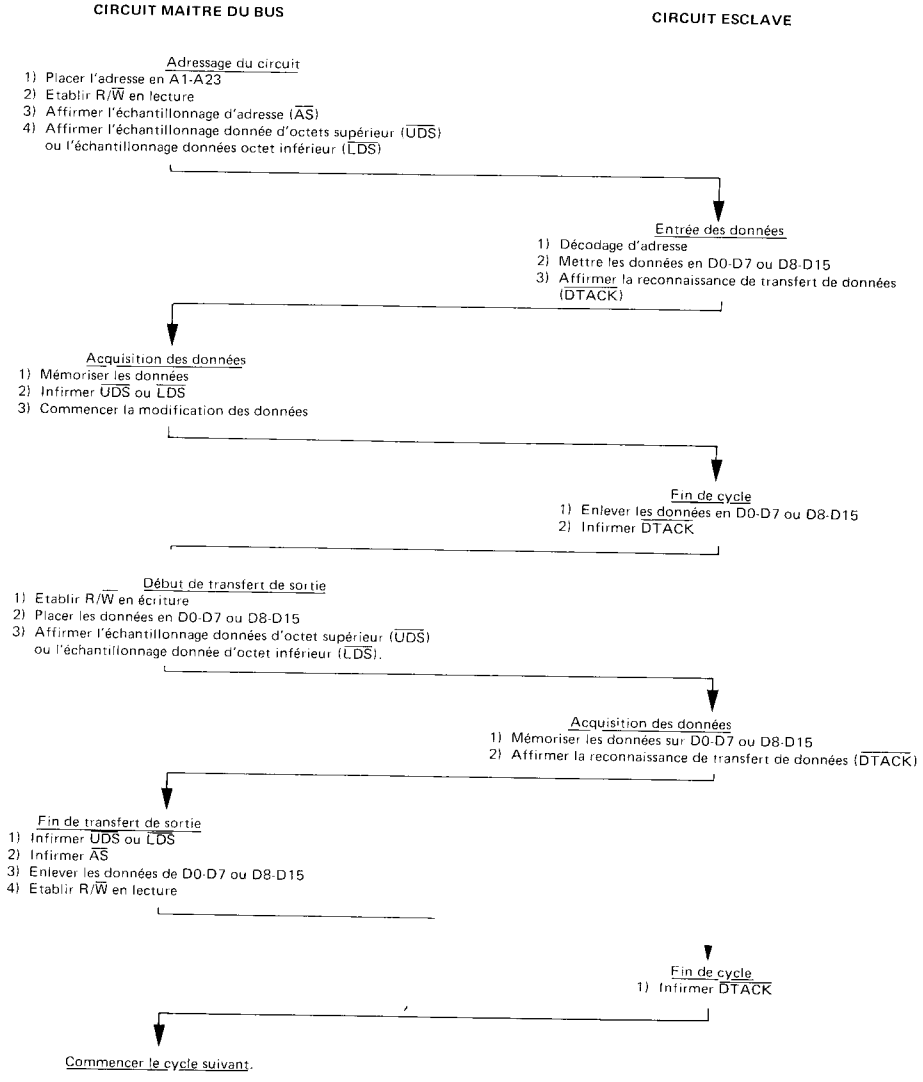
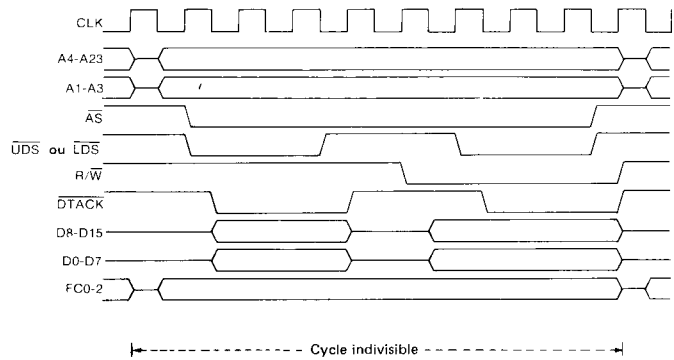


FIGURE 18 – DIAGRAMME DES TEMPS D'UN CYCLE LECTURE - MODIFICATION - ECRITURE



ARBITRAGE D'ATTRIBUTION DU BUS. Cet arbitrage est une technique utilisée par les circuits de type maître pour demander, obtenir et reconnaître la maîtrise du bus. Dans sa forme la plus simple, elle consiste en :

1. Affirmation d'une demande de maîtrise de bus
2. Réception d'une garantie que le bus sera libre à la fin du cycle en cours.
3. Reconnaissance que la maîtrise est bien assumée.

La figure 19 est un organigramme montrant en détail ce qu'implique une demande par un seul circuit. La figure 20 montre un diagramme des temps pour les mêmes opérations. Cette technique permet le traitement de requêtes de bus pendant les cycles de transfert de données.

Le diagramme des temps montre que la demande de bus est infirmée au moment où une reconnaissance est affirmée. Ce type d'opération serait vrai pour un système se composant du processeur et d'un seul circuit pouvant s'assurer la maîtrise du bus. Dans des systèmes où plusieurs circuits peuvent prendre la maîtrise du bus, la ligne de requête de bus de chaque circuit fait partie d'un OU câblé avec le processeur. Dans ce système, il apparaît clairement qu'il peut y avoir plusieurs demandes de bus faites. Le diagramme des temps montre que le signal bus accordé est infirmé quelques périodes d'horloge après la transition du signal de reconnaissance ($\overline{BGACK}$).

Cependant, si les demandes de bus sont toujours en attente, le processeur affirmera à nouveau le signal bus accordé quelques cycles d'horloge après qu'il soit infirmé. Cette affirmation supplémentaire d'accord de bus permet à la circuiterie externe d'arbitrage de sélectionner le circuit qui deviendra le maître du bus avant que le maître du bus en cours n'ait terminé ses demandes. Les paragraphes suivants fournissent des informations complémentaires sur les trois étapes du procédé d'arbitrage.

FIGURE 19 – ORGANIGRAMME DU CYCLE D'ARBITRAGE D'ATTRIBUTION DE BUS

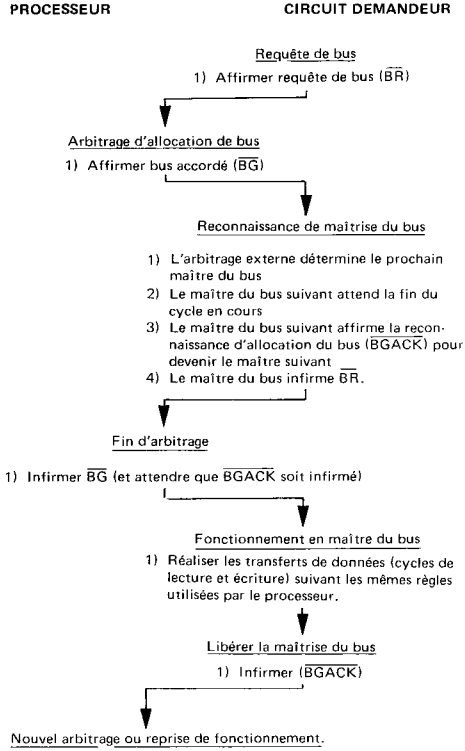
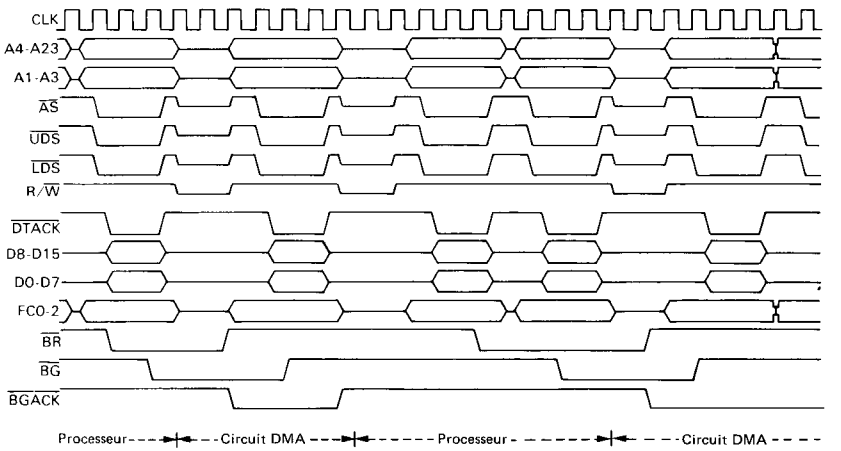


FIGURE 20 – DIAGRAMME DES TEMPS D'UN CYCLE D'ARBITRAGE D'ATTRIBUTION DU BUS.



Demande du bus. Les circuits externes capables de prendre la maîtrise du bus demandent le bus par l'affirmation du signal demande de bus (BR). C'est un signal OU câblé (bien qu'il ne nécessite pas d'être réalisé à l'aide de circuits à collecteur ouvert) qui indique au processeur qu'un circuit externe demande le contrôle du bus. Le processeur est effectivement à un niveau de priorité sur le bus inférieur à celui d'un circuit externe et il abandonnera le bus dès qu'il aura terminé le dernier cycle bus commencé.

Si aucune reconnaissance n'est reçue avant que le signal demande de bus soit inactif, le processeur continuera le traitement lorsqu'il détectera que la demande de bus est inactive. Ceci permet de continuer le traitement dans le cas où le circuit d'allocation a répondu à une sollicitation parasite par inadvertance.

Réception de bus accordé. Le processeur accorde le bus (BG) dès que possible. Normalement ceci se produit immédiatement après synchronisation interne. La seule exception à cette règle a lieu lorsque le processeur après la décision interne d'exécuter le cycle bus suivant n'a pas progressé suffisamment loin dans le cycle pour avoir affirmé le signal échantillonnage d'adresse (AS). Dans ce cas, l'accord de bus ne sera affirmé qu'un coup d'horloge après l'affirmation du signal d'échantillonnage d'adresse pour indiquer aux circuits extérieurs qu'un cycle bus est en cours d'exécution. Le signal d'accord de bus peut cheminer dans un réseau chaîné ou à travers un réseau spécifique d'encodage de priorité. Le processeur n'est pas concerné par la méthode externe d'arbitrage aussi longtemps que le protocole est respecté.

Reconnaissance de maîtrise du bus. Sur réception d'un signal allocation de bus, le circuit demandeur attend jusqu'à ce que les signaux échantillonnage d'adresse, reconnaissance de transfert de données, et signal de reconnaissance d'allocation de bus soient infirmés avant d'établir son propre BGACK. L'infirmation du signal d'échantillonnage d'adresse indique que le précédent circuit maître a terminé son cycle, l'infirmation du signal reconnaissance d'allocation de bus indique que le précédent circuit maître a abandonné le bus. (Aucun circuit n'est autorisé à "pénétrer" dans un cycle pendant que l'échantillonnage d'adresse est affirmé). L'infirmation de reconnaissance de transfert indique que le circuit esclave précédent a terminé sa connection au circuit maître précédent. Noter que dans certaines applications la reconnaissance de transfert de données peut ne pas entrer dans cette fonction. Des circuits universels pourraient être alors connectés de telle sorte qu'ils ne soient fonction que de l'échantillonnage d'adresse. Lorsque la reconnaissance d'allocation de bus est établie le circuit est maître du bus tant qu'il n'infirme pas le signal de reconnaissance d'allocation du bus. Le signal de reconnaissance d'allocation du bus ne peut être infirmé qu'après que le(s) cycle(s) bus soi(en)t terminé(s). La maîtrise du bus se termine à l'infirmation du signal de reconnaissance d'allocation du bus. La requête de bus par le circuit alloué doit tomber lorsque le signal de reconnaissance d'allocation de bus et affirmé. Si le signal requête de bus est toujours affirmé après que le signal de reconnaissance d'allocation de bus soit infirmé, le processeur réalise une autre séquence d'arbitrage et délivre un autre accord de bus. Il faut noter que le processeur ne réalise aucun cycle bus extérieur tant qu'il n'a pas réaffirmé le signal d'accord de bus.

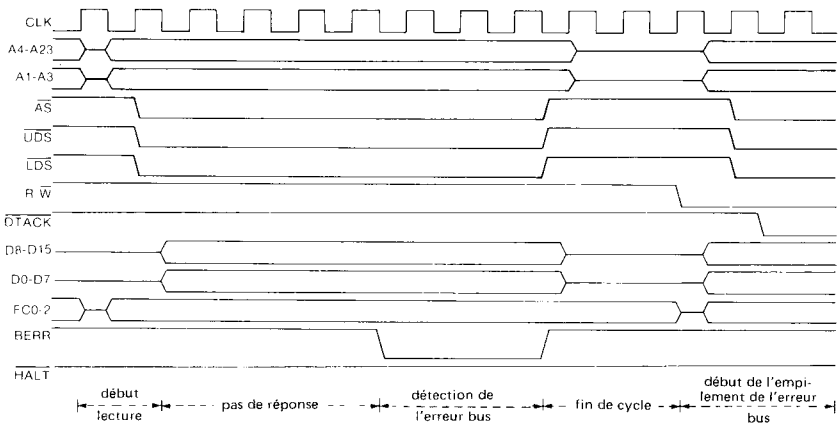
ERREUR BUS ET ARRET. Dans une architecture de bus fonctionnant en appel-réponse avec un circuit extérieur, il se peut que l'appel-réponse ne se produise pas. Puisque des systèmes différents ont des temps de réponse maximum différents, une entrée erreur bus est fournie. Une circuiterie extérieure doit être utilisée pour déterminer la durée entre l'échantillonnage d'adresse et la reconnaissance de transfert de données avant d'émettre un signal d'erreur bus. Lorsqu'un signal d'erreur bus est reçu, le processeur a deux options : soit initialiser une séquence d'exception d'erreur bus, soit essayer de relancer le cycle bus .

Séquence d'exception. La séquence d'exception d'erreur bus survient lorsque le processeur reçoit un signal d'erreur bus et que la broche arrêt est inactive. La figure 21 montre un diagramme des temps de séquence d'exception. La séquence est composée des éléments suivants :

1. Empilement du compteur programme et du registre d'état
2. Empilement de l'information d'erreur
3. Lecture de l'enregistrement en table vecteur d'erreur bus
4. Exécution du sous programme de traitement d'erreur bus.

L'empilement du compteur programme et du registre d'état se passe comme si une interruption était survenue. Lors d'une erreur bus plusieurs renseignements complémentaires sont empilés. Ces renseignements sont utilisés pour déterminer la nature de l'erreur et la corriger si possible. Le vecteur erreur bus est le vecteur numéro deux situé à l'adresse $ 000008. Le processeur charge le nouveau compteur programme depuis cet emplacement. Un sous-programme logiciel de traitement d'erreur bus est alors exécuté par le processeur. Reportez-vous au chapitre **TRAITEMENT D'EXCEPTION** pour des informations complémentaires.

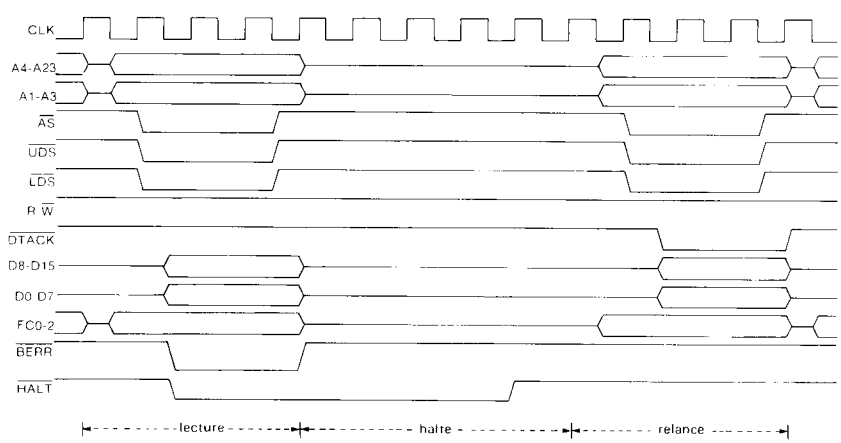
FIGURE 21 – DIAGRAMME DES TEMPS D'UNE ERREUR BUS



Relance du cycle bus. Lorsque le processeur reçoit un signal d'erreur bus et que la broche arrêt est conduite par un circuit extérieur, le processeur démarre une séquence de relance. La figure 22 montre un diagramme des temps de relance du cycle bus.

Le processeur termine le cycle bus, puis place les lignes d'adresse, de données, de code fonction et de commande en état haute impédance. Le processeur demeure "arrêté" et ne commence un autre cycle bus que lorsque le signal arrêt est effacé par une logique externe. Le processeur

FIGURE 22 – INFORMATION SUR LE DIAGRAMME DES TEMPS D'UN CYCLE DE RELANCE



peut alors relancer le cycle bus précédent en utilisant la même adresse, les mêmes codes fonction, les mêmes données (pour une opération écriture) et les mêmes commandes. Le signal d'erreur bus doit être enlevé avant que le signal arrêt soit enlevé.

NOTE

Le processeur ne relance pas un cycle modification-lecture-écriture. Cette restriction est faite pour garantir que le cycle entier se déroule correctement et que l'écriture dans une opération de test-et-établissement d'un indicateur (TAS) est réalisée sans jamais libérer AS.

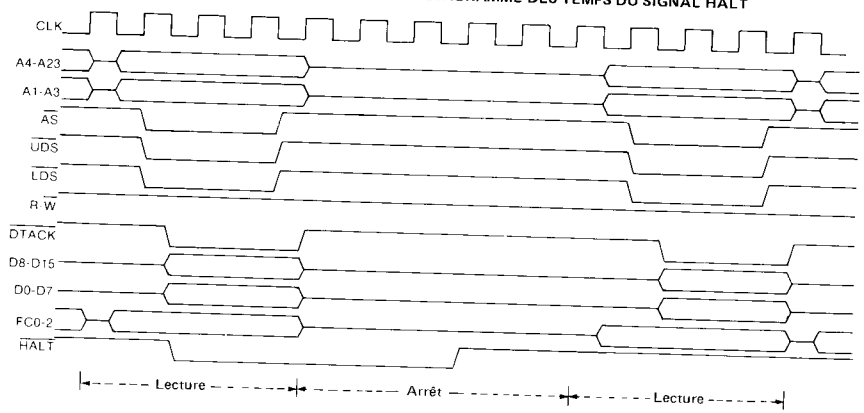
Opération d'arrêt sans erreur bus. Le signal d'entrée arrêt vers le EF68000 réalise une fonction pas à pas /marche/arrêt de la même manière que la fonction arrêt du circuit EF6800. Les modes arrêt et marche parlent d'eux mêmes dans le sens que, lorsque le signal arrêt est constamment actif, le processeur s'arrête (ne fait rien) et lorsque le signal est constamment inactif le processeur fonctionne (fait quelque chose).

Le mode pas à pas dérive de transitions correctement calées, sur l'entrée signal d'arrêt. Il force le processeur à exécuter un cycle bus unique en introduisant le mode "marche" jusqu'à ce que le processeur démarre un cycle bus puis passe en mode "arrêt".

Ainsi, le mode pas à pas permet à l'utilisateur d'exécuter (et par conséquent de mettre au point) les instructions du processeur cycle par cycle.

La figure 23 détaille le diagramme des temps requis pour des opérations pas à pas correctes. Il faut prendre soin d'éviter des intéractions nuisibles entre le signal d'erreur bus et la broche arrêt, lorsque le mode cycle unique est utilisé pour la mise au point. Ceci est vrai aussi pour des interactions entre les lignes d'arrêt et d'initialisation, celles-ci pouvant remettre à zéro la machine.

FIGURE 23 – CARACTERISTIQUES DU DIAGRAMME DES TEMPS DU SIGNAL HALT



Lorsque le processeur termine un cycle bus après avoir reconnu que le signal d'arrêt est actif, la plupart des signaux trois états sont mis à l'état haute impédance. Ceux-ci comprennent :

1. Les lignes d'adresse
2. Les lignes de données
3. Les lignes de code fonction

Ceci est nécessaire pour la réalisation correcte de l'opération de relance de cycle bus.

Notez que lorsque le processeur honore une requête d'arrêt, les codes fonction passent à l'état haute impédance (les caractéristiques de leurs amplificateurs sont les mêmes que les amplificateurs d'adresse). Tandis que le processeur honore une requête d'arrêt, l'arbitrage d'allocation du bus est réalisé comme d'habitude. Ceci étant, le passage à l'arrêt n'a pas d'effet sur l'arbitrage du bus. C'est la fonction d'arbitrage d'allocation de bus qui efface les signaux de commande du bus.

Le concepteur peut, lors de la mise au point, exécuter les instructions cycle par cycle ou exécuter son programme instruction par instruction grâce à la fonction arrêt. Ces possibilités offertes par le processeur utilisées en jonction avec un programme d'aide à la mise au point donnent une grande souplesse de mise au point.

Double erreur bus. Lorsqu'une exception erreur bus survient, le processeur essaie d'empiler certains mots contenant des informations sur l'état de la machine. Si une exception erreur bus survient durant l'empilement, il y a deux erreurs bus d'affilée. Ceci est habituellement désigné par double erreur bus. Lorsqu'une double erreur bus survient, le processeur se met sur arrêt. Lorsqu'une exception erreur bus survient, toute exception erreur bus survenant avant l'exécution de l'instruction suivante, constitue une double erreur bus.

Noter que la relance d'un cycle bus ne constitue pas une exception erreur bus et ne contribue pas à une double erreur bus. Noter aussi que ceci signifie que le processeur continue à relancer le même cycle bus aussi longtemps que la circuiterie externe le requiert.

La broche erreur bus a aussi une action sur le fonctionnement du processeur après que le processeur ait reçu une entrée de remise à zéro externe. Le processeur lit la table vecteur après une remise à zéro pour déterminer l'adresse de départ du programme à exécuter. Si une erreur bus survient pendant la lecture de la table vecteur (ou à tout instant avant que la première instruction soit exécutée), le processeur réagit comme si une double erreur bus était survenue et s'arrête. Seule une remise à zéro externe démarre un processeur à l'arrêt.

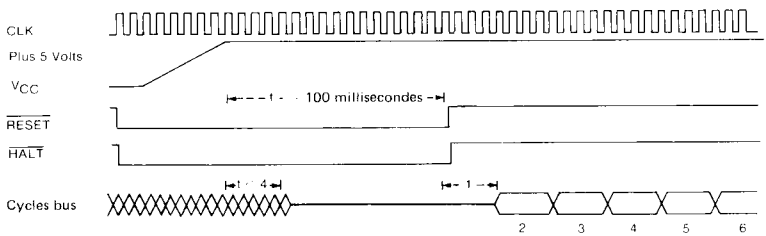
INITIALISATION. Le signal d'initialisation est un signal bi-directionnel permettant au processeur ou à un signal externe d'initialiser le système. La figure 24 est un diagramme des temps des opérations d'initialisation. Les lignes d'arrêt et d'initialisation doivent être appliquées en même temps pour assurer une mise à zéro globale du processeur.

Lorsque les lignes d'initialisation et d'arrêt sont conduites par un circuit externe, on obtient une initialisation globale du système, y compris du processeur. Le processeur répond en lisant l'enregistrement en table vecteur d'initialisation (numéro de vecteur zéro, adresse $ 000000) et le charge dans le pointeur de pile superviseur (SSP). L'enregistrement en table vecteur numéro un, à l'adresse $ 000004 est ensuite lu et chargé dans le compteur programme. Le processeur initialise le registre d'état à un niveau d'interruption sept. Aucun autre registre n'est affecté par la séquence d'initialisation.

Lorsqu'une instruction de mise à zéro des circuits externes RESET est exécutée, le processeur pilote la broche d'initialisation (reset) pendant 124 périodes d'horloge. Dans ce cas, le processeur essaie de mettre à zéro le reste du système. Par conséquent, il n'y a aucun effet sur l'état interne du processeur. Tous les registres internes du processeur et le registre d'état ne sont pas concernés par l'exécution d'une instruction de mise à zéro RESET. Tous les circuits externes connectés à la ligne RESET sont mis à zéro à l'achèvement d'une instruction RESET.

Lorsque V_{CC} est appliqué initialement au processeur, un signal externe doit être appliqué sur la broche d'initialisation (reset) pendant 100 millisecondes.

FIGURE 24 – DIAGRAMME DES TEMPS DE LA MISE SOUS TENSION



Notes
1) Temps démarrage interne
2) Lecture poids fort SSP
3) Lecture poids faible SSP
4) Lecture poids fort PC
5) Lecture poids faible PC
6) Recherche de la première instruction.

TRAITEMENT EXCEPTIONNEL

Les paragraphes suivants décrivent les actions du circuit EF68000 qui sortent du traitement normal lié à l'exécution des instructions. Les fonctions des bits dans la partie superviseur du registre d'état sont traitées : le bit superviseur/utilisateur, le bit validation du mode trace, et le masque de priorité des interruptions du processeur. Enfin, est détaillée la séquence des références mémoire et des actions prises par le processeur sur des conditions d'exception.

ETATS DE TRAITEMENT

Le circuit EF68000 est toujours dans l'un des trois états de traitement suivants : normal, d'exception, ou arrêté. L'état de traitement normal est celui associé à l'exécution des instructions. Les références mémoire servent à aller chercher instructions et opérandes et à stocker les résultats. Un cas particulier de l'état normal est l'état suspendu dans lequel le processeur se met lorsqu'une instruction d'arrêt (STOP) est exécutée. Dans ce cas, aucune référence mémoire ultérieure n'est faite.

L'état de traitement d'exception est associé aux interruptions, instructions trappe, mode trace et autres conditions exceptionnelles. L'exception peut être générée de façon interne par une instruction ou par une condition inhabituelle survenant pendant l'exécution d'une instruction. De manière externe, le traitement d'exception peut être forcé par une interruption, par une erreur bus, ou par une mise à zéro. Le traitement d'exception est conçu pour fournir un changement de contexte efficace pour que le processeur puisse traiter les conditions inhabituelles.

L'état arrêté indique une défaillance matérielle catastrophique. Par exemple, si en cours de traitement d'exception d'une erreur bus, une autre erreur bus survient, le processeur admet que le système est inutilisable et s'arrête. Seule une mise à zéro externe peut redémarrer un processeur arrêté. Noter que le processeur en état suspendu n'est pas en état arrêté, et vice versa.

ETATS PRIVILÉGIÉS

Le processeur fonctionne dans l'un des deux états dits privilégiés : l'état "utilisateur" ou l'état "superviseur". L'état privilégié, qui détermine quelles opérations sont autorisées, est utilisé par le circuit externe de gestion de la mémoire pour contrôler et translater des accès, et est utilisé pour choisir entre le pointeur de pile superviseur et le pointeur de pile utilisateur pour référencer les instructions.

L'état privilégié est un mécanisme donnant une sécurité dans un système informatique. Les programmes doivent accéder seulement à leurs propres espaces de codage et données, et ne peuvent avoir accès aux informations dont ils n'ont pas besoin et qu'ils ne doivent pas modifier.

Le mécanisme de privilège augmente la protection en permettant à la plupart des programmes de s'exécuter en état utilisateur. Dans cet état, leurs accès sont contrôlés et leurs effets sur les autres parties du système sont limités. Le système d'exploitation est exécuté en état superviseur, il a accès à toutes les ressources, et réalise les tâches générales des programmes utilisateur.

ETAT SUPERVISEUR. L'état superviseur est l'état de plus haut privilège. Pour l'exécution des instructions, l'état superviseur est déterminé par le bit S du registre d'état. Si le bit S est affirmé (haut), le processeur est en état superviseur. Toutes les instructions peuvent être exécutées dans l'état superviseur. Les cycles bus générés par des instructions exécutées en état superviseur sont classées en références superviseur. Tant que le processeur se trouve dans l'état privilégié superviseur, les instructions qui utilisent soit implicitement le pointeur de pile système, soit explicitement le registre d'adresses sept, accèdent au pointeur de pile superviseur.

Tout traitement exceptionnel est réalisé en état superviseur, indépendamment du positionnement du bit S. Les cycles bus générés en cours de traitement exceptionnel sont classés en référence superviseur. Toutes opérations d'empilement en cours de traitement exceptionnel utilisent le pointeur de pile superviseur.

ETAT UTILISATEUR. L'état utilisateur est l'état de moindre privilège. L'état utilisateur est déterminé par le bit S du registre d'état pour l'exécution des instructions, si le bit S est infirmé (bas), le processeur exécute les instructions dans l'état utilisateur. La plupart des instructions s'exécutent en état utilisateur de la même manière qu'en état superviseur.Cependant, certaines instructions qui ont des effets importants sur le système sont rendues privilégiées. Les programmes utilisateurs n'ont pas le droit d'utiliser une instruction d'arrêt (STOP), ou une instruction d'initialisation (RESET). Pour s'assurer qu'un programme utilisateur ne puisse pas passer dans l'état superviseur, sauf de manière contrôlée, les instructions qui modifient le registre d'état en totalité sont privilégiées. Pour faciliter la mise au point des systèmes d'exploitation, les instructions de transfert vers pointeur de pile utilisateur (MOVE vers USP) et transfert depuis le pointeur de pile utilisateur (MOVE depuis USP) sont aussi privilégiées.

Les cycles bus générés par une instruction exécutée en état utilisateur sont classés comme références état utilisateur. Ceci permet à un circuit externe de gestion de mémoire de translater l'adresse et de contrôler l'accès à des parties protégées de l'espace adresse. Tant que le processeur se trouve dans l'état privilégié utilisateur, les instructions qui utilisent soit le pointeur de pile système implicitement soit le registre d'adresses sept explicitement, accèdent au pointeur de pile utilisateur.

CHANGEMENT D'ETAT PRIVILÉGIÉ. Une fois que le processeur est dans l'état utilisateur, et qu'il exécute des instructions, seul un traitement d'exception peut modifier l'état privilégié. Au cours d'un traitement d'exception, la valeur en cours du bit S du registre d'état est sauvegardée et le bit S est affirmé, plaçant le traitement en état superviseur. En conséquence, lorsque l'exécution d'une instruction reprend à l'adresse spécifiée pour traiter l'exception, le processeur est dans l'état privilégié superviseur.

CLASSIFICATION DES RÉFÉRENCES. Lorsque le processeur fait une référence, il classe le type de référence faite à l'aide de codes placés sur les trois lignes de sortie codes fonction. Ceci permet une translation externe des adresses, le contrôle des accès, et la différenciation des états spéciaux du processeur tel que reconnaissance des interruptions. Le tableau 17 liste la classification des références.

TABLEAU 17 – CLASSIFICATION DES RÉFÉRENCES

Sortie codes fonction			
FC2	FC1	FC0	Classe des références
0	0	0	(non octroyé)
0	0	1	Donnée utilisateur
0	1	0	Programme utilisateur
0	1	1	(non octroyé)
1	0	0	(non octroyé)
1	0	1	Données superviseur
1	1	0	Programme superviseur
1	1	1	Reconnaissance d'interruption

TRAITEMENT D'EXCEPTION

Avant de voir en détail les interruptions, les trappes et trace, une description générale du traitement d'exception est donnée. Le traitement d'une exception se déroule en quatre étapes, avec des variantes selon les causes d'exception. Pendant la première étape, une copie temporaire du registre d'état est faite, et le registre d'état est établi pour un traitement d'exception. Dans la deuxième étape, le vecteur d'exception est déterminé et la troisième étape est la sauvegarde du contexte actuel du processeur. Dans la quatrième étape, un nouveau contexte est obtenu et le processeur passe en traitement d'instruction.

VECTEURS D'EXCEPTION. Les vecteurs d'exception sont des positions mémoire dans lesquelles le processeur recherche l'adresse d'un sous-programme qui traitera cette exception. Tous les vecteurs d'exception ont une longueur de deux mots (figure 25) sauf le vecteur d'initialisation qui tient sur quatre mots. Tous les vecteurs d'exception résident dans l'espace données superviseur, sauf le vecteur d'initialisation qui se trouve dans l'espace programme superviseur. Un numéro de vecteur est un nombre de huit bits qui, multiplié par quatre, donne l'adresse d'un vecteur d'exception. Les numéros de vecteur sont générés de façon interne ou externe, en fonction de la cause de l'exception. Dans le cas des interruptions, pendant un cycle bus de reconnaissance d'interruption, un périphérique fournit un numéro vecteur de 8 bits (figure 26) au processeur sur les lignes bus données D0 à D7. Le processeur convertit le numéro vecteur en une adresse 24 bits, comme indiqué en figure 27. La disposition en mémoire des vecteurs d'exception est donnée dans le tableau 18.

FIGURE 25 – FORMAT DES VECTEURS D'EXCEPTION

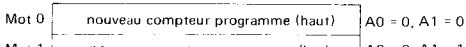
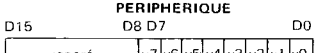
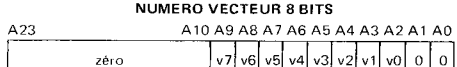


FIGURE 26 – FORMAT DU NUMÉRO VECTEUR PERIPHERIQUE



où : V7 est le bit de poids fort du numéro de vecteur
V0 est le bit de poids faible du numéro de vecteur

FIGURE 27 – CONVERSION D'ADRESSE A PARTIR DU NUMERO VECTEUR 8 BITS



Comme le montre le tableau 18, la disposition en mémoire a une longueur de 512 mots (1024 octets). Elle part de l'adresse 0 et va jusqu'à l'adresse 1023. Ceci fournit 255 vecteurs uniques. Certains sont réservés pour les trappes, d'autres pour des fonctions système. Sur les 255, 192 sont réservés aux vecteurs d'interruption utilisateur. Cependant, il n'y a pas de protection sur les 64 premiers enregistrements, ainsi des vecteurs d'interruption utilisateur peuvent y être mis à la discrétion du concepteur des systèmes.

SORTES D'EXCEPTIONS. Les exceptions peuvent être générées par des causes internes ou externes. Les exceptions générées de façon externe sont les interruptions, les demandes d'initialisation et les erreurs bus.

Les interruptions sont des requêtes de circuits périphériques demandant une action au processeur tandis que les entrées d'initialisation et erreur bus ont pour but d'initialiser et de contrôler les accès du processeur. Les exceptions générées de façon interne proviennent d'instructions ou d'erreurs d'adresse, ou du mode trace. Les instructions trappe (TRAP), trappe sur dépassement (TRAPV), test registre aux limites (CHK) et division (DIV) peuvent toutes générer des exceptions comme faisant partie de leur exécution. De plus, les instructions invalides, les recherches de mots à partir d'adresses impaires et les violations de privilège sont des causes d'exceptions. Le mode trace se comporte comme une interruption générée de manière interne, de très haute priorité, après chaque exécution d'instruction.

TABLEAU 18 – AFFECTATION DES VECTEURS D'EXCEPTION

Numéro vecteur(s)	Adresses			Affectation
	Dec	Hex	Espace	
0	0	000	SP	Initialisation de SSP
	4	004	SP	Initialisation de PC
2	8	008	SD	Erreur bus
3	12	00C	SD	Erreur adresse
4	16	010	SD	Instruction interdite
5	20	014	SD	Division par zéro
6	24	018	SD	Instruction CHK
7	28	01C	SD	Instruction TRAPV
8	32	020	SD	Violation de privilège
9	36	024	SD	Trace
10	40	028	SD	Emulateur ligne 1010
11	44	02C	SD	Emulateur ligne 1111
12*	48	030	SD	(non attribué, réservé)
13*	52	034	SD	(non attribué, réservé)
14*	56	038	SD	(non attribué, réservé)
15*	60	03C	SD	(non attribué, réservé)
16-23*	64	04C	SD	(non attribué, réservé)
	95	05F		–
24	96	060	SD	Interruption parasite
25	100	064	SD	Auto vecteur interruption niveau 1
26	104	068	SD	Auto vecteur interruption niveau 2
27	108	06C	SD	Auto vecteur interruption niveau 3
28	112	070	SD	Auto vecteur interruption niveau 4
29	116	074	SD	Auto vecteur interruption niveau 5
30	120	078	SD	Auto vecteur interruption niveau 6
31	124	07C	SD	Auto vecteur interruption niveau 7
32-47	128	080	SD	Vecteurs d'instruction TRAP
	191	0BF		–
48-63*	192	0C0	SD	(non attribué, réservé)
	255	0FF		–
64-255	256	100	SD	Vecteurs interruption utilisateur
	1023	3FF		–

* Les numéros de vecteurs de 12 à 23 et de 48 à 63 sont réservés par EFCIS pour des extensions futures. L'utilisateur ne doit affecter aucun circuit périphérique à ces numéros de vecteur.

SEQUENCE DE TRAITEMENT D'EXCEPTION.

Le traitement d'exception a lieu en quatre étapes distinctes. Dans la première étape, une copie interne du registre d'état est faite. Une fois la copie faite, le bit S est affirmé, mettant le processeur dans l'état privilégié superviseur. Le bit T est aussi infirmé, ce qui permet au programme de traitement d'exception de s'exécuter sans être gêné par le mode trace. Pour les exceptions d'interruption et d'initialisation, le masque de priorité des interruptions est aussi mis à jour.

Dans la seconde étape, le numéro du vecteur de l'exception est déterminé. Pour les interruptions, le numéro de vecteur est obtenu par une recherche du processeur, classée comme une reconnaissance d'interruption. Pour toutes les autres exceptions, la logique interne fournit le numéro de vecteur. Ce numéro de vecteur est alors utilisé pour générer l'adresse du vecteur d'exception.

La troisième étape concerne la sauvegarde de l'état actuel du processeur, sauf pour l'exception de mise à zéro. La valeur courante du compteur programme et la copie sauvegardée du registre d'état sont empilées en utilisant le pointeur de pile superviseur. La valeur du compteur programme habituellement empilée pointe l'instruction suivante à exécuter, cependant, lors d'une erreur bus et d'une erreur adresse, la valeur du compteur programme mise sur la pile est imprévisible, et peut être incrémentée par rapport à l'adresse de l'instruction qui a causé l'erreur. Des informations complémentaires définissant le contexte courant sont empilées pour les exceptions d'erreur bus et d'erreur adresse.

La dernière étape est la même pour toutes les exceptions. La nouvelle valeur du compteur programme est cherchée à partir du vecteur d'exception. Le processeur reprend alors l'exécution des instructions. L'instruction à l'adresse donnée dans le vecteur d'exception est chargée, et le décodage puis l'exécution de l'instruction sont effectués.

EXCEPTIONS MULTIPLES. Ces paragraphes décrivent le traitement qui a lieu lorsque des exceptions multiples surviennent simultanément. Les exceptions peuvent être groupées en fonction de leur apparition et de leur priorité. Les exceptions du groupe 0 concernent : initialisation, erreur bus et erreur adresse. Ces exceptions provoquent l'avortement de l'instruction en cours d'exécution, et le traitement d'exception démarre au prochain cycle mineur du processeur. Les exceptions du groupe 1 concernent les interruptions et le mode trace ainsi que les violations de privilège et les instructions invalides. Ces exceptions permettent à l'instruction en cours de s'exécuter complètement, et elles préparent l'exécution de l'instruction suivante en forçant le traitement d'exception à avoir lieu (les violations de privilège et les instructions invalides sont détectées lorsqu'elles constituent la prochaine instruction à exécuter). Les exceptions du groupe 2 font partie des instructions de traitement normal. Les exceptions TRAP, TRAPV, CHK et division par zéro appartiennent à ce groupe. Pour ces exceptions, l'exécution normale d'une instruction peut conduire à un traitement d'exception.

Les exceptions du groupe 0 ont la priorité la plus élevée, tandis que les exceptions du groupe 2 ont la priorité la plus basse. Dans le groupe 0, la mise à zéro (reset) a la priorité la plus élevée, suivie par erreur bus puis erreur adresse. Dans le groupe 1, trace est prioritaire sur les interruptions externes, qui sont à leur tour prioritaires sur les instructions invalides et les violations de privilège. Une seule instruction pouvant être exécutée à la fois, il n'y a pas de relation de priorité dans le groupe 2. La relation de priorité entre deux exceptions détermine laquelle est prise, ou prise la première si les conditions d'établissement surviennent pour les deux simultanément. Par conséquent, si une erreur bus survient au cours d'une instruction TRAP, l'erreur bus est prioritaire et l'exécution de l'instruction TRAP est avortée. Autre exemple, si une demande d'interruption survient pendant l'exécution d'une instruction lorsque le bit T est affirmé, l'exception trace est prioritaire et est traitée en premier. L'exception d'interruption sera traité avant que l'exécution des instructions se poursuive. Le traitement des instructions commence finalement dans le sous programme d'interruption. Un résumé des priorité et groupement des exceptions est donné dans le tableau 19.

TABLEAU 19 – PRIORITE ET GROUPEMENT DES EXCEPTIONS

Groupe	Exception	Traitement
0	Mise à zéro erreur bus erreur adresse	le traitement des exceptions commence au cycle mineur suivant
1	Trace Interruptions Invalide Privilégié	le traitement des exceptions commence avant l'instruction suivante
2	TRAP TRAPV CHK division par zéro	le traitement d'exception est démarré par l'exécution d'instruction normale

TRAITEMENT D'EXCEPTION, DISCUSSION DETAILLÉE.

Les exceptions ont plusieurs sources, et chaque exception a un traitement qui lui est particulier. Les paragraphes suivants détaillent les sources des exceptions, comment chacune survient, et comment chacune est traitée.

MISE A ZERO (RESET). L'entrée de mise à zéro (reset) fournit l'exception de plus haute priorité. Le traitement du signal de mise à zéro est conçu pour permettre l'initialisation du système, et le recouvrement après une panne. Tout traitement en cours au moment d'une mise à zéro est avorté et ne peut être recouvré. Le processeur est forcé dans l'état superviseur, et l'état trace est mis hors fonction. Le masque des interruptions processeur est mis au niveau sept. Le numéro de vecteur est généré de façon interne pour référencer le vecteur d'exception de mise à zéro à l'emplacement 0 de l'espace programme superviseur. Aucune supposition ne pouvant être faite sur la validité du contenu des registres, en particulier du pointeur de pile superviseur, ni le compteur programme, ni le registre d'état ne sont sauvegardés. L'adresse contenue dans les deux premiers mots du vecteur d'exception de mise à zéro est placée dans le pointeur de pile superviseur, et l'adresse dans les deux derniers mots du vecteur d'exception de mise à zéro est utilisée comme valeur initiale du compteur programme. Finalement, l'exécution d'instructions commence à l'adresse contenue dans le compteur programme. La valeur initiale du compteur programme doit pointer le programme de mise sous tension/initialisation.
L'instruction de mise à zéro (RESET) n'entraîne pas le chargement du vecteur, de mise à zéro, mais réalise l'affirmation de la ligne reset pour initialiser les circuits externes. Ceci permet au logiciel d'initialiser le système dans un état connu, puis de continuer le traitement avec l'instruction suivante.

INTERRUPTIONS. Sept niveaux d'interruptions prioritaires sont fournis. Des circuits peuvent être chaînés extérieurement sous niveaux d'interruption prioritaire, permettant à un nombre illimité de circuits périphériques d'interrompre le processeur.

Les niveaux de priorité des interruptions sont numérotés de un à sept, le niveau sept ayant la priorité la plus élevée. Le registre d'état contient un masque de trois bits qui indique la priorité en cours, et les interruptions sont inhibées pour tous les niveaux de priorité plus petits ou égaux au niveau de priorité actuel du processeur.

Une demande d'interruption est faite au processeur en codant le niveau de demande sur les lignes de demande d'interruption. Un zéro indique l'absence de demande. Les demandes d'interruption arrivant au processeur ne forcent pas un traitement d'exception immédiat, mais sont mises en attente. Les interruptions en attente sont détectées entre les exécutions d'instructions. Si la priorité de l'interruption en attente est inférieure ou égale à la priorité en cours, le traitement se poursuit avec l'instruction suivante et le traitement d'exception de l'interruption est différé. (la reconnaissance du niveau sept est un peu différente, comme expliqué dans un paragraphe suivant).

Si la priorité de l'interruption arrivante est supérieure à la priorité en cours, la séquence de traitement d'exception est lancée. Tout d'abord une copie du registre d'état est sauvegardée, et l'état privilégié passe dans l'état superviseur, le mode trace est supprimé et le processeur prend le niveau de l'interruption venant d'être reconnue. Le processeur recherche le numéro de vecteur du circuit interrompant, il classe la référence comme une reconnaissance d'interruption et il dispose le numéro du niveau de priorité de l'interruption venant d'être reconnue, sur le bus d'adresses. Si une logique externe requiert une vectorisation automatique, le processeur génère de manière interne un numéro de vecteur qui est déterminé par le numéro de niveau d'interruption. Si une logique externe indique une erreur bus, l'interruption est reconnue comme étant parasite et le numéro de vecteur généré référence le vecteur d'interruption parasite. Le processeur procède alors selon le traitement d'exception ordinaire, en sauvegardant le compteur programme et le registre d'état sur la pile superviseur. La valeur du compteur programme sauvegardée est l'adresse de l'instruction qui aurait été exécutée si l'interruption n'était pas survenue. Le contenu du vecteur interruption dont le numéro de vecteur était précédemment obtenu, est chargé dans le compteur programme, et l'exécution normale des instructions commence avec le sous-programme de traitement des interruptions.

Un organigramme de la séquence de reconnaissance des interruptions est donné en figure 28 ; un chronogramme est donné en figure 29.

FIGURE 28 – ORGANIGRAMME DE SEQUENCE DE RECONNAISSANCE D'INTERRUPTION

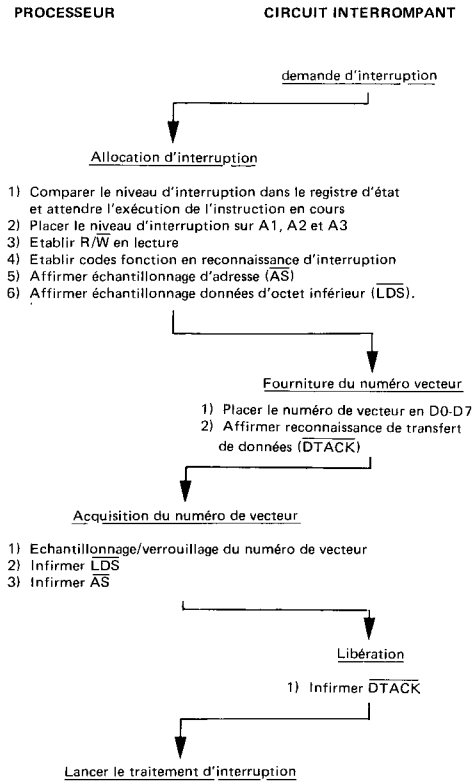
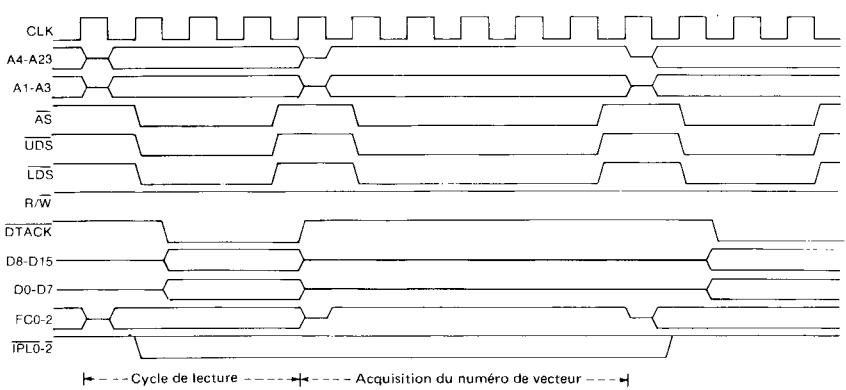


FIGURE 29 – DIAGRAMME DES TEMPS POUR LA SEQUENCE DE RECONNAISSANCE DES INTERRUPTIONS



Le niveau de priorité sept est un cas particulier. Les interruptions de niveau sept ne peuvent pas être inhibées par le masque des interruptions prioritaires, fournissant ainsi une interruption non-masquable. Une interruption est générée chaque fois que le niveau de demande d'interruption passe d'un niveau de priorité inférieur au niveau sept. Noter qu'une interruption de niveau sept peut toujours être provoquée par la comparaison des niveaux, si le niveau de demande est le niveau sept et si la priorité du processeur est mise à un niveau inférieur par une instruction.

INSTRUCTIONS TRAPPES. Les trappes sont des exceptions provoquées par des instructions. Elles surviennent soit à la suite de la reconnaissance par le processeur de conditions anormales au cours de l'exécution des instructions, soit de l'utilisation d'instructions dont le rôle est de générer un trappe.

Certaines instructions sont utilisées spécifiquement pour générer des trappes. L'instruction TRAP force toujours une exception et est très utile pour implémenter des appels système pour programme utilisateur. Les instructions TRAPV et CHK forcent une exception si le programme utilisateur détecte une erreur en cours d'exécution, qui peut être un dépassement arithmétique ou un indice hors des limites.

La division signée (DIVS) et la division non signée (DIVU) sont des instructions qui forcent une exception si une division par zéro est tentée.

INSTRUCTIONS INVALIDES ET NON IMPLÉMENTÉES. Le terme instruction invalide désigne une combinaison binaire qui ne correspond pas à la combinaison binaire du premier mot d'une instruction valide.

Si une telle instruction est rencontrée en cours d'exécution d'instructions, une exception d'instruction invalide se produit.

Les combinaisons binaires dont les bits 15 à 12 correspondent à 1010 ou 1111 se distinguent comme étant des instructions non implémentées et des vecteurs d'exception distincts leur sont attribués pour permettre une émulation efficace. Cette ressource permet au système d'exploitation de détecter des erreurs de programme, ou d'émuler des instructions non implémentées dans le logiciel.

VIOLATION DE PRIVILEGE. Afin d'assurer la sécurité du système, certaines instructions sont privilégiées. Une tentative d'exécution d'une instruction privilégiée alors que le système est en état utilisateur provoque une exception. Les instructions privilégiées sont :

STOP	AND (mot) immédiat avec SR
RESET	EOR (mot) immédiat avec SR
RTE	OR (mot) immédiat avec SR
MOVE to SR	MOVE USP

MODE TRACE. Pour aider au développement des programmes, le circuit EF68000 permet de tracer un programme instruction par instruction. En mode trace, une exception est forcée après l'exécution de chaque instruction, permettant à un programme de mise au point de contrôler l'exécution du programme testé.

Le mode trace utilise le bit T de la partie superviseur du registre d'état. Si le bit T est infirmé, le mode trace est inhibé, l'exécution des instructions se poursuit normalement instruction par instruction. Si le bit T est affirmé au début de l'exécution d'une instruction, une exception trace est générée à la fin de l'exécution de cette instruction. Si l'instruction n'est pas exécutée, soit parce qu'une interruption est prise en compte, soit parce que l'instruction est invalide ou privilégiée, l'exception trace n'a pas lieu. Elle n'apparaît pas non plus dans le cas où une instruction est avortée par une exception d'initialisation, erreur bus, ou erreur adresse.

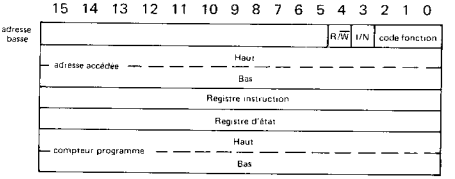
Si l'instruction est vraiment exécutée et qu'une interruption est en attente de prise en compte, l'exception trace est traitée avant l'exception d'interruption. Si, en cours d'exécution de l'instruction, une exception est forcée par cette instruction, l'exception forcée est traitée avant l'exception trace.

Comme illustration extrême des règles ci-dessus, considérer l'arrivée d'une interruption pendant l'exécution d'une instruction TRAP, tandis que le mode trace est invalidé. L'exception trappe est d'abord traitée, puis l'exception trace, enfin l'exception interruption. L'exécution d'instruction se poursuit dans le sous-programme de traitement des interruptions.

ERREUR BUS. Les exceptions erreur bus surviennent lorsque une logique externe requiert qu'une erreur bus soit traitée par une exception. Le cycle bus en cours, réalisé par le processeur, est avorté. Si le processeur exécutait une instruction ou traitait une exception, ce traitement est terminé, et le processeur commence immédiatement le traitement d'exception.

Le traitement d'exception d'erreur bus suit la séquence habituelle. Le registre d'état est copié, l'état superviseur est établi, et le mode trace est inhibé. Le numéro de vecteur est généré pour référer au vecteur erreur bus. Le processeur n'étant pas entre deux instructions lorsque la demande d'exception erreur bus est faite, le contexte du processeur est plus détaillé. Pour sauvegarder davantage de ce contexte, des informations supplémentaires sont sauvegardées sur la pile superviseur. Le compteur programme et la copie du registre d'état sont bien entendu sauvegardés. La valeur sauvegardée du compteur programme est augmentée de deux à dix octets par rapport à l'adresse du premier mot de l'instruction qui a fait la référence provoquant l'erreur bus. Si l'erreur bus est survenue pendant la recherche de l'instruction suivante, le compteur programme sauvegardé a une valeur voisine de l'instruction en cours, même si cette instruction est une instruction de branchement, de branchement inconditionnel, ou une instruction de retour. En plus des informations habituelles, le processeur sauvegarde sa copie interne du premier mot de l'instruction venant d'être traitée, et l'adresse qui a été accédée par le cycle bus avorté. Des informations spécifiques concernant l'accès sont aussi sauvegardées, indiquant si le processeur était en lecture ou en écriture, s'il était ou non en train d'exécuter une instruction. La valeur placée sur les sorties codes fonction quand l'erreur bus est survenue est sauvegardée. Le processeur traite une instruction s'il est dans l'état normal ou en train de traiter une exception de groupe 2. Le processeur ne traite pas une instruction s'il traite une exception de groupe 0 ou de groupe 1. La figure 30 illustre comment ces informations sont organisées sur la pile superviseur.

FIGURE 30 – COMMANDE PILE SUPERVISEUR



Quoique ces informations ne soient pas suffisantes en général pour recouvrer l'ensemble du contexte depuis l'erreur bus, elles permettent réellement un diagnostic logiciel. Finalement, le processeur commence le traitement d'instruction à l'adresse contenue dans le vecteur, le sous-programme de traitement d'erreur bus a la responsabilité de mise en ordre de la pile et de détermination du point de reprise d'exécution du programme.

Si une erreur bus survient au cours du traitement d'exception d'erreur bus, d'erreur adresse ou d'initialisation, le processeur est arrêté, et tout traitement cesse. Ceci simplifie la détection de pannes du système, le processeur se retirant lui même du système plutôt que de détruire tous les contenus mémoire, seule la broche RESET peut redémarrer un processeur arrêté.

ERREUR ADRESSE. Les exceptions d'erreur adresse surviennent lorsque le processeur tente d'accéder un opérande dont la longueur est un mot simple ou double, ou une instruction à une adresse impaire. L'effet est voisin d'une erreur bus générée de façon interne, de telle sorte que le cycle bus est avorté, et que le processeur cesse le traitement en cours quelque soit ce traitement, et commence un traitement d'exception. Une fois que le traitement d'exception a commencé, la séquence est identique à celle d'une erreur bus, y compris les informations qui sont sauvegardées, à l'exception du numéro de vecteur qui fait référence au vecteur d'erreur adresse. De la même façon, si une erreur d'adresse survient pendant le traitement d'exception d'une erreur bus, erreur adresse, ou d'initialisation, le processeur est arrêté.

INTERFACE AVEC LES PERIPHERIQUES EF6800

La famille des périphériques EFCIS EF6800 est directement compatible avec le circuit EF68000.

Les plus utilisés d'entre eux sont :

EF6821	Adapteur d'interface périphérique (PIA)
EF6840	Module temporisateur programmable (PTM)
EF6843	Contrôleur de disque souple (FDC)
EF6845	Contrôleur de visualisation sur écran (CRTC)
EF6850	Adaptateur d'interface de communication asynchrone (ACIA)
EF6852	Interface adapteur pour communications série synchrone (SSDA)
EF6854	Contrôleur de communications avancé (ADLC)
EF68488	Interface adapteur au bus GPIB (HP)

Pour interfacer les périphériques synchrones EF6800 avec le circuit asynchrone EF68000, le processeur modifie son cycle bus pour satisfaire aux exigences du cycle EF6800 chaque fois qu'une adresse du circuit EF6800 est détectée. Ceci est possible car les deux processeurs utilisent des E/S en configuration mémoire. La figure 31 montre l'organigramme de fonctionnement de l'interface entre le processeur et les circuits EF6800.

OPERATION DE TRANSFERT DES DONNEES.

Trois signaux du processeur fournissent l'interface du circuit EF6800. Ce sont les signaux : validation (E), validation adresse mémoire ($\overline{VMA}$), et validation adresse périphérique ($\overline{VPA}$). La figure 32 montre le diagramme des temps requis par les périphériques EF6800, celui du circuit EF6800, et celui du circuit EF68000.

Pour plus de détails sur le chronogramme des périphériques consulter les fiches techniques d'EFCIS.

Remarquer que le signal $\overline{VMA}$ du circuit EF68000 est actif à l'état bas, contrairement au signal $\overline{VMA}$ du circuit EF6800 actif à l'état haut. Ceci permet au processeur de placer ses bus à l'état haute impédance sur des demandes de $\overline{DMA}$ sans sélectionner par erreur des périphériques.

Le signal validation correspond au signal E ou $\phi 2$ des systèmes EF6800 existants. La synchronisation des transferts de données est faite par l'horloge bus utilisée par les périphériques EF6800. Le signal de validation est une horloge à fréquence fixe égale au dixième de la fréquence d'horloge en entrée du circuit EF68000. Le diagramme des temps de E permet d'utiliser des périphériques 1 MHz avec un circuit EF68000 à 8 MHz. Le signal E a un rapport cyclique de 60/40 ; c'est à dire qu'il est à l'état bas pour six périodes d'horloge d'entrée et à l'état haut pour quatre périodes d'horloge. Ce rapport cyclique permet au processeur de réaliser deux accès successifs VPA sur des impulsions E successives.

FIGURE 31 - ORGANIGRAMME D'INTERFACE DU CIRCUIT EF6800

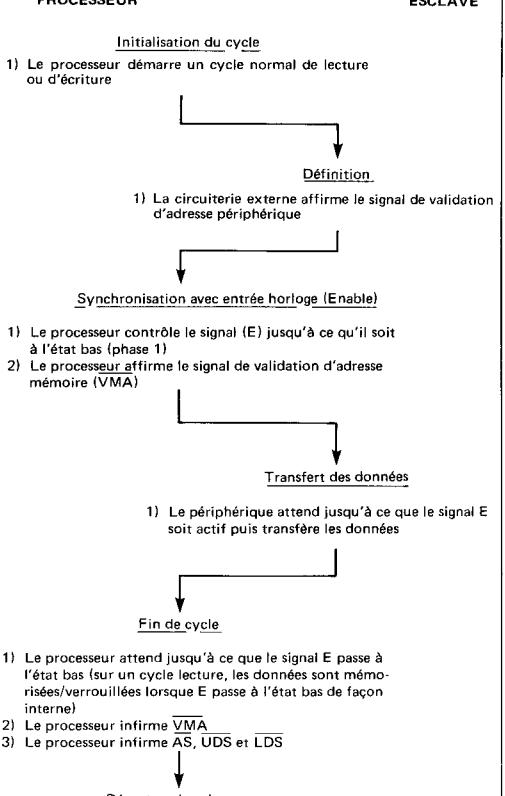
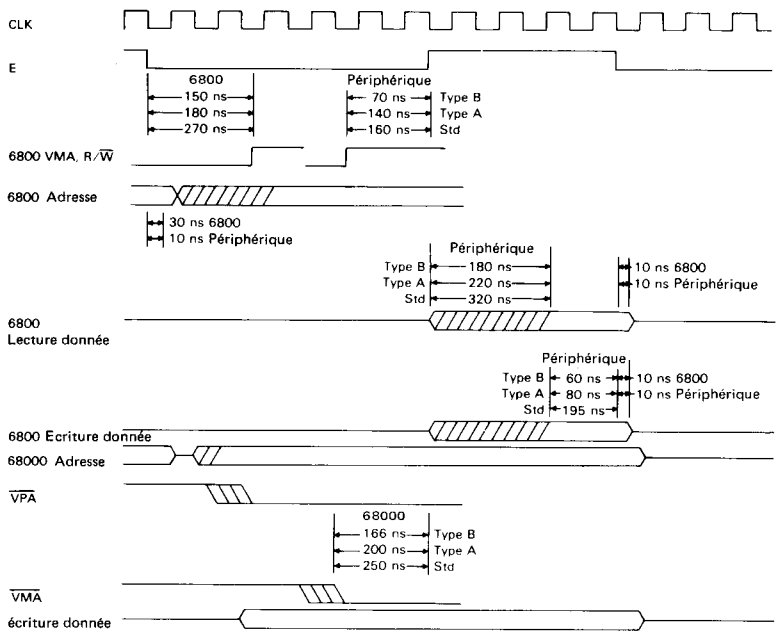


FIGURE 32 – DIAGRAMME DES TEMPS SUR L'UTILISATION DE PERIPHERIQUES EF6800 AVEC LE CIRCUIT EF68000



NOTE : Les temps sont donnés pour différentes fréquences d'horloge

Fréq. horl.	F-TYPE
2.0 MHz	B
1.5 MHz	A
1.0 MHz	Std

Temps de déplacement supplémentaire

F-TYPE	6800	68000
B	30 ns	96 ns
A	14 ns	60 ns
Std	70 ns	90 ns

L'entrée $\overline{VPA}$ signale au processeur que l'adresse sur le bus est l'adresse d'un circuit EF6800 (ou un espace réservé aux circuits EF6800) et que le bus doit se conformer aux caractéristiques de transfert ϕ 2 du bus EF6800. Le signal validation d'adresse périphérique est obtenu par décodage du bus adresse et est conditionné par le signal d'échantillonnage d'adresse.

Comme conséquence de la réception de $\overline{VPA}$, le processeur établit $\overline{VMA}$ avec une synchronisation appropriée lorsque E est à l'état bas. La validation d'adresse mémoire est utilisée dans l'équation de sélection du boîtier périphérique. Ceci assure la sélection et la désélection des périphériques à un instant correct.

Comme il est montré, les temps d'établissement et de maintien fournis par le circuit EF68000 sont meilleurs qu'il n'est nécessaire. Le temps de déplacement supplémentaire est le retard qui peut être admis sur la ligne $\overline{VMA}$. Il est défini comme étant la différence entre le temps d'établissement de $\overline{VMA}$ fourni par le processeur et celui requis par le périphérique.

TRAITEMENT DES INTERRUPTIONS

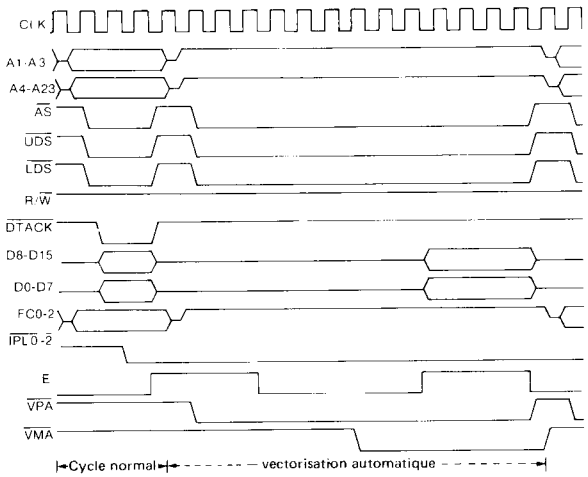
Au cours d'un cycle de reconnaissance d'interruption, lorsque le processeur va chercher le vecteur, si $\overline{VPA}$ est affirmé, le circuit EF68000 affirme $\overline{VMA}$ et complète un cycle lecture EF6800 normal, comme indiqué en figure 33. Le processeur utilise alors un vecteur généré de manière interne qui est fonction de l'interruption prise en compte. Ce procédé est connu sous le terme vectorisation automatique. Les sept "autovecteurs" sont des numéros de vecteurs allant de 25 à 31 (décimal).

Ceci se passe de la même manière (mais sans que ce soit restrictif) que pour la séquence d'interruption du circuit EF6800. La différence de base réside dans le fait qu'il y a six vecteurs d'interruption normaux et un vecteur de type NMI. Les sous-programmes d'interruption peuvent être mis n'importe où dans l'espace mémoire.

Ceci est du au fait que lorsque les numéros de vecteurs sont fixés, les contenus de la table vecteur sont attribués par l'utilisateur.

Le signal $\overline{VMA}$ étant affirmé pendant la vectorisation automatique, le décodage d'adresse périphérique EF6800 doit empêcher les accès non voulus.

FIGURE 33 – DIAGRAMME DES TEMPS DU FONCTIONNEMENT DE LA VECTORISATION AUTOMATIQUE



JEU D'INSTRUCTIONS

Les paragraphes suivants apportent des informations concernant les catégories d'adressage et jeu d'instructions du circuit EF68000.

CATEGORIES D'ADRESSAGE

Les modes d'adressage effectif peuvent être groupés selon leurs utilisations. Les classifications suivantes sont utilisées dans les définitions des instructions.

Données — Si un mode d'adressage effectif peut être utilisé pour adresser des opérandes données, il est considéré comme un mode d'adressage données effectif.

Mémoire — Si un mode d'adressage effectif peut être utilisé pour adresser des opérandes en mémoire, il est considéré comme un mode d'adressage mémoire effectif.

Modifiable — Si un mode d'adressage effectif peut être utilisé pour adresser des opérandes modifiables (par écriture), il est considéré comme un mode d'adressage modifiable effectif.

Commande — Si un mode d'adressage effectif peut être utilisé pour adresser des opérandes mémoire sans format associé, il est considéré comme un mode d'adressage commande effectif.

Le tableau 20 montre les différentes catégories auxquelles chacun des modes d'adressage effectif appartient. Le tableau 21 est un résumé du jeu d'instructions.

Le mode d'adressage registre d'état n'est pas autorisé, sauf s'il est explicitement mentionné comme mode d'adressage autorisé.

Ces catégories peuvent être combinées, de sorte que des classifications complémentaires, plus restrictives puissent être définies. Par exemple, les descriptions des instructions utilisent de telles classifications comme modifiable mémoire ou modifiable données. Le premier s'adresse aux modes d'adressage qui sont à la fois modifiables et d'adresses mémoire, et le dernier s'adresse aux modes d'adressage qui sont à la fois données et modifiables.

TABLEAU 20 – CATEGORIES DES MODES D'ADRESSAGE EFFECTIF

Modes d'adressage effectif	Mode	Registre	Catégories d'adressage			
			Données	Mémoire	Commande	Modifiable
Dn	000	numéro de registre	X			X
An	001	numéro de registre				X
An@	010	numéro de registre	X	X	X	X
An@+	011	numéro de registre	X	X		X
An@–	100	numéro de registre	X	X		X
An@(d)	101	numéro de registre	X	X	X	X
An@(d, ix)	110	numéro de registre	X	X	X	X
xxxW	111	000	X	X	X	X
xxxL	111	001	X	X	X	X
PC@(d)	111	010	X	X	X	
PC@(d, ix)	111	011	X	X	X	
# xxx	111	100	X	X		

TABLEAU 21 – JEU D'INSTRUCTION

Mnémonique	Description	Fonctionnement	Codes condition				
			X	N	Z	V	C
ABCD	Addition décimale étendue	(Destination)$_{10}$ + (Source)$_{10}$ → Destination	*	U	*	U	*
ADD	Addition binaire	(Destination) + (Source) → Destination	*	*	*	*	*
ADDA	Addition d'adresse	(Destination) + (Source) → Destination	—	—	—	—	—
ADDI	Addition immédiate	(Destination) + données immédiates → Destination	*	*	*	*	*
ADDQ	Addition rapide	(Destination) + données immédiates → Destination	*	*	*	*	*
ADDX	Addition étendue	(Destination) + (Source) + X → Destination	*	*	*	*	*
AND	ET logique	(Destination) $\wedge$ (Source) → Destination	—	*	*	0	0
ANDI	ET immédiat	(Destination) $\wedge$ données immédiates → Destination	—	*	*	0	0
ASL, ASR	Décalage arithmétique	(Destination) décalé par < compte > → Destination	*	*	*	*	*
Bcc	Branchement conditionnel	Si CC alors PC + d → PC	—	—	—	—	—
BCHG	Test de bit et changement	~ (< numéro de bit >) de destination → Z ~ (< numéro de bit >) de destination ~ < numéro de bit > de destination	—	—	*	—	—
BCLR	Test de bit et mise à zéro	~ (< numéro de bit >) de destination → Z 0 → < numéro de bit > → de destination	—	—	*	—	—
BRA	Branchement inconditionnel	PC + d → PC	—	—	—	—	—
BSET	Test de bit et mise à un	~ (< numéro de bit >) de destination → Z 1 → < numéro de bit > de destination	—	—	*	—	—
BSR	Branchement à un sous-prog.	PC → SP@ – ; PC + d → PC	—	—	—	—	—
BTST	Test de bit	~ (< numéro de bit >) de destination → Z	—	—	*	—	—
CHK	Test registre aux limites	Si Dn < 0 ou Dn > (< ea >) alors TRAP	—	*	U	U	U
CLR	Mise à zéro de l'opérande	0 → Destination	—	0	1	0	0
CMP	Comparaison	(Destination) – (Source)	—	*	*	*	*
CMPA	Comparaison d'adresse	(Destination) – (Source)	—	*	*	*	*
CMPI	Comparaison immédiate	(Destination) – Données immédiates	—	*	*	*	*
CMPM	Comparaison mémoire	(Destination) – Source	—	*	*	*	*
DBcc	Test condition, décrémentation et branchement	Si ~ CC alors Dn – 1 → Dn ; si Du = – 1 alors PC + d → PC	—	—	—	—	—
DIVS	Division signée	(Destination) / (Source) → Destination	—	*	*	*	0
DIVU	Division non signée	(Destination) / (Source) → Destination	—	*	*	*	0
EOR	OU exclusif logique	(Destination) ⊕ (Source) → Destination	—	*	*	0	0
EORI	OU exclusif immédiat	(Destination) ⊕ Données immédiates → Destination	—	*	*	0	0
EXG	Echange de registres	Rx ↔ Ry	—	—	—	—	—
EXT	Extension de signe	(Destination) signe étendu → Destination	—	*	*	0	0
JMP	Saut inconditionnel	Destination → PC	—	—	—	—	—
JSR	Saut à un sous-programme	PC → SP@ – ; Destination → PC	—	—	—	—	—
LEA	Chargement adresse effective	Destination → An	—	—	—	—	—
LINK	Chaînage de pile	An → SP@ – ; SP → An ; SP + d → SP	—	—	—	—	—
LSL, LSR	Décal. logique gauche/droite	(Destination) décalé de < compte > → Destination	*	*	*	0	*
MOVE	Transfert de données de source à destination	(Source) → Destination	—	*	*	0	0
MOVE → CCR	Transfert vers CCR	(Source) → CCR	*	*	*	*	*
MOVE → SR	Transfert vers registre d'état	(Source) → SR	*	*	*	*	*

* affecté 0 mis à zéro U indéfini
– non affecté 1 mis à un

TABLEAU 21 – JEU D'INSTRUCTION

Mnémonique	Description	Fonctionnement	Codes condition				
			X	N	Z	V	C
MOVE depuis SR	Transfert du registre d'état	SR → Destination	—	—	—	—	—
MOVE USP	Transf. pointeur de pile utilisateur	USP → An ; An → USP	—	—	—	—	—
MOVEA	Transfert adresse	(Source) → Destination	—	—	—	—	—
MOVEM	Transfert registres multiples	Registres → Destination (Source) → Registres	—	—	—	—	—
MOVEP	Transfert données périphériques	(Source) → Destination	—	—	—	—	—
MOVEQ	Transfert rapide	Données immédiates → Destination	—	*	*	0	0
MULS	Multiplication signée	(Destination)* (Source) → Destination	—	*	*	0	0
MULU	Multiplication non signée	(Destination)* (Source) → Destination	—	*	*	0	0
NBCD	Prendre l'opposé décimal étendu	$0 - (\text{Destination})_{10} - X \rightarrow$ Destination	*	U	*	U	*
NEG	Complémentation à 2	0 – (Destination) → Destination	*	*	*	*	*
NEGX	Complémentation à 2 étendue	0 – (Destination) – X → Destination	*	*	*	*	*
NOP	Non opération		—	—	—	—	—
NOT	Complément logique	~ (Destination) → Destination	—	*	*	0	0
OR	OU logique inclusif	(Destination) v (Source) → Destination	—	*	*	0	0
ORI	OU logique immédiat	(Destination) v données immédiates → Destination	—	*	*	0	0
PEA	Empilement adresse effective	Destination → SP@–	—	—	—	—	—
RESET	Mise à zéro des circuits externes		—	—	—	—	—
ROL, ROR	Décalage circulaire non étendu	(Destination) décalé de < compte > → Destination	—	*	*	0	*
ROXL, ROXR	Décalage circulaire étendu	(Destination) décalé de < compte > → Destination	*	*	*	0	*
RTE	Retour d'exception	SP@+ → SR ; SP@+ → PC	*	*	*	*	*
RTR	Retour et restaure codes condition	SP@+ → CC ; SP@+ → PC	*	*	*	*	*
RTS	Retour de sous-programme	SP@+ →	—	—	—	—	—
SBCD	Soustraction décimale étendue	$(\text{Destination})_{10} - (\text{Source})_{10} - X \rightarrow$ Destination	*	U	*	U	*
Scc	Mise à un conditionnelle	Si CC vrai alors 1 dans la destination Si CC faux alors 0 dans la destination	—	—	—	—	—
STOP	Chargement registre d'état et suspension d'exécution du prog.	Données immédiates SR ; STOP	*	*	*	*	*
SUB	Soustraction	(Destination) – (Source) → Destination	*	*	*	*	*
SUBA	Soustraction	(Destination) – (Source) → Destination	—	—	—	—	—
SUBI	Soustraction immédiate	(Destination) – (Données immédiates) → Destination	*	*	*	*	*
SUBQ	Soustraction rapide	(Destination) – Données immédiates → Destination	*	*	*	*	*
SUBX	Soustraction étendue	(Destination) – (Source) – X → Destination	*	*	*	*	*
SWAP	Echanger les moitiés de registre	Registre [31 : 16] ↔ [15 : 0]	—	*	*	0	0
TAS	Test opér. et établ. d'1 indicateur	(Destination) testée → CC ; 1 → [7] OF Destination	—	*	*	0	0
TRAP	Trappe	PC → SSP@– ; SR → SSP@– ; (Vecteur) → PC	—	—	—	—	—
TRAPV	Trappe sur dépassement	Si V alors TRAP	—	—	—	—	—
TST	Test d'un opérande	(Destination) testé → CC	—	*	*	0	0
UNLK	Rupture du lien	AN → SP ; SP@+ → An	—	—	—	—	—

[ ] = numéro du bit

TEMPS D'EXECUTION DES INSTRUCTIONS

Les paragraphes suivants contiennent les listes des temps d'exécution en termes de nombre de cycles internes requis pour l'exécution d'une instruction. Un cycle interne est égal à deux cycles de l'entrée horloge du processeur. On suppose aussi que la période d'un cycle mémoire n'est pas supérieur à quatre cycles d'entrée horloge du processeur pour empêcher l'introduction d'états d'attente dans le cycle bus.

Ces temps d'exécution concernent l'instruction complète, y compris les recherches des opérandes, les écritures des opérandes et les lectures des instructions.

TEMPS DE CALCUL DE L'ADRESSE EFFECTIVE D'UN OPERANDE

Le tableau 22 donne le temps global requis pour le calcul de l'adresse effective d'un opérande. Le temps inclut la recherche de tous les mots d'extension, le calcul d'adresse, et la recherche de l'opérande en mémoire.

TEMPS D'EXECUTION DES INSTRUCTIONS DE TRANSFERT (MOVE)

Les tableaux 23 et 24 donnent les temps d'exécution de l'instruction de transfert (MOVE). Ces temps d'exécution concernent l'instruction entière et incluent les lectures opérandes ainsi que les écritures, et les lectures instruction.

TABLEAU 22 – TEMPS DE CALCUL DE L'ADRESSE EFFECTIVE D'UN OPERANDE

Mode d'adressage		TEMPS	
		octet, mot	mot long.
	Registre		
Dn	Registre données direct	0	0
An	Registre adresse direct	0	0
	Mémoire		
An@	Registre adresse indirect	2	4
An@+	Registre adresse indirect avec postincrément	2	4
An@-	Registre adresse indirect avec prédécrément	3	5
An@(d)	Registre adresse indirect avec déplacement	4	6
An@(d, ix)	Registre adresse indirect avec index	5	7
xxx W	Absolu court	4	6
xxx L	Absolu long	6	8
PC@(d)	Compteur programme avec déplacement	4	6
PC@(d, ix)	Programme compteur avec index	5	7
#xxx	Immédiat	2	4

TABLEAU 23 – TEMPS D'EXECUTION DE L'INSTRUCTION DE TRANSFERT (MOVE) SUR OCTET ET SUR MOT

Source	Destination								
	Dn	An	An@	An@+	An@-	An@(d)	An@(d, ix)	xxx.W	xxx.L
Dn	2	2	4	4	4	6	7	6	8
An	2	2	4	4	4	6	7	6	8
An@	4	4	6	6	6	8	9	8	10
An@+	4	4	6	6	6	8	9	8	10
An@-	5	5	7	7	7	9	10	9	11
An@(d)	6	6	8	8	8	10	11	10	12
An@(d, ix)	7	7	9	9	9	11	12	11	13
xxx W	6	6	8	8	8	10	11	10	12
xxx.L	8	8	10	10	10	12	13	12	14
PC@(d)	6	6	8	8	8	10	11	10	12
PC@(d, ix)	7	7	9	9	9	11	12	11	13
#xxx	4	4	6	6	6	8	9	8	10

Les instructions de transfert (MOVE) sur octet et sur mot nécessitent chacune un accès mémoire pour la lecture et pour l'écriture de l'opérande.

TABLEAU 24 – TEMPS D'EXECUTION DES INSTRUCTIONS DE TRANSFERT SUR MOT LONG

Source	Destination								
	Dn	An	An@	An@+	An@-	An@(d)	An@(d, ix)	xxx.w	xxx.L
Dn	2	2	6	6	6	8	9	8	10
An	2	2	6	6	6	8	9	8	10
An@	6	6	10	10	10	12	13	12	14
An@+	6	6	10	10	10	12	13	12	14
An@-	7	7	11	11	11	13	14	13	15
An@(d)	8	8	12	12	12	14	15	14	16
An@(d, ix)	9	9	13	13	13	15	16	15	17
xxx.W	8	8	12	12	12	14	15	14	16
xxx L	10	10	14	14	14	16	17	16	18
PC@(d)	8	8	12	12	12	14	15	14	16
PC@(d, ix)	9	9	13	13	13	15	16	15	17
#xxx	6	6	10	10	10	12	13	12	14

Note : L'instruction de transfert (MOVE) sur mot long nécessite 2 accès mémoire pour la lecture de l'opérande et aussi pour l'écriture de l'opérande.

TEMPS D'EXECUTION D'UNE INSTRUCTION STANDARD

Les temps montrés, tableau 25, incluent le temps de réaliser l'opération, mémoriser les résultats et lire l'instruction suivante. Le temps de calcul d'adresse et de recherche de l'adresse effective doit être ajouté lorsqu'il est indiqué. Dans le tableau 25, les "en-tête" ont les significations suivantes : A = opérande registre adresse, D = opérande registre données, E = un opérande spécifié par une adresse effective, et M = opérande mémoire.

TEMPS D'EXECUTION D'UNE INSTRUCTION IMMEDIATE

Les temps montrés au tableau 26 incluent le temps de recherche des opérandes immédiats, de réalisation des opérations, de mémorisation des résultats, et de lecture de l'opération suivante. Dans le tableau 26, les rubriques ont les significations suivantes : D = opérande de registre données, I = opérande immédiat, M = opérande de mémoire, SR = registre d'état.

TABLEAU 25 – TEMPS D'EXECUTION DES INSTRUCTIONS STANDARDS

Instruction	Taille	A op E →A	D op E →D	M op D →M
ADD	octet, mot	4+	2+	4+
	mot long	3#+	3#+	6+
AND	octet, mot	—	2+	4+
	mot long	—	3#+	6+
CMP	octet, mot	3+	2+	—
	mot long	3+	3+	—
DIVS	—	—	79+*	—
DIVU	—	—	70+*	—
EOR	octet, mot	—	2	4+
	mot long	—	3#	5+
MULS	—	—	35+*	—
MULU	—	—	35+*	—
OR	octet, mot	—	2+	4+
	mot long	—	3#+	6+
SUB	octet, mot	4+	2+	4+
	mot long	3#+	3#+	6+

\+ ajouter le temps de calcul de l'adresse effective
\# 4 si l'adresse effective est registre direct
* indique la valeur maximale.

TABLEAU 26 – TEMPS D'EXECUTION DES INSTRUCTIONS IMMEDIATES

Instruction	Taille	D op I →D	M op I →M	SR op I →SR
ADDI	octet, mot	4	6+	—
	mot long	8	10+	—
ADDQ	octet, mot	2	4+	—
	mot long	4	6+	—
ANDI	octet, mot	4	6+	10
	mot long	8	10+	—
CMPI	octet, mot	4	4+	—
	mot long	7	6+	—
EORI	octet, mot	4	6+	10
	mot long	8	10+	—
MOVEQ	mot long	2	—	—
ORI	octet, mot	4	6+	10
	mot long	8	10+	—
SUBI	octet, mot	4	6+	—
	mot long	8	10+	—
SUBQ	octet, mot	2	4+	—
	mot long	4	6+	—

\+ ajouter le temps de calcul de l'adresse effective.

TEMPS D'EXECUTION DES INSTRUCTIONS SIMPLE OPERANDE

(voir tableau 27).

TABLEAU 27 – TEMPS D'EXECUTION DES INSTRUCTIONS SIMPLE OPERANDE

Instruction	Taille	Registre	Mémoire
CLR	octet, mot mot long	2 3	4+ 6+
NBCD	octet	3	4+
NEG	octet, mot mot long	2 3	4+ 6+
NEGX	octet, mot mot long	2 3	4+ 6+
NOT	octet, mot mot long	2 3	4+ 6+
Scc	octet, faux octet, vrai	2 3	4+ 4+
TAS	octet	2	5+
TST	octet, mot mot long	2 2	2+ 2+

\+ ajouter le temps de calcul de l'adresse effective.

TEMPS D'EXECUTION DES INSTRUCTIONS DE DÉCALAGE ET DE ROTATION

(Voir tableau 28)

TABLEAU 28 – TEMPS D'EXECUTION DES INSTRUCTIONS DE DECALAGE ET DE ROTATION

Instruction	Taille	Registre *	Mémoire
ASR, ASL	octet, mot mot long	3 + n 4 + n	4+ —
LSR, LSL	octet, mot mot long	3 + n 4 + n	4+ —
ROR, ROL	octet, mot mot long	3 + n 4 + n	4+ —
ROXR, ROXL	octet, mot mot long	3 + n 4 + n	4+ —

\+ ajouter le temps de calcul de l'adresse effective.
*n = nombre de décalages

TEMPS D'EXECUTION DES INSTRUCTIONS DE MANIPULATION SUR LES BITS.

(Voir tableau 29)

TABLEAU 29 – TEMPS D'EXECUTION DES INSTRUCTIONS SUR BIT

		Dynamique		Statique	
Instruction	Taille	registre	mémoire	registre	mémoire
BCHG	mot	—	4+	—	6+
	mot long	4*	—	6*	—
BCLR	mot	—	4+	—	6+
	mot long	5*	—	7*	—
BSET	mot	—	4+	—	6+
	mot long	4*	—	6*	—
BTST	mot	—	2+	—	4+
	mot long	3	—	5	—

\+ ajouter le temps de calcul de l'adresse effective
* indique la valeur maximale

TEMPS D'EXECUTION DES INSTRUCTIONS CONDITIONNELLES

(voir tableau 30)

TABLEAU 30 – TEMPS D'EXECUTION DES INSTRUCTIONS CONDITIONNELLES

Instruction	Déplacement	(TRAP)	(No TRAP)
Bcc	octet mot	5 5	4 6
BRA	octet mot	5 5	— —
BSR	octet mot	9 9	— —
DBcc	—	5	7
CHK	—	19*+	4+
TRAP	—	16	—
TRAPV	—	16	2

\+ ajouter le temps de calcul de l'adresse effective.
* indique la valeur maximale.

TEMPS D'EXECUTION DES INSTRUCTIONS JMP, JSR, LEA, PEA, MOVEM

(Voir tableau 31)

TABLEAU 31 – TEMPS D'EXECUTION DES INSTRUCTIONS JMP, JSR, LEA, PEA, MOVEM

Instruction	Taille	An@	An@+	An@-	An@(d)	An@(d, ix)	xxx.W	xxx.L	PC@(d)	PC@(d, ix)
JMP	—	4	—	—	5	7	5	6	5	7
JSR	—	8	—	—	9	11	9	10	9	11
LEA	—	2	—	—	4	6	4	6	4	6
MOVEM	mot	6 + 2n	6 + 2n	—	8 + 2n	9 + 2n	8 + 2n	10 + 2n	8 + 2n	9 + 2n
M ►R	long	6 + 4n	6 + 4n	—	8 + 4n	9 + 4n	8 + 4n	10 + 4n	8 + 4n	9 + 4n
MOVEM	mot	4 + 2n	—	4 + 2n	6 + 2n	7 + 2n	6 + 2n	8 + 2n	—	—
R ►M	long	4 + 4n	—	4 + 4n	6 + 4n	7 + 4n	6 + 4n	8 + 4n	—	—
PEA		6	—	—	8	9	8	10	8	9

n : nombre de registres à transférer.

TEMPS D'EXECUTION DES EXCEPTIONS

(Voir tableau 32).

Ces temps comprennent l'empilement automatique, la recherche du vecteur et la recherche de la première instruction du programme de traitement.

TABLEAU 32 – TEMPS D'EXECUTION DES EXCEPTIONS

Exception	Tps
Erreur adresse	24
Erreur bus	24
Interruption	21*
Instruction illégale	16
Instruction privilégiée	16
Trace	16

* on suppose que la reconnaissance d'interruption prend 2 cycles

TEMPS D'EXECUTION D'INSTRUCTIONS MULTI-PRECISION

(Voir tableau 33).

Ces temps comprennent le temps nécessaire pour rechercher les 2 opérandes, pour réaliser les opérations, ranger les résultats et lire les instructions suivantes.

TABLEAU 33 – TEMPS D'EXECUTION D'INSTRUCTION MULTI-PRECISION

Instruction	Taille	D op D ►D	M op M ►M
ADDX	octet, mot	2	9
	mot long	4	15
CMPM	octet, mot	—	6
	mot long	—	10
SUBX	octet, mot	2	9
	mot long	4	15
ABCD	octet	3	9
SBCD	octet	3	9

TEMPS D'EXECUTION D'INSTRUCTIONS SPECIALES

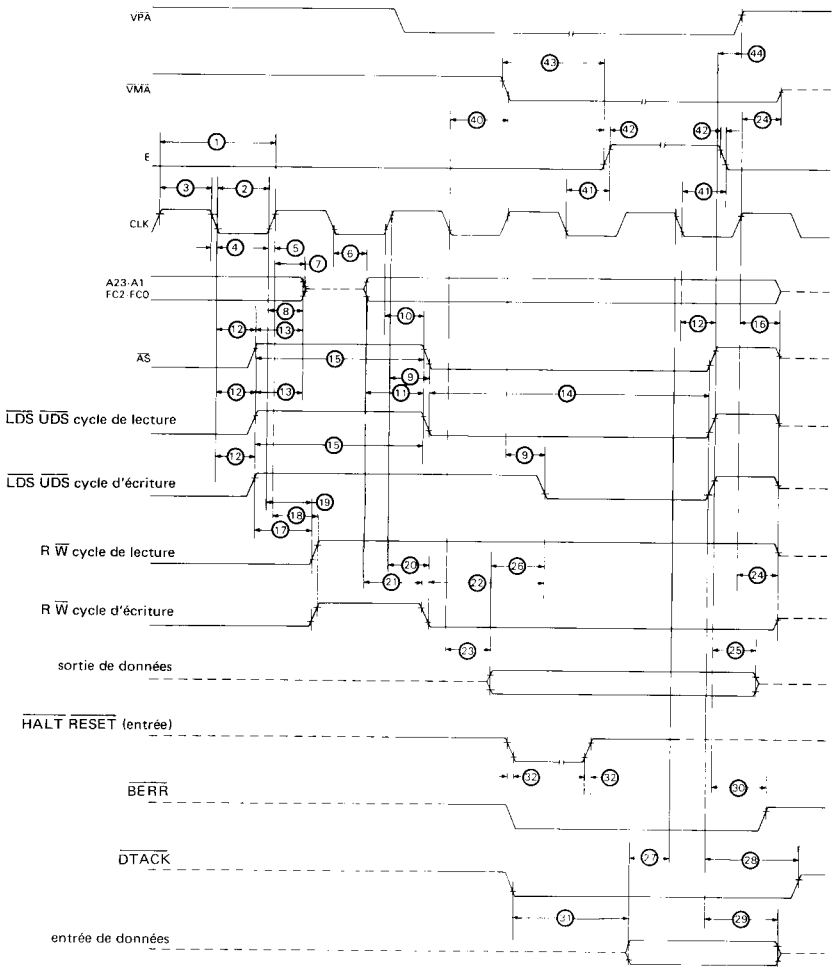
(Voir tableau 34).

TABLEAU 34 - TEMPS D'EXECUTION D'INSTRUCTIONS SPECIALES

Instruction	Taille	Reg.	Mem.	Reg. ►Mem.	Mem. ►Reg.
MOVE de SR	—	3	4+	—	—
MOVE vers CCR	—	6	6+	—	—
MOVE vers SR	—	6	6+	—	—
MOVEP	mot	—	—	8	8
	long	—	—	12	12
EXG	—	3	—	—	—
EXT	mot	2	—	—	—
	long	2	—	—	—
LINK	—	8	—	—	—
MOVE de USP	—	2	—	—	—
MOVE vers USP	—	2		—	—
NOP	—	2	—	—	—
RESET	—	66	—		
RTE	—	10	—	—	—
RTR	—	10	—		—
RTS	—	8	—	—	
STOP	—	2	—	—	—
SWAP	—	2			
UNLK	—	6		—	—

\+ ajouter le temps de calcul de l'adresse effective.

FORME DES SIGNAUX ELECTRIQUES



NOTES :

1. Ce diagramme des temps doit seulement être pris comme référence en tenant compte de la mesure des temps de transition à transition selon les spécifications données dans les caractéristiques et conditions de fonctionnement dynamiques. Il n'est pas supposé être une description fonctionnelle des entrées et des sorties. Se reporter à d'autres descriptions fonctionnelles et leurs diagrammes des temps associés pour le fonctionnement du circuit. Le fonctionnement du bus est supposé avoir quatre cycles LECTURE et une ECRITURE de quatre cycles dans ce diagramme.

2. Les mesures des formes de signaux sont ainsi spécifiées :
pour toutes les sorties telles que : V_{OH} = 2,4 volts
V_{OL} = 0,4 volt
pour toutes les entrées telles que : V_{IH} = 2,0 volts
V_{IL} = 0,8 volt

CARACTERISTIQUES ET CONDITIONS DE FONCTIONNEMENT DYNAMIQUES

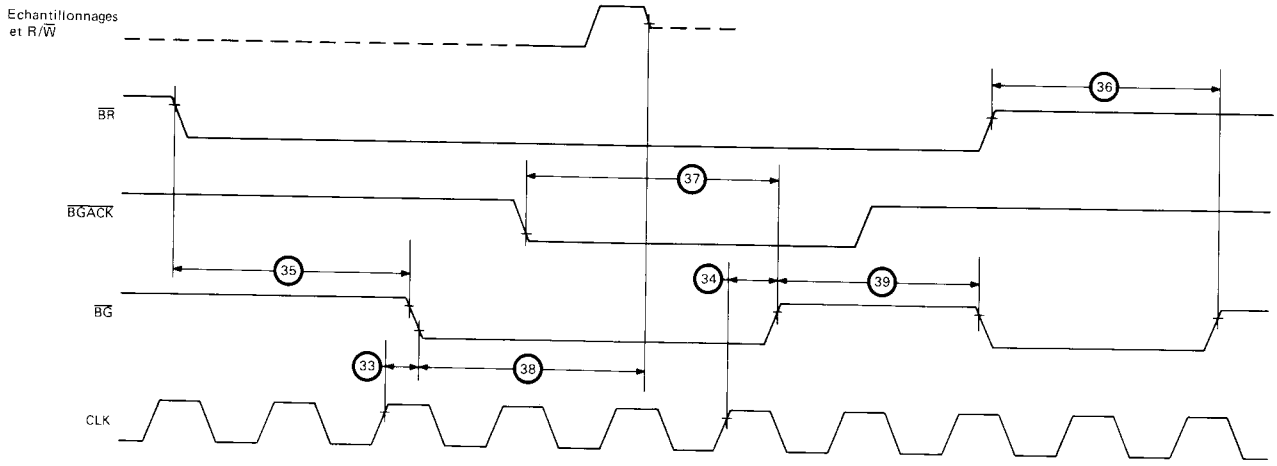
(V_{DD} = 5.0 V ± 5% ; V_{SS} = 0 V ; T_A = 25°C)

Numéro	Caractéristiques	Symboles	Min	Typ	Max	Unités
1	Période horloge	t_{CYC}	125	–	–	ns
2	Largeur d'impulsion d'horloge – niveau bas	t_{CL}	55	–	–	ns
3	Largeur d'impulsion d'horloge – niveau haut	t_{CH}	55	–	–	ns
4	Temps de descente du signal d'horloge	t_{Cf}	–	–	10	ns
5	Temps de montée du signal d'horloge	t_{Cr}	–	–	10	ns
6	De l'horloge basse à l'adresse / FC valide	t_{CLAV}	–	–	60	ns
7	De l'horloge haute à l'adresse ou FC à haute impédance (max)	t_{CHAZx}	–	–	60	ns
8	De l'horloge haute à l'adresse ou au FC invalide (min)	t_{CHAZn}	20	–	–	ns
9	De l'horloge haute à $\overline{AS}$, $\overline{DS}$ bas (max)	t_{CHSLx}	–	–	60	ns
10	De l'horloge haute à $\overline{AS}$, $\overline{DS}$ bas (min)	t_{CHSLn}	20	–	–	ns
11	De l'adresse / FC valide à $\overline{AS}$, $\overline{DS}$ (lecture) bas	t_{AVSL}	30*	–	–	ns
12	De l'horloge basse à $\overline{AS}$, $\overline{DS}$ haut	t_{CLSH}	–	–	40	ns
13	De $\overline{AS}$, $\overline{DS}$ haut à l'adresse / FC invalide	t_{SHAZ}	40*	–	–	ns
14	Largeur d'impulsion des signaux $\overline{AS}$, $\overline{DS}$ (mesurée à l'état bas) - lecture -	t_{SL}	150	–	–	ns
15	Largeur d'impulsion des signaux $\overline{AS}$, $\overline{DS}$ (mesurée à niveau haut)	t_{SH}	150	–	–	ns
16	De l'horloge haute à $\overline{AS}$, $\overline{DS}$ à haute impédance	t_{CHSZ}	–	–	60	ns
17	De $\overline{DS}$ haut à R/$\overline{W}$ haut	t_{SHRH}	60*	–	–	ns
18	De l'horloge haute à R/$\overline{W}$ haut (max)	t_{CHRHx}	–	–	60	ns
19	De l'horloge haute à R/$\overline{W}$ haut (min)	t_{CHRHn}	20	–	–	ns
20	De l'horloge haute à R/$\overline{W}$ bas	t_{CHRL}	–	–	60	ns
21	De l'adresse / FC valide à R/$\overline{W}$ bas	t_{AVRL}	50*	–	–	ns
22	De R/$\overline{W}$ bas à $\overline{DS}$ bas (écriture)	t_{RLSL}	80*	–	–	ns
23	De l'horloge basse aux données sortantes valides	t_{CLDO}	–	–	50	ns
24	De l'horloge haute à R/$\overline{W}$, $\overline{VMA}$ à haute impédance	t_{CHRZ}	–	–	60	ns
25	De $\overline{DS}$ haut aux données sortantes invalides	t_{SHDO}	30*	–	–	ns
26	Des données sortantes valides à $\overline{DS}$ bas (écriture)	t_{DOSL}	30*	–	–	ns
27	Des données entrantes à l'horloge basse (temps d'établissement)	t_{DICL}	30	–	–	ns
28	De $\overline{DS}$ haut à $\overline{DTACK}$ haut	t_{SHDAH}	0	–	120	ns
29	De $\overline{DS}$ haut aux données entrantes (temps de maintien)	t_{SHDI}	0	–	–	ns
30	De $\overline{AS}$, $\overline{DS}$ hauts à $\overline{BERR}$ haut	t_{SHBEH}	0	–	–	ns
31	De $\overline{DTACK}$ bas aux données entrantes (temps d'établissement)	t_{DALDI}	–	–	90*	ns
32	Temps de montée et de descente des signaux $\overline{HALT}$ et $\overline{RESET}$	t_{RHrf}	–	–	200	ns
33	De l'horloge haute à $\overline{BG}$ bas	t_{CHGL}	–	–	60	ns
34	De l'horloge haute à $\overline{BG}$ haut	t_{CHGH}	–	–	60	ns
35	De $\overline{BR}$ basse à $\overline{BG}$ bas	t_{BRLGL}	15	–	30	**
36	De $\overline{BR}$ haut à $\overline{BG}$ haut	t_{BRHGH}	15	–	30	**
37	De $\overline{BGACK}$ basse à $\overline{BG}$ haut	t_{GALGH}	15	–	20	**
38	De $\overline{BG}$ bas au bus à haute impédance (avec $\overline{AS}$ haut)	t_{GLZ}	0	–	–	**
39	Durée de $\overline{BG}$ haut	t_{GH}	15	–	–	**
40	De l'horloge basse à $\overline{VMA}$ bas	t_{CLVML}	–	–	60	ns
41	De l'horloge basse à la transition de E	t_{CLE}	–	–	55	ns
42	Temps de montée et de descente du signal E d'activation	t_{Erf}	–	–	25	ns
43	De $\overline{VMA}$ bas à E haut	t_{VMLEH}	20	–	30	**
44	De $\overline{AS}$, $\overline{DS}$ à $\overline{VPA}$ haut	t_{SHVPH}	0	–	–	ns

*Valeur fonction de la période d'horloge. Ces chiffres sont fondés sur un fonctionnement à 8 MHz.

** Période d'horloge

FORME DES SIGNAUX ~ ARBITRAGE D'ATTRIBUTION DU BUS



CARACTERISTIQUES ET CONDITIONS DE FONCTIONNEMENT DYNAMIQUES

(V_{DD} = 5.0 V ± 5% ; V_{SS} = 0 V ; T_A = 25°C) **ARBITRAGE D'ATTRIBUTION DU BUS.**

Numéro	Caractéristiques	Symboles	Min	Typ	Max	Unités
33	de l'horloge haute à $\overline{BG}$ bas	t_{CHGL}	—	60	—	ns
34	de l'horloge haute à $\overline{BG}$ haut	t_{CHGH}	—	60	—	ns
35	de $\overline{BR}$ bas à $\overline{BG}$ bas	t_{BRLGL}	1.0	—	3.0	Période d'horloge
36	de $\overline{BR}$ haut à $\overline{BG}$ haut	t_{BRHGH}	1.0	—	3.0	
37	de $\overline{BGACK}$ bas à $\overline{BG}$ haut	t_{GALGH}	1.0	—	2.0	
38	de $\overline{BG}$ bas au passage du bus dans l'état haute impédance (avec $\overline{AS}$ haut)	t_{GLZ}	0	—	1.5	
39	largeur de $\overline{BG}$ haut	t_{GH}	1.5	—	—	

NOTES

Informations préliminaires : ces spécifications peuvent changer sans préavis.
Consultez notre réseau de vente pour connaître la disponibilité des différentes versions de ce circuit.